

# Chapter 3. Creating Reference Images

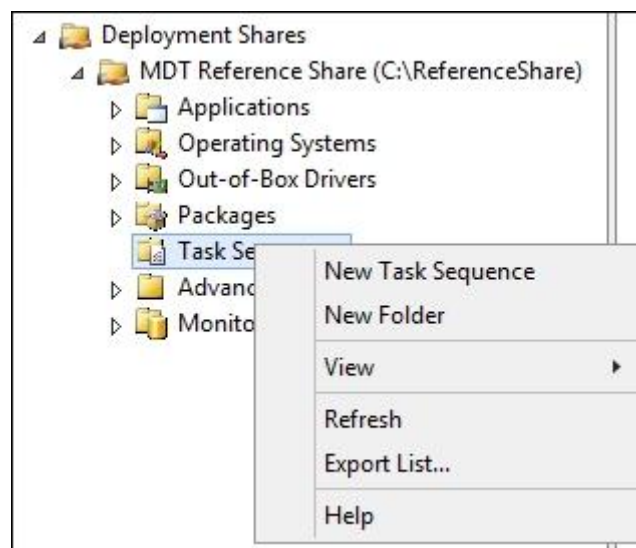
In the previous chapter, we constructed a reference share on our deployment host. This chapter utilizes this share to craft a task sequence that will be used to create a golden image for later deployment.

In this chapter, we will cover the following topics:

- Task sequence concepts
- Image customizations that can be carried on to the default user profile
- The image factory that will keep reference images up to date with the latest Windows Updates via a Windows Server Update Services configuration
- Sysprep and its run support
- Reference task sequence boot media

## Creating a reference image in the management console

In this chapter, we'll spend most of our time modifying the task sequence for **reference image creation**. To create one in the management console, select **Task Sequences**, as shown in the following image, and then select **New Task Sequence**:



## Specifying the general settings

Generally speaking, I recommend **Task sequence ID** to be specified as numerical values. This suites well when we force task sequences based on things such as model number, for instance. **Task sequence name** is better suited for a longer description of what the task sequences' purpose is. Comments, of course, can contain entries such as version control, author, and other operational details, as shown in the following screenshot:

General Settings

Select Template

Select OS

Specify Product Key

OS Settings

Admin Password

Summary

Progress

Confirmation

Specify general information about this task sequence. The task sequence ID is used internally as part of the deployment process. The task sequence name and comments are displayed by the deployment wizard.

Task sequence ID:

1

Task sequence name:

Reference Build for Windows 7 SP1 x64

Task sequence comments:

Version 1|

## Selecting the template and the OS

On the following screen, **Standard Client Task Sequence** works very well for our intentions here. In fact, the built-in task sequences have a lot of logic built into them, which is hard to reproduce. I recommend always using these as building blocks, instead of building your own from scratch:

General Settings

Select Template

Select OS

Specify Product Key

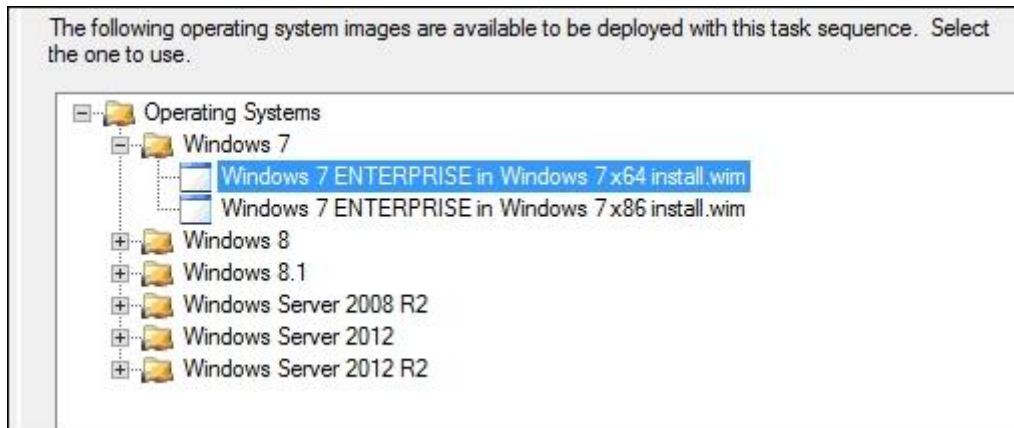
OS Settings

The following task sequence templates are available. Select the one you would like to use as a starting point.

Standard Client Task Sequence

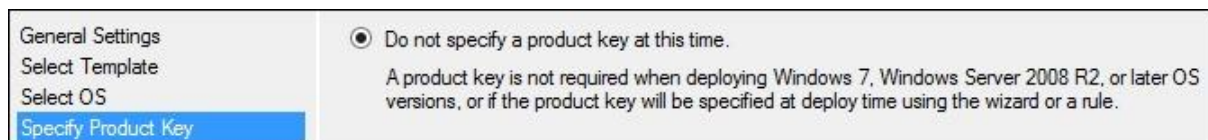
A complete task sequence for deploying a client operating system

In the following window, we will select our base OS used for the task sequence. On our **General Settings** screen, I indicated this would be a Windows 7 SP1 x64 Reference build, so let's select Windows 7 from the provided options:

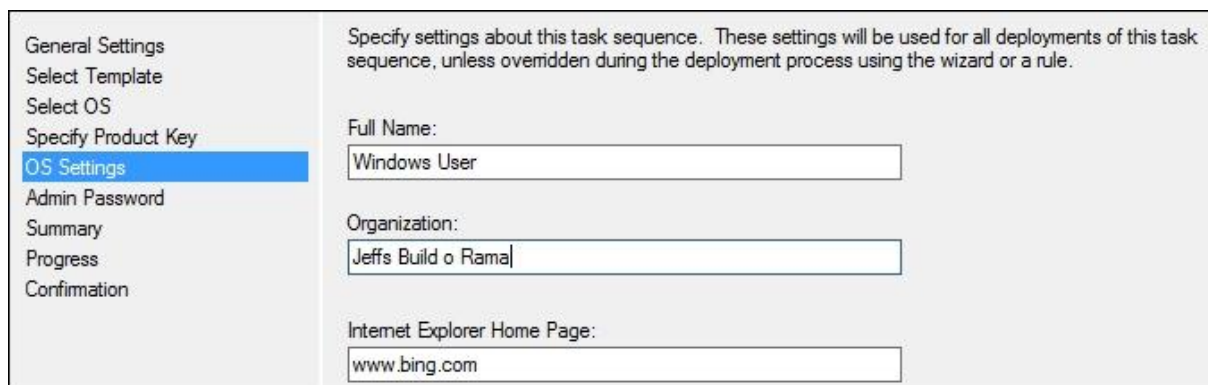


## Specifying a product key and OS settings

As we are not building a system per se, but rather constructing a reference image for future use, we do not need to specify a product key at this time:



Then, on the next page of the wizard, simply fill in the provided blanks. These are not permanent selections, they can be modified later, so do not worry about them at this point:



## Passwords and security

On the next screen, we specify an administrator account password for the reference image. Make this a fairly simply password, as the password is obfuscated, not encrypted in any strong way; mine will be P@ssword1, for example:

General Settings	Specify the local Administrator password for this task sequence.
Select Template	
Select OS	
Specify Product Key	
OS Settings	
<b>Admin Password</b>	
Summary	
Progress	
Confirmation	<input checked="" type="radio"/> Use the specified local Administrator password. Administrator Password: <input type="password" value="....."/> Please confirm Administrator Password: <input type="password" value="..... "/>

Incidentally, if you chose the other radio button option here, instead of having the reference image automatically build for you, you'll be prompted for the Administrator Password each run. This isn't really desirable in a reference scenario, as we want this to ultimately and ideally be a repeatable and reliable process that can be done hands-off.

## Summarising our entries

Finally, we get a summary screen of our entries as shown in the following image:

TaskSequenceID:	1
TaskSequenceName:	Reference Build for Windows 7 SP1 x64
TaskSequenceComments:	Version 1
Template:	Client.xml
OperatingSystem:	Windows 7 ENTERPRISE in Windows 7 x64 install.wim
FullName:	Windows User
OrgName:	Jeffs Build o Rama
HomePage:	www.bing.com
AdminPassword:	*****


## Finalising the task sequence

We will click the **Next** button where we see our task sequence is created in a confirmation page. If we chose to do so, we could view the script used to create the task sequence for later use in automation, or save the output as well so that we can audit for a change management tool or for documentation on what was done for change control or the like.

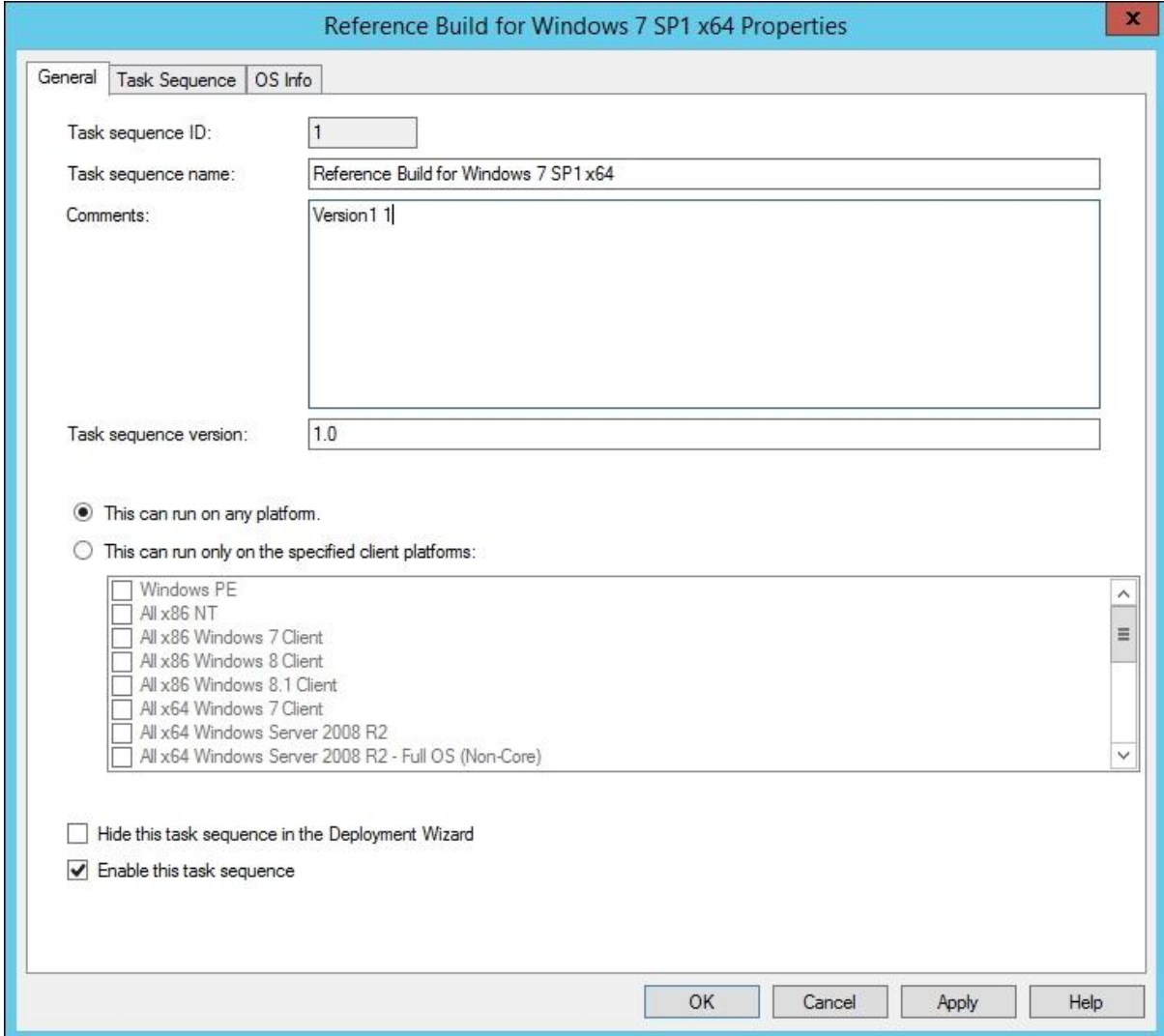
### TIP

To review the script or save output, simply click on the appropriate button at the end of our wizard screen and save the output; by default, it will open in Notepad.

After completing the wizard by selecting **Finish**, there should be a task sequence populated, as shown in the following image:

Name	ID	Version	TaskSequenceTemplate	enable	guid
 Reference Build for Windows 7 SP1 x64	1	1.0	Client.xml	True	{9838bfe8-0e29-42fc-87dc-05188247e399}

Right-click and select **Properties** on the task sequence, the following screen shows up and let's walk through the basic concepts:



The dialog box is titled "Reference Build for Windows 7 SP1 x64 Properties" and has three tabs: General, Task Sequence, and OS Info. The Task Sequence tab is selected.

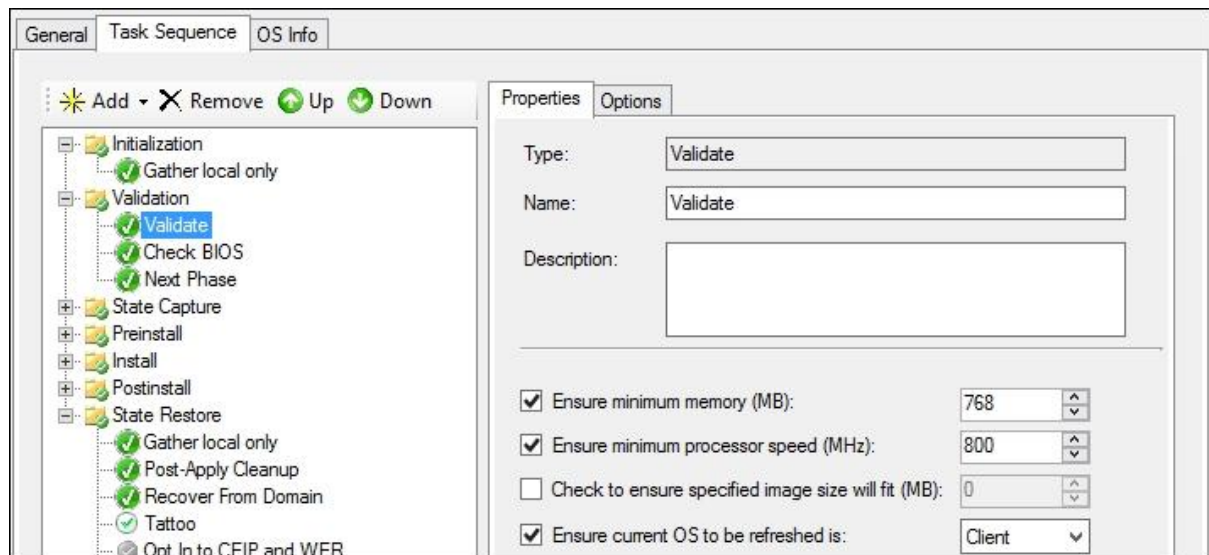
Fields and options in the Task Sequence tab:

- Task sequence ID: 1
- Task sequence name: Reference Build for Windows 7 SP1 x64
- Comments: Version 1 1|
- Task sequence version: 1.0
- Platform selection:
  - ☒ This can run on any platform.
  - ☐ This can run only on the specified client platforms:
    - ☐ Windows PE
    - ☐ All x86 NT
    - ☐ All x86 Windows 7 Client
    - ☐ All x86 Windows 8 Client
    - ☐ All x86 Windows 8.1 Client
    - ☐ All x64 Windows 7 Client
    - ☐ All x64 Windows Server 2008 R2
    - ☐ All x64 Windows Server 2008 R2 - Full OS (Non-Core)
- ☐ Hide this task sequence in the Deployment Wizard
- ☒ Enable this task sequence

Buttons at the bottom: OK, Cancel, Apply, Help.

In most deployments, it isn't necessary to select a client platform. However, if you needed to, the flexibility is there to enable the task sequence to only be runnable on certain OSes. If the sequence is a work in progress, it may make sense to hide the sequence in the deployment wizard, or disable it entirely.

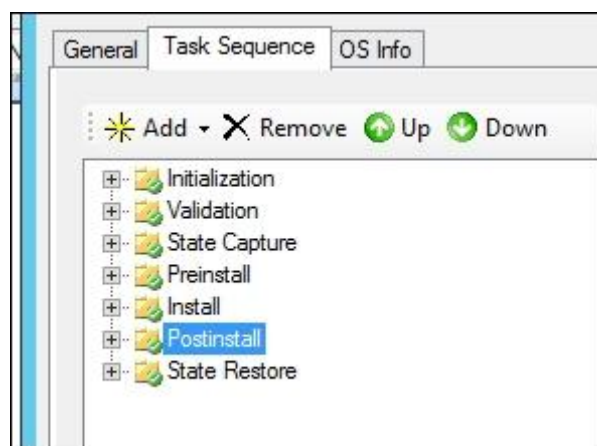
However, a common *gotcha* in virtualized reference builds is to have a virtual machine for the reference task sequence that does not have enough memory for Windows PE. You can see from the task sequence that in the **Validation** folder, the **Validate** step looks for a minimum of 768 MB of RAM available to run the task sequence. We also require an 800 MHz processor, and the OS that we are installing is Windows **Client** rather than **Server**:



So one must make sure that the virtual machine (or physical host, if you have to go that route) has more than these requirements. You *could* reduce the requirements; however, doing so would likely cause Windows PE to not load, crashing the rest of the task sequence and wasting your time in troubleshooting.

## Observing the task sequence

As we observe on the **Task Sequence** tab, the engine breaks the sequence up into seven steps. A brief introduction of them would be as follows:



- **Initialization:** In this step, initial variable information is gathered for later consumption.
- **Validation:** In this step, some of the variables are compared against modifiable rules to validate the task sequence can continue.
- **State Capture:** In this step, the state of the machine can be captured. All the task sequences can be run from either inside an OS, or from a WinPE ISO. **User State Migration Tool (USMT)**-driven state capture is done here.

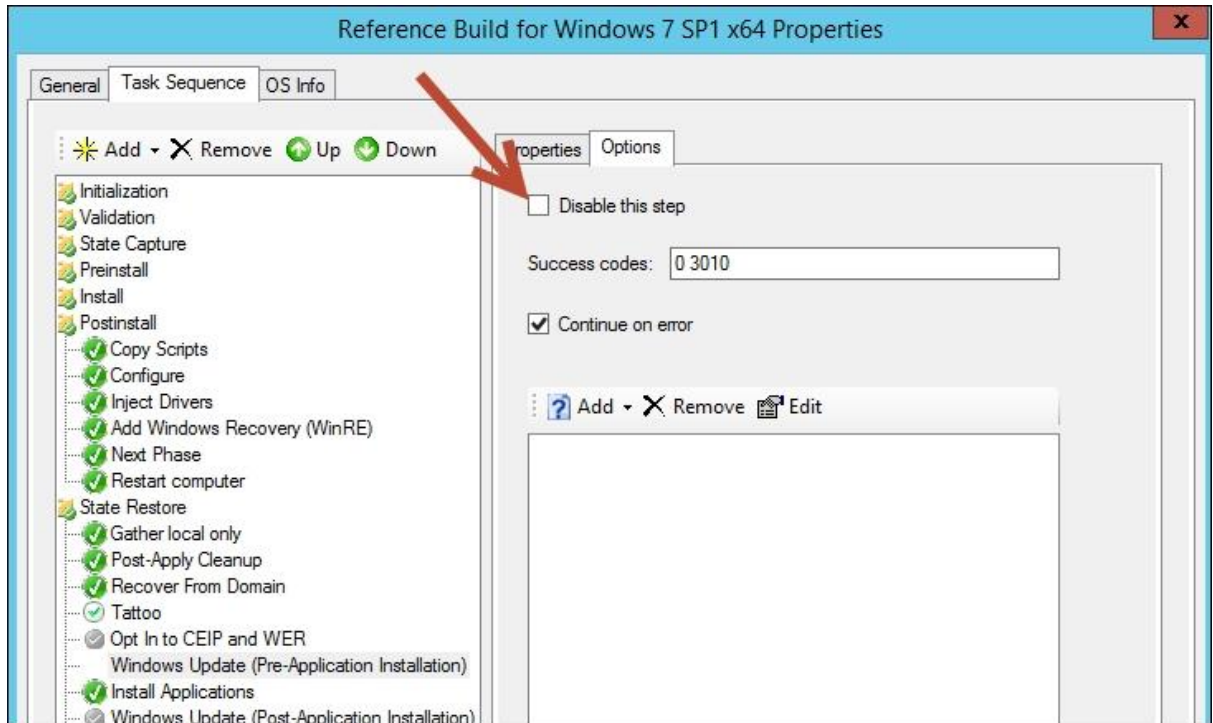


- **Preinstall:** This step is broken into multiple areas, based on the stage or type of sequence being run: **New Computer only**, **Offline User State Capture**, and **Refresh only**. This is where the driver injection happens, which can be controlled via selection profiles, which we will cover later.
- **Install:** In this step, the sequence will install the OS specified, laying down the WIM.
- **Postinstall:** In this step, drivers are injected into the Windows installation. Windows Recovery is added to the disk, and finally a reboot takes place.
- **State Restore:** The last step, if any USMT work was done to capture user data, it is restored here. In addition, join domain, installation of applications, enabling BitLocker, and Local GPO packages are applied.

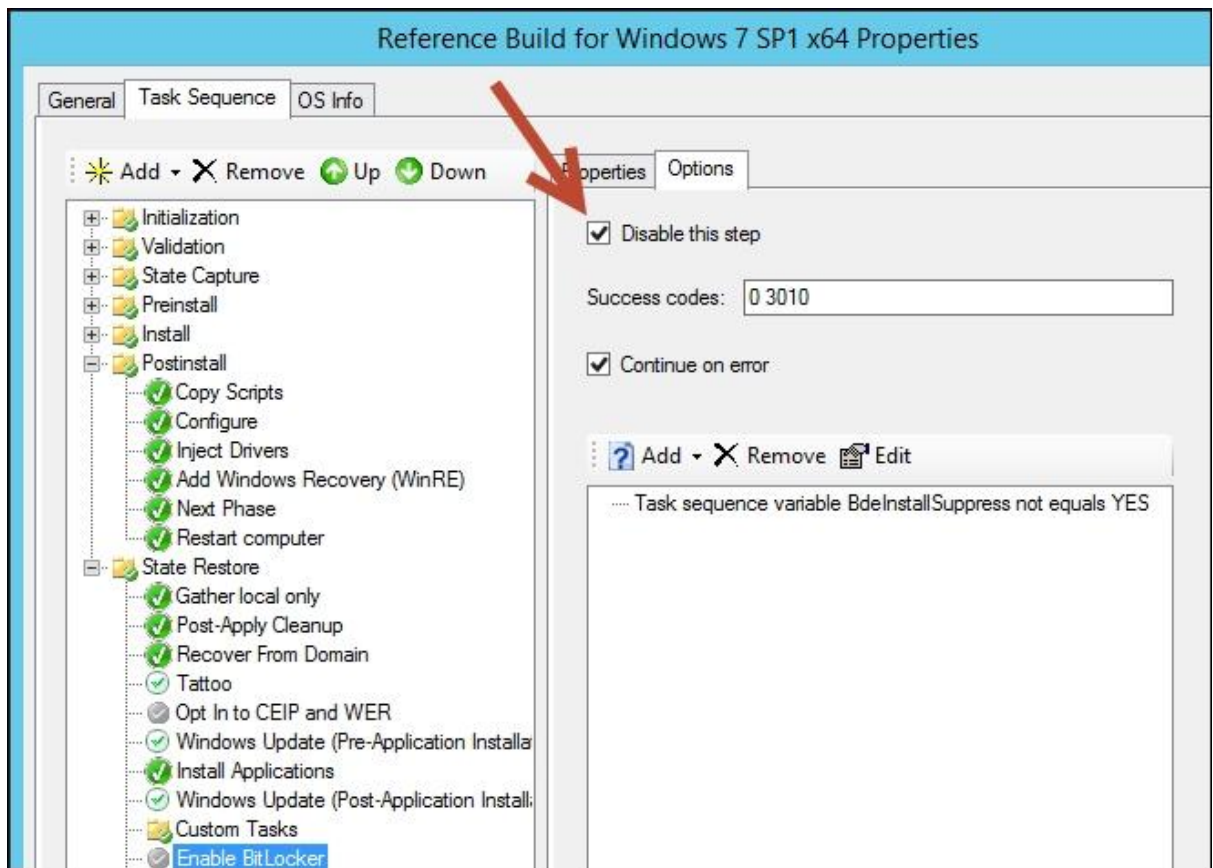
## Making configuration changes

Many options are the options with regards to what configuration changes you can make. It all depends on what your needs for your organization are. The following task sequence configuration will suit most reference image capture scenarios:

1. Enable the **Windows Update (Pre-Application Installation)** and **Windows Update (Post-Application Installation)** steps in **State Restore**. To do this, select each item to enable and click on the **Options** tab and uncheck the **Disable this step** checkbox:



2. Disable the **Enable BitLocker** step by the same method:

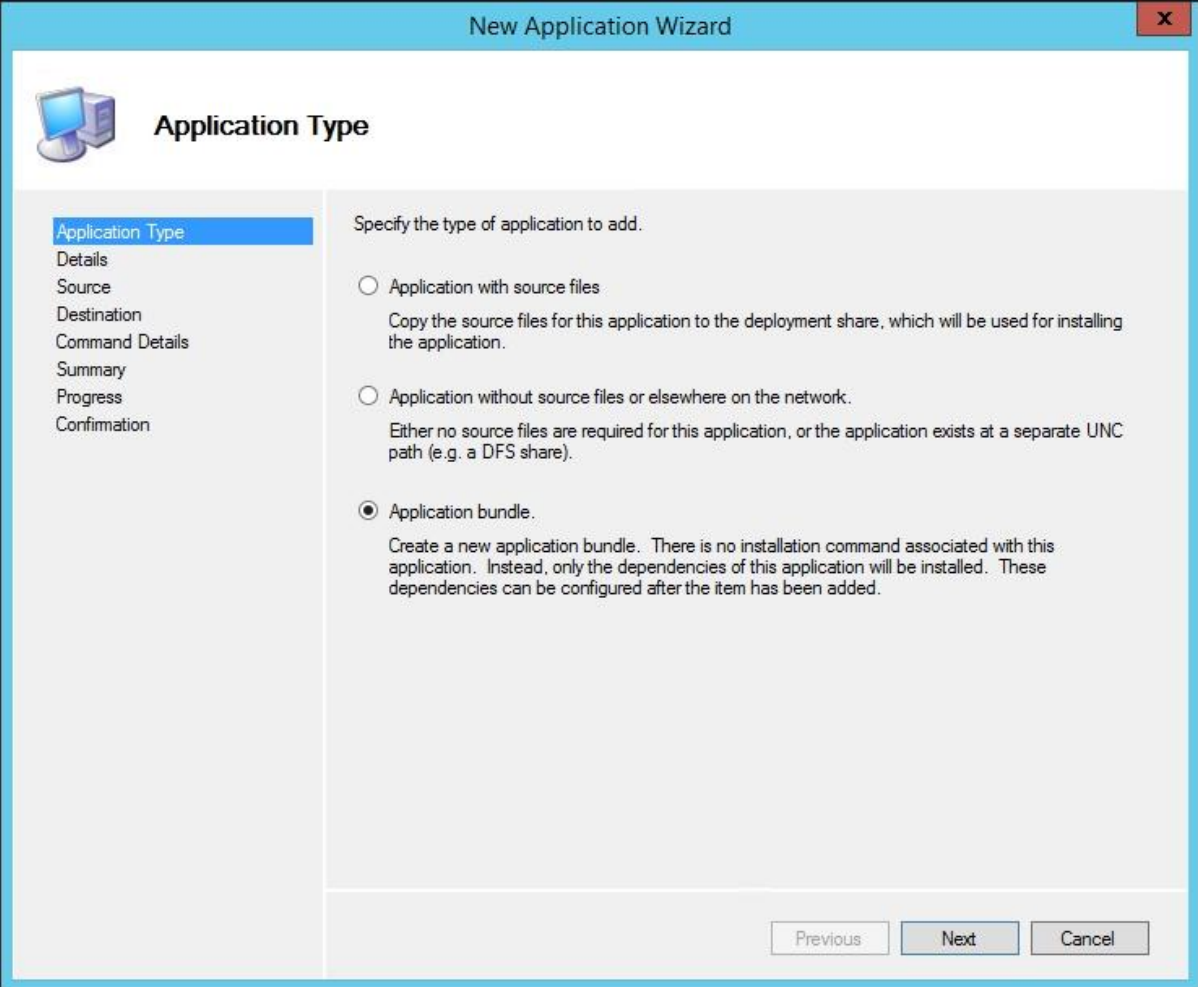


## Creating an application bundle

After making these housekeeping item changes, we need to consider what applications we want to place in our default image. What applications do we want in the image for all users, both from a cost and licensing perspective, as well as a workload perspective?

In our examples in the previous chapter, we created the applications for Foxit Reader, Java, and 7-Zip. We could individually make these mandatory applications of the image, but let's do something that is a little easier to manage: make the task sequence install an application bundle containing all three:





The image shows a Windows-style dialog box titled "New Application Wizard". It has a blue header bar with the title and a close button (X). Below the header, there is a sidebar on the left with a tree view containing "Application Type" (selected), "Details", "Source", "Destination", "Command Details", "Summary", "Progress", and "Confirmation". The main area is titled "Application Type" and contains the instruction "Specify the type of application to add." There are three radio button options: "Application with source files" (unselected), "Application without source files or elsewhere on the network" (unselected), and "Application bundle" (selected). Each option has a descriptive text block below it. At the bottom right, there are three buttons: "Previous", "Next", and "Cancel".

**New Application Wizard**

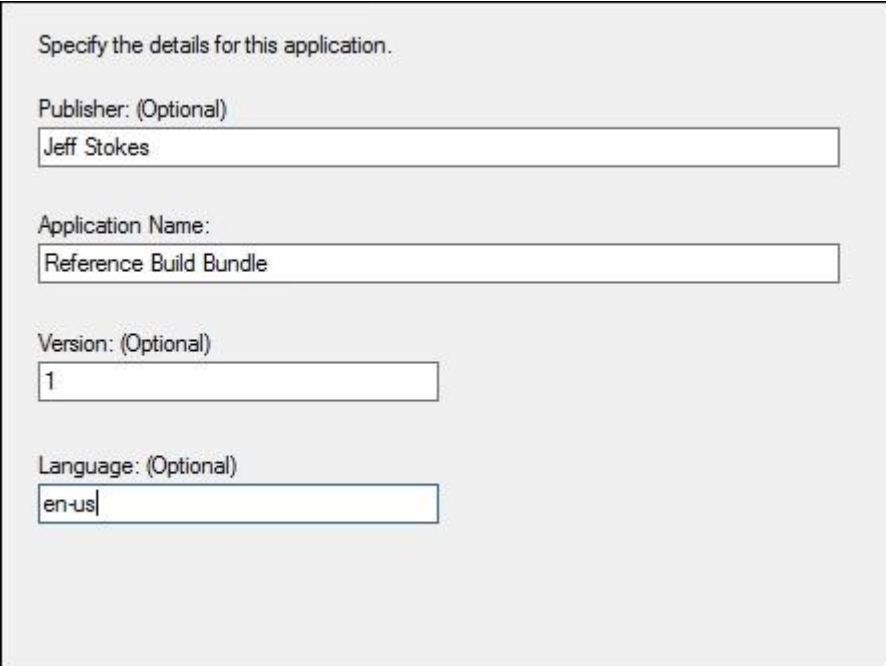
**Application Type**

Specify the type of application to add.

- ☐ Application with source files  
Copy the source files for this application to the deployment share, which will be used for installing the application.
- ☐ Application without source files or elsewhere on the network.  
Either no source files are required for this application, or the application exists at a separate UNC path (e.g. a DFS share).
- ☒ Application bundle.  
Create a new application bundle. There is no installation command associated with this application. Instead, only the dependencies of this application will be installed. These dependencies can be configured after the item has been added.

Previous Next Cancel

So, here we are creating an application bundle with more fluid details, as shown in the following image:



The image shows a form titled "Specify the details for this application." It contains four text input fields with labels: "Publisher: (Optional)" with the value "Jeff Stokes", "Application Name:" with the value "Reference Build Bundle", "Version: (Optional)" with the value "1", and "Language: (Optional)" with the value "en-us".

Specify the details for this application.

Publisher: (Optional)  
Jeff Stokes

Application Name:  
Reference Build Bundle

Version: (Optional)  
1

Language: (Optional)  
en-us

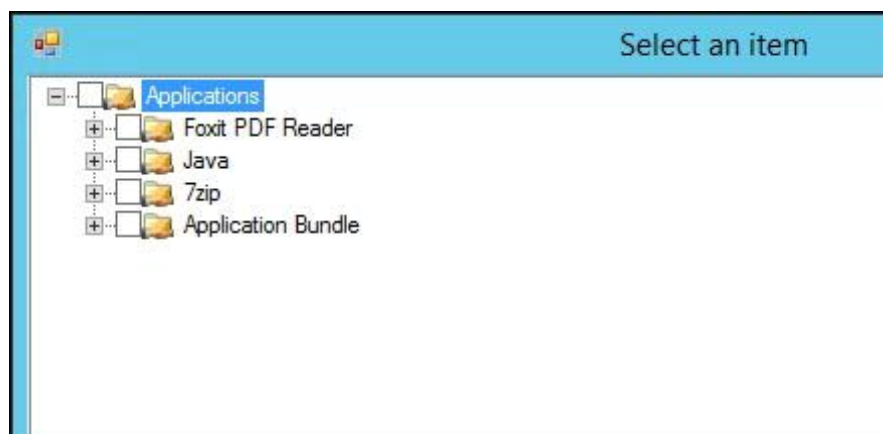
You can click on **Next** and **Finish** to complete the creation of the bundle.

## Making an application bundle object

Making an application bundle object is quite easy and straightforward. The details are in the post creation properties, in the **Dependencies** tab:



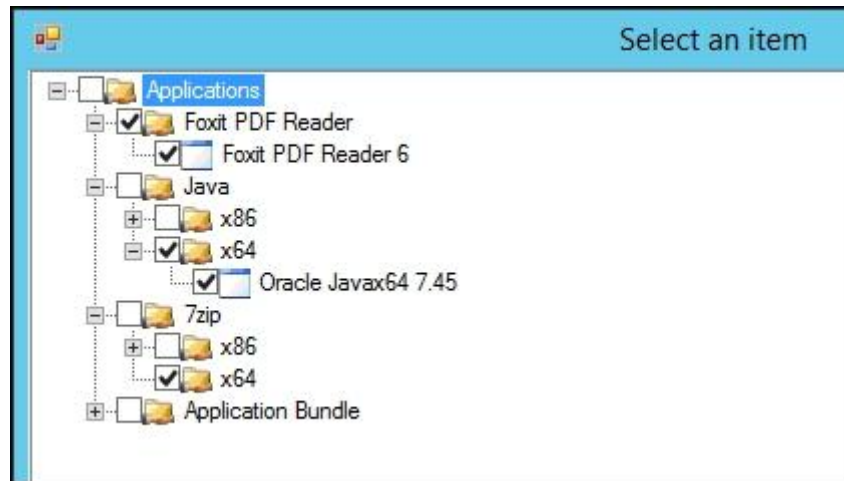
Now, it is as simple as adding each application that we want to include in our bundle:



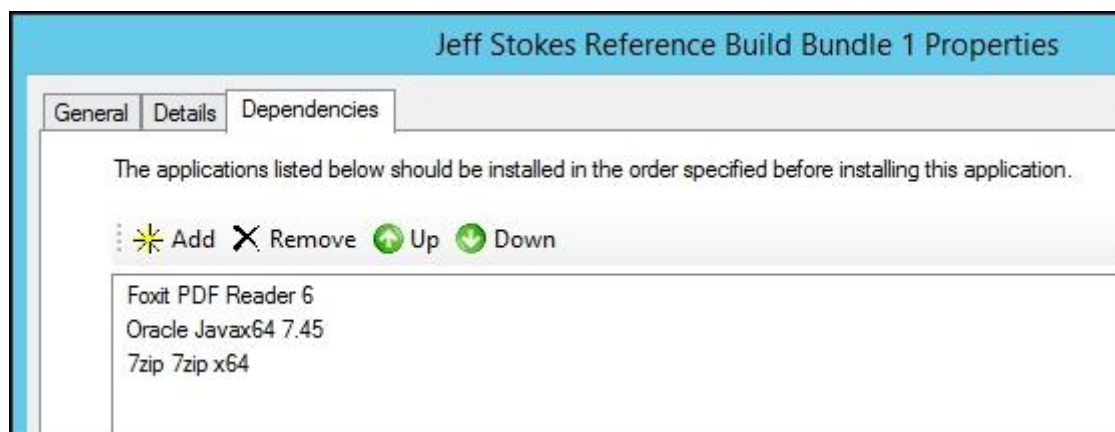
## NOTE

The order, in this particular case, doesn't really matter. However, it is important to recall that in some cases, such as when installing an application with a Java or .NET dependency, we will install these first.

So, we have selected, for this build, the x64 versions of the applications that have them, and of course, we have *not* selected **Application Bundle**, as that would be recursive:

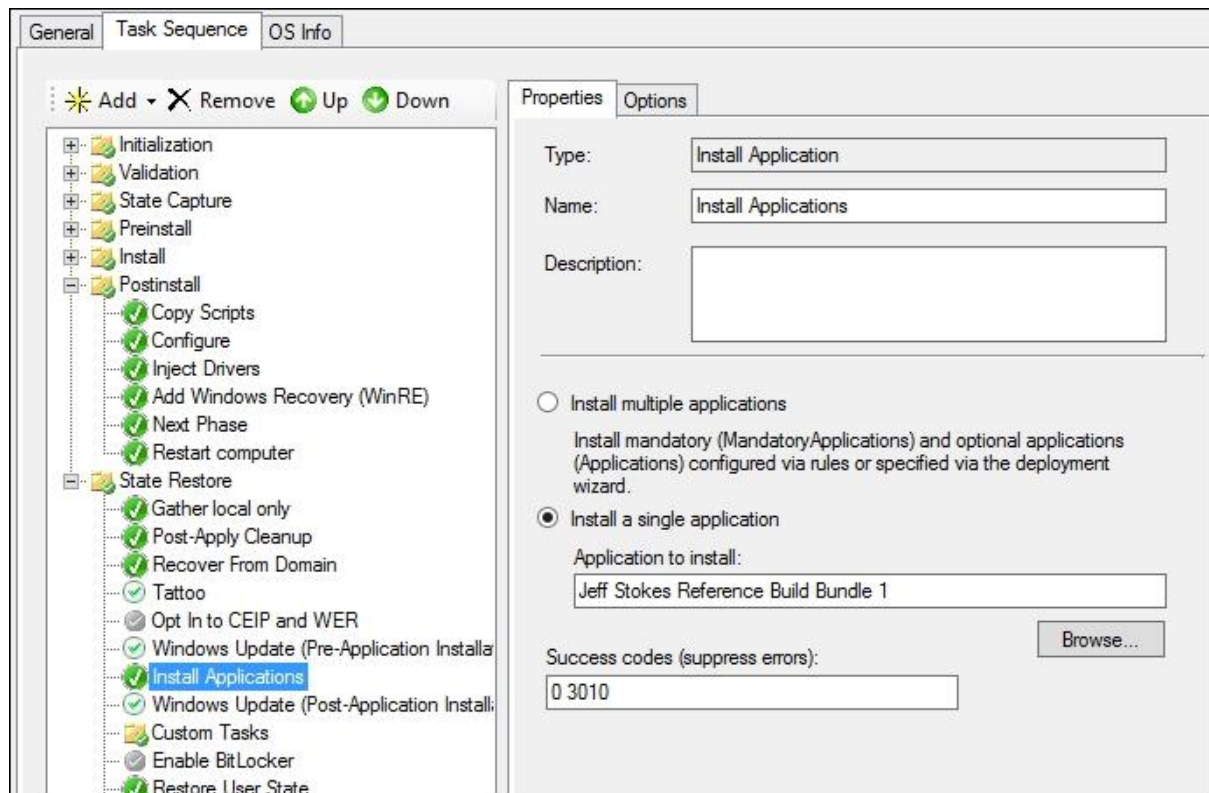


It is then as simple as applying the changes to the application bundle, as shown in the following image:



## Installing the bundle

After creating our bundle for [Task Sequence 1](#), we will go back to the properties of the task sequence and go to the **Install Applications** item, where we can force our bundle to be installed:



## Modifying the bundle

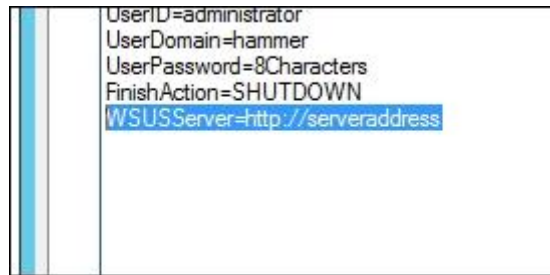
Modifying the bundle is as simple as browsing and selecting the application bundle.

If we did not select the bundle or another application individually, the default behavior is to install multiple applications, and in the task sequence, present a checkbox list for the technician or end user to select and install manually.

For a reference image though, this doesn't make sense. We want our task sequence to be repeatable, as close to a zero-touch event as possible.

## Managing updates

So now we have it, a completed task sequence to build a reference image. Additionally, in a large managed environment, we would generally build a standalone **Windows Server Update Services (WSUS)** server to manage our updates and approve, or not, the various updates available from the Windows Update Catalog site. This is simply done by adding a property to our [Rules](#) section, as shown in the following image:



## TIP

It is unsupported by Microsoft to use a WSUS instance that is SCCM-controlled. For updates that are available from the Catalog, but not available in Windows Updates (so WSUS will not display them), you can either import them directly into WSUS, or add them as a package in MDT.

### Sysprep run supportability

Sysprep is primarily a tool used to prepare a Windows installation for cloning. It has been around for some time now, so we won't go into it too much here. Suffice it to say that the tool is alive and well in Windows 7 and 8, and it is one of the tools called by MDT scripts to enable Audit mode, customize images, and build reference images.

Sysprep is famous for having a three-times limit for its run supportability on a single image before it is no longer in a supported state. This is often misunderstood, so let's get a little clarity here.

From [http://technet.microsoft.com/en-us/library/cc766514\(v=WS.10\).aspx](http://technet.microsoft.com/en-us/library/cc766514(v=WS.10).aspx) :

"There is no limit to the number of times Sysprep can run on a computer. However, the clock for Windows Product Activation begins its countdown the first time Windows starts. You can use the sysprep /generalize command to reset Windows Product Activation a maximum of three times. After the third time you run the sysprep /generalize command, the clock can no longer be reset."

The number of /generalize passes can be determined by performing slmgr /dlv (display license status verbosely) in a command prompt on running the Windows installation. It does require administrative rights.

The following table details the changes in Sysprep limitations, based on the OS:

Windows Version

Documented Sysprep Limitations

Windows XP

Reference and destination computers must have compatible HAL types.

The size of the hard disk on the destination computer must be at least the same size as the hard disk on the reference computer.

SkipRearm limit of three.

Windows 7

SkipRearm limit of three.

Windows 8

In most Windows 8 deployment scenarios, you no longer have to use the SkipRearm answer file setting to reset the Windows Product Activation clock when you run the sysprep command multiple times on a computer. In Windows 8, the SkipRearm setting is used to specify the Windows licensing state. If you specify a retail product key or volume license product key, Windows is automatically activated. You can run the sysprep command up to eight additional times on a single Windows image. After running Sysprep eight times on a Windows 8 image, you must recreate your Windows image (from <http://technet.microsoft.com/en-us/library/hh825195.aspx>).

## Boot media for the reference task sequence

Boot media is created when the reference share is right-clicked and you select **Update Deployment Share**.

This action runs a PowerShell script to update the WinPE share. This action is known as **UpdateDP**:

```
Import-Module "C:\Program Files\Microsoft Deployment  
Toolkit\bin\MicrosoftDeploymentToolkit.psdl"  
New-PSDrive -Name "DS001" -PSProvider MDTProvider -Root  
"C:\ReferenceShare"  
update-MDTDeploymentShare -path "DS001:" -Verbose
```

This script updates the boot directory with a WIM and ISO media for easy use in a VM-boot scenario or the creation of USB stick media.

### NOTE

#### Windows 10

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



- 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 applications to an inplace upgrade Image.
- For a normal *wipe and reload* Windows 10 image, you can add applications as shown previously.
- WSUS 4.0 should be patched with KB3095113.
- Changes to the SKU via the Provisioning Package or another mechanism cause an automatic reboot to occur afterward. This reboot is currently not controllable or interruptible. This will result in a **dirty environment** message in MDT due to this reboot *outside* of MDT. Speaking of Windows 10 v1511 / MDT 2013 U2, don't use this feature currently.