Tell us about your PDF experience.

Microsoft Deployment Toolkit documentation

Microsoft Deployment Toolkit (MDT) provides a unified collection of tools, processes, and guidance for automating desktop and server deployments.

Get MDT

DOWNLOAD

Download MDT

Release notes

Known issues

FAQ

HOW-TO GUIDE

How to use MDT

Samples guide

Scenario guides

QUICKSTART

Lite-touch (LTI)

Zero-touch (ZTI)

User-driven (UDI)

MDT References

REFERENCE

Toolkit reference

Troubleshoot MDT

Task Sequence Steps

Properties

Scripts

Support Files

Utilities

MDT Windows PowerShell Cmdlets

Tables and Views in the MDT DB

Windows 7 Feature Dependency Reference

UDI References

REFERENCE

UDI developer's guide

UDI Reference

Microsoft Deployment Toolkit release notes

This article provides details on the latest release of the Microsoft Deployment Toolkit (MDT). These details include supported platforms, prerequisites, and any limitations. It assumes familiarity with MDT version concepts, features, and capabilities.

⊗ Caution

Microsoft Deployment Toolkit (MDT) is retired.

> **Microsoft Deployment Toolkit (MDT) is retired.** MDT integration with Configuration Manager and MDT Standalone are **no longer supported**. Customers should **remove all MDT task sequence steps** and then **remove MDT integration** to prevent task sequence corruption and modification failures. **Consider moving to modern provisioning solutions such as Windows Autopilot**, which provides cloud-driven, zero-touch provisioning for Windows devices. Learn more about Autopilot: **here**. For customers with on-premises infrastructure and existing Configuration Manager environments, **OSD** remains a fully supported option.

> For full details on this retirement, see the **Removed and Deprecated Features** page.

Latest release

MDT build 8456 is the latest version available on the Microsoft Download Center.

This update begins support for Windows 10, version 1809, and Windows Server 2019. For more information, see the supported platforms section.

Significant changes

Here is a summary of the significant changes in MDT build 8456.

Supported configuration updates

- Windows ADK for Windows 10, version 1809
- Windows 10, version 1809
- Configuration Manager, version 1810

Major changes

The following list is a summary of the major changes in this version:

- Nested task sequence support for LTI scenario
- Modern language pack support Known issue
- Support for Configuration Manager version 1810
- IsVM evaluates to False on Parallels VMs
- IsVM = False when VMware VM is configured with EFI boot firmware
- Gather doesn't recognize All-in-One chassis type
- MDT doesn't automatically install BitLocker on Windows Server 2016
- BDEDisablePreProvisioning typo in ZTIGather.xml

Supported platforms

MDT releases are no longer tagged with year or update version. To align better with the current branches of Windows 10 and Configuration Manager, and to simplify the branding and release process, it's now simply **Microsoft Deployment Toolkit**. The build number is used to distinguish each release. For example, the latest build available for download is 8456.

Unlike Configuration Manager with a predetermined release schedule, MDT only releases as required to support new versions of Windows, the Windows ADK, or Configuration Manager current branch. Any known issues with these components will be documented in this article as necessary.

The following OS versions are supported for deployment with this build of MDT:

- Windows 10, version 1809
- Windows 10, version 1803
- Windows 10, version 1709
- Other supported versions of Windows 10
- Windows Server 2019
- Windows Server 2016

> ⓘ **Note**
>
> MDT doesn't support Windows 10 ARM64 devices or any Windows versions released after those listed above.

FAQ: Is this release only supported with Windows 10, Windows ADK, or Configuration Manager version *X*?

Prerequisites

MDT requires the following components, which are included in Windows:

- Microsoft .NET Framework 4.0
- Windows PowerShell version 3.0

MDT requires the latest Windows ADK for Windows 10. MDT also requires the **Windows PE add-on** for the Windows ADK.

> **Note**
>
> Windows recommends using the Windows ADK that matches the version of Windows you're deploying. For example, use the Windows ADK for Windows 10 version 1809 when deploying Windows 10 version 1809. For more information on Windows ADK component supportability, see DISM supported platforms and USMT requirements.

When integrating MDT with Configuration Manager for ZTI and UDI scenarios, use the latest version of Configuration Manager current branch.

Upgrade MDT

The MDT installation process removes any existing instances of MDT installed on the same computer. Existing deployment shares, distribution points, and databases are preserved during this process. They must be upgraded when the installation is complete.

The current release of MDT supports upgrading from the following versions of MDT:

- MDT build 8450

> **Tip**
>
> Create a backup of the existing MDT infrastructure before attempting an upgrade.

LTI

After installing MDT, upgrade an existing deployment share by running the **Open Deployment Share Wizard** from the Deployment Shares node in the Deployment Workbench. Specify the path to the existing deployment share directory, and then select the **Upgrade** check box. This process also upgrades existing network deployment shares and media deployment shares, so

those shares should be accessible. Don't upgrade with active deployments, because in-use files can cause upgrade problems.

ZTI

Existing MDT task sequences present in Configuration Manager aren't modified during the installation process of MDT. They should continue to work without any issue. No mechanism is provided to upgrade these task sequences. If you want to use any of the new MDT capabilities, create new MDT-integrated task sequences in Configuration Manager.

When the upgrade process is complete:

- Run the **Configure ConfigMgr Integration Wizard** after the upgrade. It registers the new components and installs the updated ZTI task sequence templates.
- Create a new **Microsoft Deployment Toolkit Files** package for any new ZTI task sequences you create. You can use the existing MDT Files package for any ZTI task sequences created before the upgrade. Create a new MDT Files package for new ZTI task sequences.

Next steps

Known issues

Frequently asked questions

Last updated on 01/06/2026

Microsoft Deployment Toolkit known issues

Article • 01/18/2023

This article provides details of any current known issues and limitations with the Microsoft Deployment Toolkit (MDT). It assumes familiarity with MDT version concepts, features, and capabilities.

> **Important**
>
> MDT is not supported with Windows 11. Any listed known issues for Windows 11 or the ADK for Windows 11 is for informational purposes only and does not imply support. For additional information, please see Supported platforms

The Create Boot Image using MDT wizard fails when creating a boot image in Microsoft Configuration Manager after upgrading to ADK for Windows 11, version 22H2

After upgrading to the ADK for Windows 11, version 22H2, the **Create Boot Image using MDT** wizard fails when trying to create a boot image with the following error:

Could not find a part of the path 'C:\Program Files (x86)\Windows Kits\10\Assessment and Deployment Kit\Windows Preinstallation Environment\x86\WinPE_OCs'.

This error occurs regardless if the boot image being created is x64.

This error is an expected error since starting with the ADK for Windows 11, version 22H2, the 32-bit versions of Windows PE are no longer included. Additionally, MDT isn't supported with Windows 11 or the ADK for Windows 11. For more information, see Download and install the Windows ADK.

The **Create Boot Image using MDT** wizard was created when Configuration Manager had no out of box functionality to create boot images using the currently installed ADK directly in the console. Integrating MDT with Configuration Manager added the functionality to create boot images using the currently installed ADK in the console. However, Configuration Manager has since added the ability to create boot images in console out of the box without the need for MDT integration.

Additionally, boot images created using the **Create Boot Image using MDT** wizard aren't required for task sequences created using the **Create MDT Task Sequence** wizard. These boot images and task sequences are commonly called "MDT" boot images and "MDT" task sequences. However, there's nothing special about an "MDT" boot image and they're binary equivalent to a Configuration Manager boot image. "MDT" task sequences are also not special. They're just Configuration Manager task sequences that happen to run MDT scripts. In other words, "MDT" boot images are not required for "MDT" task sequences to work. There are no additional binaries in an "MDT" boot image that are required by "MDT" task sequences. Configuration Manager boot images will work without issue with "MDT" task sequences.

Instead of using the **Create Boot Image using MDT** wizard to create boot images in Configuration Manager, use the out of box functionality in Configuration Manager to create boot images. For more information, see Managing boot images with Configuration Manager: Update distribution points with the boot image.

To create a new boot image using the out of box Configuration Manager functionality:

1. Navigate to the path that hosts the default x64 boot image on the Configuration Manager site server. This path would normally be `<Configuration_Manager_install_directory>\OSD\boot\x64`.

2. In the `\OSD\boot\x64` directory, make a copy of `boot.wim` and rename it to the name of your choice.

3. In the Configuration Manager console, go to the **Software Library** node and then navigate to **Overview** > **Operating Systems** > **Boot Images**.

4. Right-click on **Boot Images** and select **Add Boot Image**.

5. Follow the **Add Boot Image Wizard** to import the copy of `boot.wim` created in Step 2 as a new boot image. For more information on adding a boot image using the **Add Boot Image Wizard**, see Manage boot images with Configuration Manager: Add a boot image.

6. Once the **Add Boot Image Wizard** completes and the new boot image has been added, right-click on the newly created boot image and select **Update Distribution Points**.

   > **Note**
   >
   > Don't first distribute the newly created boot image to distribution points before selecting **Update Distribution Points**. The next steps will finish

updating the newly created boot image so that it is the same version as the currently installed ADK and Windows PE. This can be accomplished without first distributing the newly created boot image to the distribution points. It is better to finish properly creating the boot image before distributing the boot image to distribution points to avoid unnecessary updates to the distribution points. Distributing the boot image first before finishing updating the boot image will result in unneeded updates to the distribution points.

7. In the **Update Distribution Points Wizard**, select the option **Reload this boot image with the current Windows PE version from the Windows ADK**, select **Next >**, and then **Next >** again.

8. Allow the **Update Distribution Points Wizard** to complete.

Once the **Update Distribution Points Wizard** completes, the newly created boot image will be at the same version as the currently installed ADK and Windows PE.

If additional components need to be added to the boot image:

1. Right-click on the newly created boot image and select **Properties**.

2. In the boot image properties window, select the **Optional Components** tab

3. Add in the desired optional components. For more information, see Manage boot images with Configuration Manager: Optional components.

Make any additional desired changes to the newly created boot image, such as adding drivers, and then distribute the boot image to distribution points.

> ⓘ **Note**
>
> The above guide only shows x64 boot images since only x64 boot images are supported with the ADK for Windows 11, version 22H2 or newer.

HTA applications report Script error after upgrading to ADK for Windows 11, version 22H2

After you updated your MDT boot image to ADK for Windows 11, version 22H2, HTA applications stop working and a message box is displayed:

Script Error - An error has occurred in the script on this page.

HTA applications rely on MSHTML and starting with Windows 11, version 22H2, the default legacy scripting engine was changed.

To work around this issue, you need to add the following registry value in WinPE:

Windows Command Prompt

```
 reg.exe add "HKLM\Software\Microsoft\Internet Explorer\Main" /t REG_DWORD
/v JscriptReplacement /d 0 /f
```

To enable this change in MDT, we recommend that you back up the following file: `C:\Program Files\Microsoft Deployment Toolkit\Templates\Unattend_PE_x64.xml` and to modify it as follows:

XML

```
<unattend xmlns="urn:schemas-microsoft-com:unattend">
    <settings pass="windowsPE">
        <component name="Microsoft-Windows-Setup"
processorArchitecture="amd64" publicKeyToken="31bf3856ad364e35"
language="neutral" versionScope="nonSxS"
xmlns:wcm="http://schemas.microsoft.com/WMIConfig/2002/State">
            <Display>
                <ColorDepth>32</ColorDepth>
                <HorizontalResolution>1024</HorizontalResolution>
                <RefreshRate>60</RefreshRate>
                <VerticalResolution>768</VerticalResolution>
            </Display>
            <RunSynchronous>
                <RunSynchronousCommand wcm:action="add">
                    <Description>Fix HTA scripts error Windows 11 ADK
22H2</Description>
                    <Order>1</Order>
                    <Path>reg.exe add "HKLM\Software\Microsoft\Internet
Explorer\Main" /t REG_DWORD /v JscriptReplacement /d 0 /f</Path>
                </RunSynchronousCommand>
                <RunSynchronousCommand wcm:action="add">
                    <Description>Lite Touch PE</Description>
                    <Order>2</Order>
                    <Path>wscript.exe X:\Deploy\Scripts\LiteTouch.wsf</Path>
                </RunSynchronousCommand>
            </RunSynchronous>
        </component>
    </settings>
</unattend>
```

After saving the changes, you'll need to completely regenerate the boot images.

Windows Deployment Services (WDS) multicast stops working after upgrading to ADK for Windows 11

After you updated your MDT boot image to ADK for Windows 11, you might see popups in Windows PE (WinPE) multicast enabled environments prompting wdscommonlib.dll and imagelib.dll are missing in WinPE.

The right way to add WDS multicast to WinPE is to install WinPE-WDS-Tools OC (WinPE optional components) into WinPE.

Follow this example to install WinPE-WDS-Tools OC in WinPE (assuming the mount folder E:\mnt exists).

Windows Command Prompt

```
Dism /mount-wim
/WimFile:"E:\DeploymentShare\Boot\LiteTouchPE_multicast_x64.wim" /Index:1
/MountDir:E:\mnt
Dism /Image:"E:\mnt" /Add-Package /PackagePath:"C:\Program Files
(x86)\Windows Kits\10\Assessment and Deployment Kit\Windows Preinstallation
Environment\amd64\WinPE_OCs\WinPE-WDS-Tools.cab"
Dism /Image:"E:\mnt" /Add-Package /PackagePath:"C:\Program Files
(x86)\Windows Kits\10\Assessment and Deployment Kit\Windows Preinstallation
Environment\amd64\WinPE_OCs\en-us\WinPE-WDS-Tools_en-us.cab"
Dism /Unmount-Wim /MountDir:E:\mnt /Commit
```

Add or replace the multicast enabled boot image in WDS snap-in for Microsoft Management Console (MMC).

ZTI extensions with version 2013 or 2107

If you install a new Configuration Manager site with version 2103 or 2107, when you run the MDT **Configure ConfigMgr Integration Wizard**, the MDT extensions aren't added to the site.

To work around this issue, disable the hierarchy setting for approved console extensions. For more information, see Enable or disable hierarchy approved console extensions.

Windows 10, version 2004

When you use MDT build 8456 with the Windows ADK for Windows 10, version 2004, the BIOS firmware type is incorrectly identified as UEFI. This issue results in failures when

refreshing an existing computer with a new version of Windows. To mitigate this issue, install the MDT hotfix 4564442.

Modern language pack support

Starting with Windows 10 version 1809, language interface packs (LIPs) are delivered as local experience packs (LXPs). LXPs are AppX bundles. When specified in the unattend.xml file, they aren't automatically selected, and the deployment fails. Don't set LXPs as default. Users should select an applied LXP from Windows settings.

Security risk when run over the network

Binaries or scripts that run over the network aren't verified against a digital signature. This issue increases the risk of an attacker tampering with the binaries and injecting malicious code.

To mitigate this issue, protect the network connection with IPsec or SMB signing.

Next steps

Release notes

Frequently asked questions

Feedback

Was this page helpful? Yes No

Provide product feedback

Microsoft Deployment Toolkit frequently asked questions (FAQ)

This article provides frequently asked questions (FAQ) about the Microsoft Deployment Toolkit (MDT). It assumes familiarity with MDT version concepts, features, and capabilities.

What's the MDT support life cycle?

For more information, see Microsoft Deployment Toolkit support life cycle.

Is this release only supported with version 'X' of Windows client, Windows ADK, or Configuration Manager?

We primarily tested this build of MDT with the configuration listed in the supported platforms list. Unless there are any explicit known issues, anything outside of the above configuration has a high probability of still working. Your mileage may vary as we haven't explicitly tested other combinations.

> **Note**
>
> This answer also applies to Windows 11.

How do I get help with MDT?

Use one of the following methods to get help with MDT (in prioritized order):

1. Post to the MDT forum. MVPs and others in the community watch and respond to posts there. This method is probably the most efficient way to get help.
2. Contact Microsoft Support. Open a support case and get some professional help.
3. If you can consistently reproduce an issue and think it's a product bug, file it in the Windows 10 Feedback Hub. The product team investigates everything that's reported. When filing feedback, use the **Enterprise Management** category and **OS Deployment** subcategory. This categorization helps classify and route your feedback to the MDT team.

Next steps

Release notes

Known issues

Using the Microsoft Deployment Toolkit

Article • 10/26/2022

Microsoft® Deployment Toolkit (MDT) 2013 allows you to automate the deployment of computers in your organization. This document provides guidance on how to plan, build, and deploy Windows® operating systems and applications using MDT 2013.

> **Note**
>
> In this document, *Windows* applies to the Windows 8.1, Windows 8, Windows 7, Windows Server® 2012 R2, Windows Server 2012, and Windows Server 2008 R2 unless otherwise noted. MDT does not support ARM processor-based versions of Windows. Similarly, *MDT* refers to MDT 2013 unless otherwise stated.

MDT performs deployments by using the Lite Touch Installation (LTI), Zero Touch Installation (ZTI), and User-Driven Installation (UDI) deployment methods. Only MDT is used in LTI deployments, while ZTI and UDI deployments are performed using MDT with Microsoft System Center 2012 R2 Configuration Manager.

This document covers these deployment methods and shows how to customize the deployment process for an organization. In addition to this document, Table 1 lists other documents that will help you perform deployments using MDT in your organization.

Table 1. Additional MDT Documents for Use in the Deployment Process

Expand table

Document	Description
Microsoft Deployment Toolkit Samples Guide	Provides scenario-based samples of how to solve common deployment issues using MDT. Most samples include example configuration files and step-by-step processes for implementing the solution. Use this document to help solve difficult deployment problems.
Toolkit Reference	Provides information about the task sequence steps, properties, support files, utilities, and databases used in MDT deployments. Use this document to help customize MDT deployments for your organization.
Troubleshooting Reference	Provides information to assist in troubleshooting common problems with MDT deployments, including lists of known issues, reviewing logs, and identifying issues by error code.

This document is organized to help you through the planning, building, and deploying processes in your organization. The following sections in this guide help you perform LTI, ZTI, and UDI deployments and customize the deployment process. Table 2 lists each section in this guide, with a brief overview of each.

Table 2. Sections in This Document

Expand table

Section	Overview
Planning MDT Deployments	Provides guidance for completing the planning tasks in preparation for MDT deployments. Review this section to help identify prerequisites, perform capacity planning, and perform any infrastructure remediation prior to deployment in a production environment, and learn how to install MDT in the environment.
Installing or Upgrading to MDT	Provides guidance for performing a new installation of MDT or upgrading to MDT from previous versions of MDT.
Performing LTI Deployments	Provides guidance for deploying Windows operating systems and applications using only MDT. The guidance presented in this section starts immediately after the installation of MDT and provides the steps for creating a reference computer, capturing an image of the reference computer, and then deploying the captured image to target computers in the organization.
Managing LTI Deployments in the Deployment Workbench	Provides guidance and step-by-step instructions for managing LTI deployments using the Deployment Workbench, including managing operating systems, device drivers, applications, the MDT database (MDT DB), and other deployment elements in the Deployment Workbench.
Running the Deployment Wizard	Provides guidance of how to run the Deployment Wizard to deploy operating systems, device drivers, and applications to the target computers in your organization.
Performing ZTI Deployments Using Configuration Manager	Provides guidance for deploying Windows operating systems and applications using MDT and Configuration Manager. The guidance presented in this section starts immediately after the installation of MDT and provides the steps for creating a reference computer, capturing an image of the reference computer, and then deploying the captured image to target computers in the organization using Configuration Manager.
Managing ZTI Deployments in the Configuration Manager	Provides guidance and step-by-step instructions for managing ZTI deployments using the Configuration Manager console in Configuration Manager, including managing operating systems, device drivers, applications, and other deployment elements.

Section	Overview
Console in Configuration Manager	
Performing UDI Deployments	Provides guidance for deploying Windows operating systems and applications using MDT, Configuration Manager, and the UDI Wizard. The guidance presented in this section starts immediately after the installation of MDT and provides the steps for creating a reference computer, capturing an image of the reference computer, and then deploying the captured image to target computers in the organization using the Configuration Manager console and the UDI Wizard.
Managing UDI Deployments	Provides guidance and step-by-step instructions for managing UDI deployments using the Configuration Manager console, the UDI Wizard Designer, and the UDI Wizard, including: - Managing operating systems, device drivers, applications, and other deployment elements in the Configuration Manager console. Most of the deployment process and step-by-step instructions for ZTI are applicable to UDI. - Managing the behavior of the UDI Wizard.
Running the UDI Wizard	Provides guidance of how to run the UDI Wizard to deploy operating systems, device drivers, and applications to the target computers in your organization.
Configuring MDT Deployments	Provides guidance on how to customize the process for more advanced deployment scenarios, including a discussion of the MDT configuration files, how to apply configuration settings to groups of computers, and how to apply configuration settings to individual computers.
Performing Deployments Using the MDT DB	Provides guidance on customizing the process for more advanced deployment scenarios, including how to add entries to and retrieve configuration settings from the MDT DB.
Preparing the MDT Migration Resources	Provides guidance on preparing the appropriate resources used in the MDT deployment process, including network shared folders and database access.
Preparing Windows Deployment Services	Provides guidance on preparing Windows Deployment Services for use in initiating the LTI, ZTI, and UDI deployment processes, including creating images and configuring for integration with LTI, ZTI, and UDI deployments.
Planning for Application Deployment	Provides guidance on how to deploy applications by using MDT, Configuration Manager, and Microsoft Application Virtualization (App-V), including deployment of applications with the operating system image or after the image is deployed.

Planning MDT Deployments

The planning process helps you prepare for deployments in a production environment. The process starts with conceptual designs, which are proven and refined in a test environment. The result of the planning process is a set of design documents that you can use to build an MDT deployment infrastructure and perform automated operating system and application deployments in a production environment.

Overview of the MDT Deployment Process

The purpose of MDT is to help automate the deployment of Windows operating systems and applications to desktop, portable, and server computers in the environment. At a high level, MDT automates the deployment process by configuring the unattended Setup files for Windows and packaging the necessary files into a consolidated image file that you then deploy to reference and target computers.

Figure 1 illustrates the high-level LTI, ZTI, and UDI deployment processes.

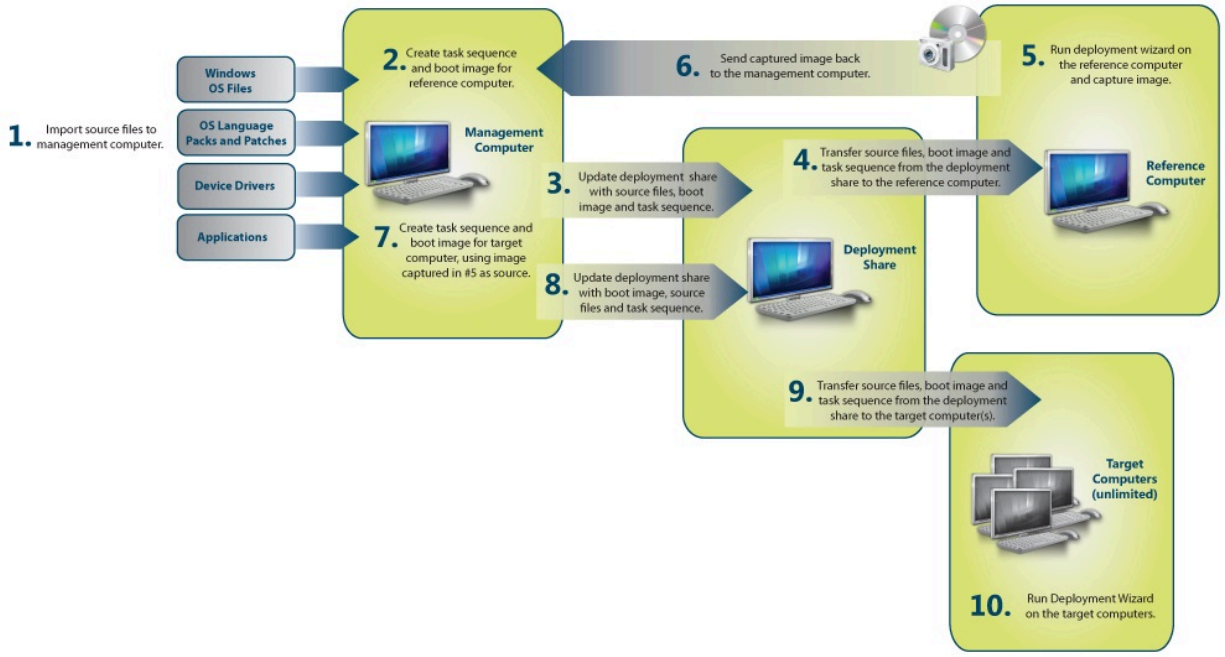


Figure 1. High-level deployment process

The high-level LTI, ZTI, and UDI deployment process is as follows:

1. Collect the files necessary to perform an MDT deployment, including:
    - Windows operating system source files or images
    - Windows operating system language packs

- Device drivers for reference and target computers

2. Create the system images, configuration settings, and task sequences to be used in deploying Windows and applications to the reference computers.

3. Deploy the system images to the reference computer and capture an image of the reference computer.

4. Create the configuration settings and task sequences that will deploy the captured images of the reference computers to the target computers.

5. Deploy the captured images of the reference computers to the target computers.

Planning Checklist

Table 3 provides a planning checklist in the form of a list of questions that you can use to help in the planning process. For each question, use the information provided in the **Overview** column to help find answers based on your organization's requirements.

Table 3. Planning Checklist

Expand table

Question	Overview
Where will you store your distribution files?	Files for the operating system and applications are stored in deployment shares for LTI and distribution points for ZTI and UDI. These files can require many gigabytes of space. Some organizations might need multiple deployment shares or distribution points for different regional offices.For more information, see Estimating MDT Storage Requirements.
Will you deploy across the network, with removable media, or both? Will you use multicast deployments?	If you are deploying across the network, verify that there is sufficient bandwidth between the deployment shares, distribution points, and the target computers, and provide regional distribution points.For more information, see Choosing a Deployment Method and Evaluating Network Requirements.
What is your imaging and source file strategy?	For more information, see Using Reference Computers in MDT Deployments.
Will you deploy a full set of operating system files or a custom image?	For more information, see Using Reference Computers in MDT Deployments.

Question	Overview
How will you handle product keys and licensing?	Small organizations might assign each user an individual product key. Larger organizations should use Key Management Service (KMS) or Multiple Activation Key (MAK) activation. For more information, see Product Activation and Key Information.
Are you going to allow users to choose their own operating system, applications, locale, time zone, and administrative password?	Users can select this information at deployment time, or you can configure the information ahead of time. For more information, see Choosing LTI, ZTI, or UDI Deployments.
Will users refresh their current computer in place, migrate settings to a new operating system installation, or get a new computer?	For more information, see Identifying Deployment Scenarios.
Which users will be able to install which applications?	For more information, see Planning for Application Deployment.
Are you going to migrate user state?	For more information, see Estimate Storage Requirements for User State Migration Data.
Do you want to back up computers before deployment?	For more information, see Estimate Storage Requirements for Target Computer Backup.
Do you want to use BitLocker® Drive Encryption?	For more information, see Planning for BitLocker Drive Encryption.
Will you deploy 32-bit and 64-bit operating systems?	For more information, see Estimating MDT Storage Requirements.
Will you deploy different product editions (such as Professional, Ultimate, or Business)?	For more information, see Estimating MDT Storage Requirements.
What type of deployments will be performed (for example, deploy a new computer, replace an existing computer)?	For more information, see Identifying Deployment Scenarios.

Estimating MDT Storage Requirements

LTI deployments store the system images, applications, and other files in deployment shares. ZTI and UDI deployments store these files on Configuration Manager distribution points. To determine your storage needs, estimate storage requirements for:

- Computers running MDT as described in Estimate Storage Requirements for Computers Running MDT
- Each MDT deployment share as described in Estimate Storage Requirements for MDT Deployment Shares
- Each Configuration Manager distribution point as described in Estimate Storage Requirements for Configuration Manager Distribution Points
- User state migration data as described in Estimate Storage Requirements for User State Migration Data
- Backing up existing computers prior to deployment for Refresh Computer deployment scenario as described in Estimate Storage Requirements for Target Computer Backup

Estimate Storage Requirements for Computers Running MDT

The computer running MDT has the following storage requirements:

- At least 4 gigabytes (GB) of free space is required on the drive containing the %TEMP% folder if you will create a media images. Otherwise, 1 GB of free space is required on the drive containing the %TEMP% folder.
- Free space of 1 GB is required on the drive containing the MDT program files.

Estimate Storage Requirements for MDT Deployment Shares

Ensure that sufficient space is available for storing the operating system images, language packs, and device drivers used in the Deployment Workbench. You store these images in the MDT deployment shares created in the Deployment Workbench.

Determine the storage requirements for each of the following items in the deployment share:

- Windows operating system
- Operating system language pack
- Device drivers

- Applications

  Determine the size of each image and the number of images required in the deployment. Create a unique image for each:

- **Version of the Windows operating system to be deployed.** A separate image is required for each version of Windows, such as Windows 8.1, Windows 7, Windows Server 2012 R2, or Windows Server 2008 R2.

- **Edition of the Windows operating system to be deployed.** A separate image is required for each edition of Windows, such as Tablet PC, Ultimate, Business, Enterprise, or Datacenter.

- **Processor type.** A separate image is required for 32-bit and 64-bit versions of Windows.

Estimate Storage Requirements for Configuration Manager Distribution Points

Estimate the storage requirements for Configuration Manager distribution points using the same calculations described in Estimate Storage Requirements for MDT Deployment Shares. If the images are distributed to multiple distribution points, the storage requirements apply to each distribution point.

For more information about planning Configuration Manager distribution points, see the section, "Distribution Point," in the section, "Planning Configuration Manager Site Systems for Operating System Deployments," in the Configuration Manager Documentation Library, which is installed with Configuration Manager.

Estimate Storage Requirements for User State Migration Data

Estimate the amount of storage required for user state migration data that the Windows User State Migration Tool (USMT) saved during the deployment process by:

- Determining whether to store the user state migration data locally on the target computers or network shared folders as described in Determine Where to Store User State Migration Data

- Determining the storage requirements for the user state migration data as described in Determine Storage Requirements for User State Migration Data

Determine Where to Store User State Migration Data

After determining the storage requirements for the user state migration data, determine where to store the data. Store user state migration data in these locations:

- On the local computer to reduce the time to deploy Windows as well as network utilization (recommended)

  > ⓘ **Note**
  >
  > This option can be used only in a Refresh Computer scenario.

- On a shared folder located on a local server to provide a consistent method of storing user state migration data or when local storage is not available.

  If user state migration data will be stored locally on the target computers, designate a shared folder in which the deploy process can store the data. By default, the process attempts to store user state data on the local hard disk for the Replace Computer and Refresh Computer scenarios. However, you can override this behavior with configuration settings in CustomSettings.ini. In the event that there is insufficient disk space for the user state data and new image, the deployment scripts attempt to store the information in a shared folder. Providing the shared folder as an alternate storage location makes the deployment process more reliable.

  Create a share on a server designated during the planning process for holding the USMT store files. MDT uses values found in CustomSettings.ini to locate the user state store folder.

Determine Storage Requirements for User State Migration Data

For planning purposes, complete the following tasks to estimate the user state migration storage requirements:

- Run Scanstate.exe in the USMT with the **/p** option to estimate the size of the user state migration data. By using the **/p** option, you can estimate the disk space requirements without actually performing the migration.

- View the size of the contents of the folders in the user profile. Randomly sample targeted computers to determine a typical amount of storage required to back up the user state migration. Keep in mind that there may be several profiles (user name folders) on each target computer, so include each profile to be migrated.

  Calculate the total capacity required by multiplying the average size of the user state migration data by the number of days to retain the data, and then

multiplying that result by the number of users to be migrated during the retention period. For example, if the average user state migration size is 3 GB, data must be stored for five days, 100 users are being migrated each day, and the total storage requirement is 1,500 GB (3 GB × 5 days × 100 users per day).

Estimate Storage Requirements for Target Computer Backup

As an optional step in the deployment process for the Refresh Computer scenario, you can perform a backup of a target computer before deploying the target operating system.

You perform the backup process in MDT by using the Imagex.exe tool. The backup process creates an image of the disk volume on which the user state migration data is stored. The purpose of this backup is for recovery of user state migration data, not to restore the target computer from the image.

The storage requirements are a function of the average size of the target computer hard disks, the number of target computers deployed each day, and the length of time you want to retain the backup. For example, if the average target computer hard disk contains 80 GB of data, you are deploying 100 computers per day, and you want to retain the data for one week, the storage requirements for backups are 56 terabytes (TB), or 80 GB × 100 × 7.

> ⓘ **Note**
>
> By default, the MDT backup process does not back up multiple partitions. If you need to back up multiple partitions, modify the MDT deployment process or use an alternative backup method.

Planning for Application Deployment

Applications can be deployed as a part of the operating system image or after the operating system is deployed to the target computer. In preparation for deployment, perform these tasks:

- **Create an application portfolio.** Application portfolios include a list of applications and the compatibility status of each application. You can create this application portfolio by using software-inventory software such as the Application Compatibility Toolkit (ACT), the Asset and Compliance feature in Configuration Manager.

- **Identify any dependencies between applications.** Applications may have dependencies on other applications. For example, an application may rely on Microsoft Office Excel® 2007. Identify these dependencies, and include the dependent software in the deployment plans.

- **Determine whether to deploy applications with the operating system image or afterwards.** You can deploy applications as part of the operating system image or after the operating system is deployed to the target computer. If the application is deployed after the operating system is deployed, you can use any software-deployment software, such as MDT, Group Policy Software Installation, the Application Management feature in Configuration Manager.

- **Determine the appropriate method for running applications.** You can install and run applications on the local computer or deploy them dynamically in a virtualized application environment, such as App-V.

- **Identify the users approved to install applications.** Determine whether users will install their applications or if the applications need to be installed by deployment technicians. Ensure that the user installing the application has the appropriate rights and permissions.

- **Identify applications that require a restart of the operating system.** Applications that require a restart of the operating system after installation require additional configuration. For more information, see Configure the Computer to Restart After Application Installation.

Defining Operating System Components and Settings

As part of establishing a standardized configuration, determine which operating system components to include and the settings for these components. This determination includes optional components in all operating systems, server roles in Windows Server operating systems, and components to include in Windows Preinstallation Environment (Windows PE). For example, you may decide to remove unnecessary Windows operating system components from desktop and portable computer deployments to reduce the security footprint of those computers.

For each operating system image, determine the:

- **Operating system components.**Select the components required for the applications and user roles performed on the target computers. Install only the components that are required to help reduce the attack surface of the target computer and the image size.

- **Server roles.** Select the server roles required for the server computers. Install only the server roles that are required to help reduce the attack surface of the target computer and the image size.
- **Windows PE components.** These components include Microsoft ActiveX® Data Objects (ADO) support, fonts, and the necessary drivers and packages. You can select the components for 32-bit and 64-bit versions of Windows PE.
- **Configuration settings.** Identify the configuration settings for components included in the images. Select configuration settings that meet the business and security requirements of the organization. For more information about target computer security, see Planning Target Computer Security.

Choosing a Deployment Method

Typically, target computers have high-speed, persistent connections to the deployment infrastructure. However, some target computers may connect to an intranet remotely or not at all. MDT includes the following methods for deploying operating systems and applications using LTI based on the network connectivity:

- **Deployment share.** This method uses a network shared folder in which all the deployment files reside. The target computer starts Windows PE, and then connects to the deployment share to perform the deployment. Select this method when the target computers have high-speed, persistent connections to the deployment infrastructure.
- **Media.** This method creates an image that you can use to perform deployments from removable media, such as DVDs or USB flash drives (UFDs). You use Windows PE to start the computer from the media. Select this method when the target computers may be remotely connected or may not have connectivity at all.

Evaluating Network Requirements

Because of the size of the images being distributed to the target computers (500 megabytes [MB] to 4 GB), computers must have a high-speed, persistent connection to the servers used in the deployment process. These servers need to be on adjacent subnets to the target computers to ensure high-speed connectivity to the computers.

> ⓘ **Note**
>
> Network-based deployments using MDT are not supported for wireless networks. Use media-based deployments for computers connected by wireless networks or

networks with slow or unreliable connectivity.

If the organization cannot provide sufficient network capacity to deploy images, software, and migration data to computers, perform one of the following actions:

- Temporarily place the appropriate servers (for example, servers hosting the various shared folders or the server running Windows Deployment Services) closer to the target computers for the duration of the migration.
- Temporarily move the target computers to a staging area where the computers can be deployed, and then return them to their original location.
- Store user state migration data locally on the target computers.
- Perform automated deployments locally using media deployments in LTI.

  In addition to network capacity, you must enable the appropriate network protocols and traffic. For example, if you want to initiate LTI, ZTI, or UDI deployment using Windows Deployment Services and multicast deployment, you must enable multicast traffic between the MDT infrastructure and target computers.

Using Reference Computers in MDT Deployments

The MDT deployment process uses the reference computer as a baseline for the configuration of target computers when the deployment process is complete. You configure the reference computer to comply with the business, technical, and security requirements of the organization. After configuring the reference computer, capture an image of the reference computer that you can then deploy to the target computers.

Only in rare circumstances will you be able to deploy the images from the Windows distribution media unmodified to the reference and target computers. Instead, create customized images that include the Windows operating system, language, packs, applications, device drivers, software updates, and other software.

The MDT deployment process allows for the creation of customized images that are first deployed to a reference computer, then captured from the reference computer, and finally deployed to target computers. MDT manages the customization of images so that you can create them with less effort and higher levels of automation. For example, the Deployment Workbench in MDT can automatically inject the appropriate device drivers into images.

VMs work well when creating a reference image for Windows because the historical HAL issues are no longer applicable.

> **Note**
>
> VMs typically do not have the same performance as physical computers, so creating the reference images may take longer.

Choosing Thick, Thin, or Hybrid Images

As part of the planning process, determine the types of images that you will create. The types of images you can create fall into these categories:

- **Thick images.** Thick images are monolithic images that contain core applications, language packs, and other files. Part of the image-development process is installing core applications and language packs on the reference computer before capturing the disk image.
- **Thin images.** Thin images contain few if any core applications or language packs, as these components are installed separately from the disk image, which typically takes more network transfer time at the computer.
- **Hybrid images.** Hybrid images mix thin and thick image strategies by installing applications and language packs from a network shared folder. Hybrid images have most of the advantages of thin images, but they are not as complex to develop and do not require a software-distribution infrastructure. They do require longer installation times, however, which can raise initial deployment costs.

Table 4 lists the advantages and disadvantages of the thick, thin, and hybrid images types.

Table 4. Advantages and Disadvantages of Thick, Thin, and Hybrid Images

Expand table

Method	Advantage	Disadvantage
Thick	- Can be simpler to deploy, because all applications and language packs are in the image.	- Requires more storage for each image. - Requires more time to download over network connections than thin or hybrid images.

Method	Advantage	Disadvantage
	- Reduced initial complexity, because advanced scripting is not typically required. - Applications and language packs are available immediately after deployment is complete. - Does not require software-distribution software, such as the Application Management feature in Configuration Manager.	- Requires an increased image maintenance effort, because any updates to operating systems, device drivers, applications, or language packs requires the creation of a new image.
Thin	- Requires less storage for each image. - Requires less time than thick images to download over network connections. -Reduced image maintenance effort, because the image contains fewer components.	- Can be more complex to createinitially, because additional steps are required during image creation. - Potential for increased complexity, because advanced scripting may be required. - Applications and languages are not immediately available after image deployment is complete.
Hybrid	- Requires less storage than thick images for each image. - Requires less time to than thick a thick image to download over network connections. - Reduced image maintenance effort, because the image contains fewer components. - Does not require separate software-distribution software.	- Can be more complex than a thick image (but not than a thin image) to create, because additional steps are required during image creation. - Potential for increased complexity, because advanced scripting (though not as advanced as in thin images) may be required. - pplications and languages are not immediately available after image deployment is complete.

The costs associated with building, maintaining, and deploying disk images includes:

- **Development costs.** Development costs include creating a well-engineered image to lower future support costs and improve security and reliability. Higher levels of automation reduce development costs.
- **Test costs.** These costs include the time and labor involved in testing the standard image and the applications that might reside inside it in addition to applications

applied after deployment. Test costs also include the development time required to stabilize disk images.

- **Storage costs.** Storage costs include storing the distribution points, disk images, migration data, and backup images. Storage costs can be significant depending on the number of disk images, the number of computers in each deployment run, and so on.

- **Network costs.** Network costs include moving disk images to distribution points and to computers. The disk-imaging technologies that Microsoft provides do not support multicasting, so network costs scale linearly with the number of distribution points you must replicate and the number of computers in the deployment project.

  As the size of image files increases, costs increase. Large images have more updating, testing, distribution, network, and storage costs associated with them. Even if only a small portion of the image is updated, the entire image must be redistributed.

Identifying Deployment Scenarios

Table 5 lists the deployment scenarios and provides a brief description of each.

Table 5. Deployment Scenarios

Expand table

Scenario	Description	Migrates user state	Uses existing target computer	Preserves file system
New Computer	A new installation of a Windows operating system is deployed to a new computer.	No	No	No
Refresh Computer	A computer is refreshed, including computers that must be re-imaged for image standardization or to address a problem.	Yes	Yes	No
Replace Computer	One computer replaces another computer. The existing user state migration data is saved from the original computer. Then, a new installation of Windows is deployed	Yes	No	No

Scenario	Description	Migrates user state	Uses existing target computer	Preserves file system
	to a new computer. Finally, the user state data is restored to the new computer.			

MDT does not support in-place upgrade deployments. You can perform:

- An in-place upgrade manually by running Setup.exe from the original Windows media

  > **Note**
  >
  > To perform an in-place upgrade manually by running Setup.exe from the original Windows media, use the original install.wim file. Custom .wim files are not supported for in-place upgrades.

- The Refresh Computer scenario as an alternative for deploying a new operating system and applications on the target computer

  As part of the Replace Computer deployment scenario, wipe the disk partitions of the original computer. The standard format as performed by Windows operating systems does not perform a secure wipe of the disk as defined by U.S. Department of Defense standard 5520.22M. If required, perform secure wipes of hard disks in target computers using tools provided by non-Microsoft vendors.

Planning for BitLocker Drive Encryption

BitLocker is included in Windows so include planning decisions for BitLocker in your environment. One BitLocker decision you must make is the storage of the recovery keys. You can store BitLocker recovery keys in:

- `A local folder.` Select this option to store the recovery key on UFDs, which each user manages.
- **A network folder.** Select this option to centrally store the recovery keys in a network shared folder, which network administrators manage.
- **Active Directory® Domain Services (AD DS).** Select this option to store the recovery keys in AD DS, which Active Directory administrators manage.

Also, elect the methods users will employ to start their computers after BitLocker is enabled. Users can start their computers using one of the following methods:

- **Trusted Platform Module (TPM) version 1.2 or later.** TPM is a cryptographic hardware chip installed on the target computer. If the target computer does not support TPM, a UFD or PIN must be used to start the computer. This is the preferred method if the target computer supports TPM.

  > ⓘ **Note**
  >
  > You can provide a PIN that users can enter in conjunction with TPM, or you can use a UFD to strengthen the security when starting a computer.

- **UFD.** In this method, the required encryption keys are stored on a UFD, which must be present in the computer when the computer starts. This is the preferred method if the target computer does not support TPM.

  For more information on BitLocker, see BitLocker Drive Encryption Overview.

Evaluating Target Computer Readiness

As part of the planning process, evaluate target computer readiness for the deployment of the target operating system, device drivers, applications, and other components. Evaluate target computer readiness using automated hardware and software inventory tools, such as Configuration Manager or the Microsoft Assessment and Planning (MAP) Toolkit.

Evaluate target computer readiness for deployment by:

- Verifying target computer readiness for running the MDT scripts as described in Verify Target Computer Readiness for Running MDT Scripts
- Verifying that target computers have adequate software and hardware system resources as described in Verify Adequate Target Computer Resources
- Identifying the differences in the deployment process between 32-bit and 64-bit computers as described in Identify Differences in 64-bit and 32-bit Deployments

Verify Target Computer Readiness for Running MDT Scripts

Before running the rest of the MDT scripts, run ZTIPrereq.vbs to ensure that the target computer meets the requirements for running the remaining MDT scripts. Script prerequisites include:

- Windows Script Host (WSH) version 5.6 or later installed and running
- Microsoft XML Core Services (MSXML) version 3.0 (any service pack level) installed and running

  > ⓘ **Note**
  >
  > The version of MSXML must be version 3.0. MSXML versions 4.0 and 6.0 are not compatible with the MDT scripts.

Verify Adequate Target Computer Resources

After ZTIPrereq.vbs determines that the computer meets the requirements for running the remaining scripts, ZTIValidate.wsf determines whether the target computer has the appropriate software and hardware system resources to deploy the target operating system. These requirements include:

- The target computer has WSH 5.6 or later installed
- In any scenario except New Computer (which does not migrate user data), the existing operating system must be a client operating system if the new operating system is a client operating system. Similarly, only a server operating system can be deployed to a computer currently running a server operating system.
- The `OSInstall` property, if defined, must be set to `YES` for the deployment to continue
- The target computer memory must meet the requirements of the operating system

  > ⓘ **Note**
  >
  > The minimum recommended amount of physical memory for the target computer is 1 GB.

- The target computer processor must meet the requirements of the operating system

- The target computer must have sufficient available disk space for the image being deployed to it
- The current operating system on the target computer must be running on the C partition (Refresh Computer scenario only)
- Drive C must be the first partition on the first disk of the target computer (Refresh Computer scenario only)
- Additional available disk space is required when user state migration data and deployment logs are stored locally on the target computer
- The target computer must have sufficient free disk space (approximately 150 MB) to hold Windows PE log files
- The target computer must have sufficient total disk space to hold Windows PE and the image (expanded image size plus 150 MB)
- The target computer must have a direct network connection to Windows Deployment Services servers and deployment shares (Unsupported network connections include virtual private network [VPN] and wireless connections.)

  > ⓘ **Note**
  >
  > Target computers that attempt to install an image over a VPN or wireless connection will not be able to connect to a deployment share after restarting in Windows PE, causing the deployment process to fail.

  Determine whether any existing computers have inadequate system resources using Configuration Manager or another software inventory tool. Upgrade the system resources on these target computers prior to deploying Windows, if necessary.

Identify Differences in 64-bit and 32-bit Deployments

Most functions and features found in 32-bit versions of Windows are the same in 64-bit versions of Windows. However, take the following differences into consideration when deploying 64-bit versions of Windows:

- For LTI deployments, the version of Windows PE must match the version of Windows being deployed. If deploying a 64-bit version of Windows, use a 64-bit version of Windows PE.

- Applications are installed in separate Program Files folders. On 64-bit versions of Windows, 64-bit applications are installed in the Program Files folder, and 32-bit applications are installed in the Program Files (x86) folder. Check the appropriate folder structure when looking for previously installed applications.
- Processor architecture discovery in Windows Deployment Services may need to be forced for 64-bit computers. Not all 64-bit computers properly report the processor type; therefore, MDT may not properly detect that the processor is a 64-bit processor. Use the following command to force Windows Deployment Services to deploy 64-bit versions:

  Windows Command Prompt

  ```
  WDSUTIL /set-server /architecturediscovery:yes
  ```

  For more information, see the Windows Deployment Services Help files.
- 64-bit versions of Windows PE do not run 32-bit applications. Ensure that any compiled applications used by a 64-bit version of Windows PE are 64-bit versions.
- 64-bit versions of Windows require 64-bit device drivers. You cannot use 32-bit device drivers in 64-bit versions of Windows.

Planning Performance and Power Management

Windows includes a number of features that help improve the performance and power utilization of computers. You can incorporate these improvements as part of the configuration settings you deploy to the target computers using MDT.

Review the following resources to identify performance and power-management configuration settings to include when performing your target computer deployments:

- Windows Performance Analysis Tools
- Sustainable Computing: Enforce Power Management Settings in your Organization with Group Policy
- Mobile Battery Life Solutions for Windows 7
- Power Policy Configuration and Deployment in Windows

Planning Target Computer Security

When planning the configuration of the Windows operating systems for target computers, ensure that the target computers are deployed in compliance with the requirements in your organization. Microsoft has developed Security Solution Accelerators that can help you deploy your target computers in a secured configuration.

The Security Solution Accelerators include guidance and tools to help you secure Windows. For more information about deploying target computers in a secured configuration using these solution accelerators, see Security Solution Accelerators.

Choosing LTI, ZTI, or UDI Deployments

LTI, ZTI, and UDI deployments use the same common set of scripts and configuration files (such as CustomSettings.ini) for deploying target computers. Table 6 compares LTI, ZTI, and UDI deployments.

Table 6. Comparison of LTI, ZTI, and UDI Deployments

Expand table

LTI deployment	ZTI deployment	UDI deployment
Allows selection of the level of automation	Supports only fully automated deployments	Allows selection of the level of automation
Has minimal infrastructure requirements	Requires Configuration Manager	Requires Configuration Manager
Supports deployments over a network using a shared folder or locally using removable storage such as a CD, DVD, or UFD	Supports deployments over a network using Configuration Manager distribution points or locally using removable storage such as a CD, DVD, or UFD	Supports deployments over a network using Configuration Manager distribution points or locally using removable storage such as a CD, DVD, or UFD
The deployment process can be initiated manually or automatically using Windows Deployment Services	The installation process can be initiated by Configuration Manager, or Windows Deployment Services	The installation process can be initiated by Configuration Manager, or Windows Deployment Services
The deployment process is configured using the Deployment Workbench	The deployment process is configured using the Configuration Manager console	The deployment process is configured using the Configuration Manager console and the UDI Wizard Designer.
Can require less initial IT administration configuration	Requires more initial IT administration configuration	Requires more initial IT administration configuration

LTI deployment	ZTI deployment	UDI deployment
time	time	time
Can require interaction by the user or deployment technician	Requires no interaction by the user or deployment technician	Can require interaction by the user or deployment technician
Increases the risk of introducing configuration errors	Reduces the risk of introducing configuration errors	Increases the risk of introducing configuration errors
Requires users or deployment technicians to have credentials with elevated permissions	Users and deployment technicians are not required to have credentials with elevated permissions	Requires users or deployment technicians to have credentials with elevated permissions
Requires that users or deployment technicians know some configuration settings prior to initiating the MDT deployment process	Users and deployment technicians do not need to know configuration settings prior to initiating the MDT deployment process	Requires that users or deployment technicians know some configuration settings prior to initiating the MDT deployment process
Can be used with slow connections or in instances where no network connectivity exists	Requires a high-speed, persistent connection	Requires a high-speed, persistent connection
Requires little or no infrastructure to support deployment	Requires an infrastructure sufficient to deploy operating system images	Requires an infrastructure sufficient to deploy operating system images
Supports deployment over the network or local to the computer from media	Supports deployment over the network or local to the computer from media	Supports deployment over the network or local to the computer from media
Does not require management of target computers using Configuration Manager	Requires that target computers be managed using Configuration Manager	Requires that target computers be managed using Configuration Manager
Supports security policies in which automatic software installation is prohibited	Supports only security in which automatic software installation is allowed.	Supports only security in which automatic software installation is allowed.
Supports deployment to target computers isolated by firewalls	Requires remote procedure call (RPC) communication with target computers	Requires RPC communication with target computers

At some point in the MDT process, you must provide all the information necessary to install Windows and the applications on target computers. The question is, when do you provide this information? The more information you provide in advance, the less interaction is required during deployment.

Table 7 lists the advantages and disadvantages of performing fully automated deployments (using LTI, ZTI, or UDI) and partially automated deployments (using LTI or UDI).

Table 7. Advantages and Disadvantages of Fully and Partially Automated Deployments

Expand table

Method	Advantages	Disadvantages
Fully	- No interaction with the user or deployment technician is required. - The risk of introducing configuration errors is decreased. - Users or deployment technicians do not need to know any configuration information prior to initiating the MDT deployment process.	- More time is needed to provide configuration information required for fully automated deployment. - Credentials to access network resources and that have elevated permissions are stored in configuration files that must be protected.
Partially	- Less time is required to prepare for deployment, because configuration information can be provided interactively.	- Interaction with the user or deployment technician is required. - The risk of introducing configuration errors is increased. - Users or deployment technicians must have credentials that require elevated permissions. - Users or deployment technicians must know some configuration information prior to initiating the MDT deployment process.

Reviewing Known Issues, Limitations, and Recommendations for MDT

Review the know issues, limitations, and recommendations for:

- General issues in MDT as described in Review General Known Issues, Limitations, and Recommendations for MDT
- Windows as described in Review Known Issues, Limitations, and Recommendations That Relate to Windows
- Disks and partitioning as described in Review Known Issues, Limitations, and Recommendations That Relate to Disks and Partitioning
- BitLocker as described in Review Known Issues, Limitations, and Recommendations That Relate to BitLocker
- LTI deployments as described in Review Known Issues, Limitations, and Recommendations for LTI Deployments
- ZTI deployments using Configuration Manager as described in Review Known Issues, Limitations, and Recommendations for ZTI Deployments Using Configuration Manager
- UDI deployments as described in Review Known Issues, Limitations, and Recommendations for UDI Deployments
- Running task sequences on target computers as described in Review Known Issues, Limitations, and Recommendations for Running Task Sequences on Target Computers
- Saving and restoring user information as described in Review Known Issues, Limitations, and Recommendations for Saving and Restoring User Information

Review General Known Issues, Limitations, and Recommendations for MDT

The following are a list of known general issues, limitations, and recommendations that relate to MDT:

- MDT supports the Windows Assessment and Deployment Kit (Windows ADK) for Windows 8.1, Windows PE version 5.0, System Center 2012 R2 Configuration Manager.
- Language packs, applications, and device drivers that are disabled in the Deployment Workbench are not installed, unless you add them manually to the CustomSettings.ini file.

- When you select the **Install Language Pack Offline** action, you must select language pack CAB files within subfolders of the main package.
- When you specify IP addresses (for example, when identifying Domain Name System [DNS] and Windows Internet Naming Service [WINS] servers), they must exclude unnecessary zero prefixes, which will be misevaluated. For example, if the IP address is typed **10.010.10.1**, the Deployment Workbench evaluates it as **10.8.10.1**. To avoid this problem, enter IP addresses carefully, and do not add unnecessary zeroes.
- When specifying a Run As account, you must specify a user who is a member of the Administrators group on the server. Otherwise, the account will not have sufficient privileges to access network connections established by administrators.
- When creating deployment shares on computers that have 8.3 file names disabled (see the Microsoft Support article How to Disable the 8.3 Name Creation on NTFS Partitions), the Deployment Workbench fails to generate the Windows PE image. If 8.3 file names have been disabled, re-enable them by setting the **HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\FileSystem\NtfsDisable8dot3NameCreation** registry value to **0**.
- Within a folder, avoid creating a subfolder and a file with the same name. For example, within the Files folder, you should not name a subfolder Item, and then create a file named Item.
- When performing an upgrade, network and media deployment shares created in earlier versions of MDT must have valid shares.
- If custom images captured by directly running ImageX (without using MDT to capture the image) do not work properly, troubleshoot the issues by capturing and adding the image using MDT to ensure that all prerequisites are configured properly. Add Setup files to the Deployment Workbench by adding a complete operating system distribution or by pointing the Deployment Wizard to the location of source files. When manually capturing images, use the Wimscript.ini file that MDT supplies in the \Distribution\tools_*platform* folder, where *platform* is either x86 (for 32-bit) or x64 (for 64-bit), to exclude the folders or files from the image. Also be aware of preexisting Unattend.xml files in the image. Specify the correct **/FLAGS** value when capturing Windows images using ImageX or Windows Deployment Services capture processes.
- At the completion of MDT deployment, a summary page displays warnings about errors encountered during the process. (This page is not displayed when conducting the Server Core installation because it does not include the required

Windows Internet Explorer® components.) It is possible for deployment to finish but still trigger several errors or warnings if the errors are nonfatal. It is useful to inspect these errors and warnings—for example, by opening corresponding logs files and running verification tests—and determine whether they are unexpected.

- Some device drivers can stall the deployment process. If this happens, isolate the device driver and remove it from the target computer, or contact the vendor for an updated version of the device driver.
- The error "Cannot Find Driver Path" can appear if you choose to optimize a Windows PE image and select drivers that are not available from the distribution point. To work around this problem, select the**Completely Regenerate The Boot Images** option.
- The USMT fails if you enter a path surrounded by quotation marks that also ends in a backslash (\). To prevent problems, simply leave off the final backslash when specifying a folder. For example, the following paths will cause an error:
  - "D:\"
  - "D:\folder\"

    However, these paths will not cause an error:
  - D:
  - D:\
  - D:\folder\
  - D:\folder
  - "D:\folder"
  - "D:\"
  - "D:"
- Configure the Internet Explorer home page using CustomSettings.ini, in the MDT DB, or by using the Windows Internet Explorer Administration Kit (IEAK). Configuring the Internet Explorer home page works only in Windows unattended installation.
- During LTI deployments to new computers, some pages (such as the **User Locale** and **Keyboard Locale** pages) do not display text correctly if required fonts are not

installed in Windows PE. In the Refresh Computer scenario, the same symptoms appear if required fonts are not installed in the operating system being replaced.

- Some keyboard layouts might require language packs or input method editors that MDT does not automatically include in the Windows PE startup image. MDT does not verify that the keyboard layout is valid. For more information, see Custom Input Method Editor (IME) requirements.

- A maximum of two WINS server addresses can be added when configuring static IP configuration settings for a network adapter. If more than two WINS server addresses are added using MDT, only the first two WINS server addresses are used.

- Hash value errors may occur in Configuration Manager during download on demand or when creating a media deployment DVD. This can occur if the packages on the distribution points are inconsistent with the information in the Configuration Manager database.

To correct hash value errors for Configuration Manager

1. Select **Start**, point to **All Programs**, and then point to **Microsoft System Center 2012**. Point to **Configuration Manager**, and then select **Configuration Manager Console**.

2. In the Configuration Manager console, in the **Navigation** pane, select **Software Library**.

3. In the Software Library workspace, go to Overview/Application Management/Packages.

4. In the preview pane, select ***package_name*** (where *package_name* is the name of the package that is inconsistent).

5. On the Ribbon, on the **Home** tab, in the **Properties** group, select **Properties**.

   The ***package_name*Properties** dialog box opens (where *package_name* is the name of the package).

6. In the ***package_name*Properties** dialog box (where *package_name* is the name of the package), on the **Content Locations** tab, in **Distribution points or distribution point groups** box, select ***distribution_point*** (where *distribution_point* is the name of the distribution point or distribution point group), then select **Redistribute**.

7. In the **Configuration Manager** dialog box, select **OK**.

8. Repeat steps f through g for each distribution point or distribution point group.
9. In the ***package_name*Properties** dialog box, select **OK**.
10. Repeat steps d through i for each package that is inconsistent.

- In LTI deployments, you set several kinds of information in the Create Task Sequence Wizard. The UI does not provide an option to edit this information at a later time. However, you can edit the information directly in the Unattend.xml file. The information includes:
  - Organization name
  - Full name
  - Internet Explorer home page
  - Local Administrator password
- No user state configuration settings can be or need to be specified using the CustomSettings.ini file for Configuration Manager scenarios. The network location is determined automatically by the **Request State Store** task.
- In Configuration Manager deployments, you can install multiple application packages on a computer by specifying them in the CustomSettings.ini file according to the following parameters:
  - Specify a base variable named PACKAGES in the task sequence in the **Install Software** task.
  - Each PACKAGES variable name should have a suffix starting with *001*.
  - The **PACKAGESxxx** value should have the format *PACKAGEID:ProgramName* (use a colon between items).
  - The **ProgramName** value is case-sensitive.

    The following is an example of specifying packages in CustomSettings.ini:

    ini

    ```ini
    PACKAGES001=DEP0002B:Install Office 2007
    PACKAGES002=DEP00011:Install Office Communicator
    ```

- In LTI New Computer deployments, applications marked as hidden in the Deployment Workbench are not installed when you do not skip the Application page in the Deployment Wizard and specify the application globally unique identifier (GUID) in CustomSettings.ini. Specify hidden applications using the **MandatoryApplications** property instead of the **Applications** property.

- Close the Configuration Manager console before running the integration option from MDT. Otherwise, some files may not be properly updated.

- During the Scanstate and Loadstate processes, multiple copies of log files may be created. You can use a new USMT template for excluding the log files or log directories while running Scanstate and Loadstate.

- New Computer and Replace Computer deployment scenarios format Disk 0 by default. Using MDT on computers with original equipment manufacturer (OEM) partitions or multiple fixed or external hard disks may require additional configuration and scripting in addition to thorough testing.

- The Task Sequencer will not accept XML files that contain Unicode content (from an XML file encoded as UTF-8). Attempting to do use XML files results in a task sequence failure. The Task Sequencer does not properly deal with encoded UTF-7 data: It does not cause a failure, but it does cause the data to be translated unexpectedly.

- After uninstalling MDT using the Control Panel Add or Remove Programs item, the Distribution share directory (if created) must be removed manually. MDT does not remove files or folders that it did not initially install.

- When using MDT extensions (add-in wizards) with Configuration Manager, MDT must be installed on every server running Configuration Manager used to administer operating system deployments.

- The **Priority** property in CustomSettings.ini has no maximum line length. However, if the property name is longer than 55 characters when the Configure Database Wizard runs, the wizard will truncate the **Priority** property, and you will need to manually edit the property. As a workaround, run the Configure Database Wizard before performing any other customization, or clear the check boxes for queries in the wizard that are not needed.

- MDT supports deployment from a UFD. See the Windows ADK for information about preparing the device, then copy (using the**xcopy** command) all files and folders from the \Media\content folder to the UFD.

- Dialing properties that are not configured, even if present in the answer file, include the country/region code, area code, long-distance access, and dialing rules. To work around this issue, configure dialing rules by creating and testing a .reg file in a lab environment, and then import that .reg file as a custom task during the task sequence.

Review Known Issues, Limitations, and Recommendations That Relate to Windows

The following is a list of known issues, limitations, and recommendations that relate to Windows:

- Deployment will fail on computers configured for a language other than English when the Windows Media® Player Network Sharing Service is run. As a workaround, stop the Windows Media Player Network Sharing Service until after deployment is complete.

- You can use AD DS to back up BitLocker and TPM data. Recovery information includes the recovery password for each encrypted value, the TPM owner password, and the information necessary to associate recovery information with computers and volumes. Another option is to save a package containing the keys used to encrypt data in addition to the recovery password required to access those keys. For more information, see BitLocker FAQ for AD DS in the Microsoft Download Center.

- When enabling BitLocker, key files are generated as hidden, read-only system files. To see them, set the Windows Explorer option to show hidden and system files.

- BitLockerduring LTI deployment requires at least two partitions. The first partition is the primary partition and can be any size; it stores operating system files and user data. In BitLocker terminology, this is called the *boot partition*. For Windows 7, it should be at least 300 MB. This partition stores startup files required during the first phase of startup and is called the *system partition*. A BitLocker partition is created for all Windows 7 deployments, regardless of whether you are deploying BitLocker.

- If a user with a limited account maps a drive (such as drive Z) to the MDT distribution point (*server*\distribution$, where *server* is the name of the computer hosting the distribution point), runs LiteTouch.vbs, and then provides Administrator credentials in the **User Credentials** dialog box, MDT displays the error, "Cannot find script file 'Z:\Scripts\LiteTouch.wsf' because the account that the user provided in the User Credentials dialog box cannot access the mapped drive created by the

limited user account." To resolve this issue, use an account with Administrator credentials to map the drive to the distribution point.

- BitLockerdeployment can fail with the error, "Unable to merge BDEPartition, return code=87," when the user does not specify a locale. Restarting the computer does not allow the operating system to start. To avoid this error, specify a user language, or edit the CustomSettings.ini file to specify the **UILanguage** property. For example, you could add `UILanguage = en-us` to the CustomSettings.ini file.

- If activating BitLocker during installation fails in Refresh Computer scenario, verify that MDT is able to shrink the partition as required by following these steps:

    1. At a command prompt, type **diskpart shrink querymax**, and note the value displayed.

    2. If the value is less than 2,000 MB, then manually defragment the disk. MDT performs an automatic defragmentation, however, so this might not resolve the problem.

    3. If defragmenting the disk does not resolve the issue, back up the computer's hard disk, create a new partition, and repeat these steps until typing **diskpart shrink querymax** returns a value greater than 2,000 MB. There might be files in specific areas of the partition that cannot be relocated or removed.

- The **BDERequired** flag is no longer used. By default, all sample templates that enable BitLocker and encounter an error will stop. You can edit the task sequence to enable deployment to continue if an error occurs.

- When deploying an image that is using a different language, Setup will prompt for the keyboard layout, language, and time and currency settings during the Windows PE phase. As a workaround, import Setup files with the custom image.

- MDT supports Windows language pack selection during deployment for all scenarios if the language packs are configured in the Deployment Workbench. Selecting multiple language packs is possible when deploying Enterprise or Ultimate editions of the operating systems. When other editions of Windows are deployed, only one language pack can be selected because of Windows licensing restrictions.

Review Known Issues, Limitations, and Recommendations That Relate to Disks and Partitioning

The following is a list of known issues, limitations, and recommendations that relate to disk and partitioning:

- LTI does not support the deployment of the target operating system to logical drives or dynamic disks.
- Deployments to existing disk partitions created by newer operating system versions are not supported in Refresh Computer deployment scenarios.

  However, you can deploy different processor architecture versions to the existing partitions created by the same operating system version. For example, you can deploy a 64-bit version of Windows on a computer that is currently running a 32-bit version of Windows or vice versa.
- In the **Format and Partition Disk** task sequence step types, always configure the logical partitions that will reside on an extended partition immediately after the extended partition. If you do not specify the logical partitions immediately after the extended partition, creating the logical partition sizes using a percentage produces unexpected results.

  For example, the following partition creation order is incorrect, because the logical partitions (partition 4 and partition 5) are not immediately after the extended partition (partition 2):

  Console

  ```
  Partition 1: Primary
  Partition 2: Extended
  Partition 3: Primary
  Partition 4: Logical
  Partition 5: Logical
  Partition 6: Primary
  ```

  Instead, create the partitions in the following order:

  Console

  ```
  Partition 1: Primary
  Partition 2: Extended
  Partition 3: Logical
  Partition 4: Logical
  Partition 5: Primary
  Partition 6: Primary
  ```

- Windows always hides the system volume during deployment, so a drive letter is not assigned to the system volume. For example, if the target computer has one drive with two partitions, Partition_1 and Partition_2, and you deploy Windows to Partition_2, Windows will be properly deployed to Partition_ 2. However, a drive letter will not be assigned to Partition_1.

- After starting Windows PE, the drive letters assigned to each storage device may change. For example, if the destination computer has a CD-ROM assigned to drive D and a hard disk drive assigned to drive E, the hard disk drive will be on drive D and the CD-ROM will be on drive E when Windows PE starts. If a DVD deployment fails, check that the drives have not been reassigned on the target computer. To simplify deployment, save user data to a network location instead of to a local drive.

- Avoid editing the Unattend.xml files to format or alter the partitions. MDT might store state and user data on the partition before calling Setup.exe (in LTI scenarios), and instructions added to Unattend.xml would cause Setup to destroy that data, resulting in a deployment failure.

- While configuring the **Format and Partition Disk** task, always specify the extended and logical partitions together, and do not add a primary partition in-between, which gives undesirable results when a logical partition size is configured using a percentage. In other words, do not add a primary partition between an extended and logical partition.

Review Known Issues, Limitations, and Recommendations That Relate to BitLocker

The following is a list of known issues, limitations, and recommendations that relate to BitLocker:

- Windows Server may crash if the operating system image used to perform the deployment does not have the optional BitLocker component. This situation can occur in the following scenarios:

    - Performing the MDT Refresh Computer deployment scenario (in LTI, ZTI, or UDI), where BitLocker is enabled on the existing operating system. In this situation, BitLocker is suspended in the existing operating system by MDT, but without the optional component in the new operating system image, Windows is unable to boot from the disk on which BitLocker is suspended.

    - Performing the MDT New Computer deployment scenario (in LTI, ZTI, or UDI) on a Trusted Platform Module-enabled server on which BitLocker has been

enabled. In this situation, BitLocker will be enabled offline using BitLocker pre-provisioning, but without the BitLocker optional component in the new operating system image, the new operating system is unable to boot from the disk on which BitLocker has been pre-provisioned.

The workaround for any of these situations is to deploy a custom operating system image that includes the BitLocker component in the image.

- If you want to use an alphanumeric PIN for BitLocker during deployment, you must enable the **Allow enhanced PINs for Startup** group policy setting. The**Allow enhanced PINs for Startup** group policy setting is located in Computer Configuration/Policies/Administrative Templates/Windows Components/BitLocker Drive Encryption/Operating System Drives.
- If a BitLocker recovery prompt appears after restarting the target computer (because the BitLocker key required to unlock the volume could not be obtained), work around the problem by using one of the following approaches:
  - Remove the media (such as the deployment DVD) while Windows PE is still running. Doing so prevents the operating system from seeing the DVD when it starts.
  - Change the boot order of the computer so that the DVD drive follows the hard disk.
  - Deploy the computer with no startup media; for example, use a Pre-Boot Execution Environment (PXE) deployment.

Review Known Issues, Limitations, and Recommendations for LTI Deployments

The following is a list of known issues, limitations, and recommendations that relate to LTI deployments:

- The network credentials specified for accessing network resources (the USMT store location, computer backup location, and so on) are not validated if a user is logged on to the computer using a domain account and if the computer already has a connection established to another share on the same server.

Review Known Issues, Limitations, and Recommendations for ZTI Deployments Using Configuration Manager

The following is a list of known issues, limitations, and recommendations that relate to ZTI deployments using Configuration Manager:

- When deploying a non-English-language target operating system, the installation method prompts for user language, because the template for the unattend.xml file contains settings for United States English (en-us). To work around this problem, perform one of the following tasks:
    - Modify the unattend.xml template file to reflect the language of the target operating system.
    - Configure the **KeyboardLocale**, **UserLocale**, and **UILanguage** properties in the CustomSettings.ini file or the MDT DB to reflect the language of the target operating system.
- When deploying computers using Configuration Manager and backing up the computer data locally, computers with two partitions may not be able to retain the backup. To prevent backups from being removed, save to a network location instead of to a local drive.
- In a Configuration Manager task sequence, the **Format and Partition** task might not run successfully on a computer if it has only one unformatted partition. To work around this issue, either remove the partition or format it.
- While installing the server roles, Configuration Manager might display a prompt for DLLs needed to complete the role installation. If this happens, specify a valid location for the required files. To avoid this step, add a step earlier in the task sequence that copies the required DLLs to the Windows Setup files folder defined in the registry. This folder location is defined in the **SourcePath** registry value, located in **HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Setup**.

Review Known Issues, Limitations, and Recommendations for UDI Deployments

The following is a list of known issues, limitations, and recommendations that relate to UDI deployments:

- Applications are disabled and cannot be automatically installed. This issue arises when the application requires administrator approval but has not yet been approved. If the **Require administrator approval if users request this application**check box is selected for the application, verify that the application has been approved.

For more information on how to require administrator approval and grant approval, see How to Deploy Applications in Configuration Manager.

- When performing the MDT Refresh Computer deployment scenario with a USB hard disk attached, task sequence errors may occur, because the Configuration Manager task sequencer placed the _SMSTaskSequence folder on the USB drive. By default, the Configuration Manager task sequencer places the _SMSTaskSequence folder on the drive with the most available free disk space, which can cause problems later in the deployment process if the USB drive is removed.

  If the _SMSTaskSequence folder is located on a USB drive, the **CheckSMSFolderOnUSB** UDI task will detect this condition and prevent the deplomen from continuing. To resolve this issue and perform the deployment, complete the following steps:

  1. Disconnect the USB drive from the target computer before starting the task sequence.
  2. Start the task sequence.
  3. Wait until the UDI Wizard starts.
  4. Connect the USB drive.
  5. Complete the UDI Wizard.

Review Known Issues, Limitations, and Recommendations for Running Task Sequences on Target Computers

The following is a list of known issues, limitations, and recommendations for running task sequences on target computers in MDT:

- For LTI deployments, ensure that User Account Control (UAC) is disabled for the built-in local Administrator account on the target computers until the task sequence finishes. Running task sequences on computers with UAC enabled for the local Administrator account causes task sequences to fail.

  > **Note**
  >
  > UAC should be disabled only for the built-in local Administrator account and enabled for all other accounts. By default, the built-in local Administrator account is excluded from UAC because of the **User Account Control: Admin**

Approval Mode for the built-in Administrator account (disabled) policy setting.

For more information about UAC Group Policy settings, see UAC Group Policy Settings and Registry Key Settings.

Review Known Issues, Limitations, and Recommendations for Saving and Restoring User Information

The following is a list of known issues, limitations, and recommendations for saving and restoring user information in MDT:

- For LTI deployments, do not add any of the following USMT command-line parameters to the **ScanStateArgs** or **LoadStateArgs** properties, as they cause the saving and restoration of user state information to fail:
  - **/hardlink**
  - **/nocompress**
  - **/encrypt**
  - **/key**
  - **/keyfile**
  - **/vsc**
  - **/l**
  - **/l**
- User state migration data may not be restored properly depending on the disk configuration of the target computer when deploying Windows.

  This issue can occur when the target computer has two physical hard disks, Disk_0 and Disk_1. Disk_0 contains the C volume, which is encrypted using BitLocker. The MDT deployment process is configured to deploy a new operating system to Disk_1. The problems in the deployment process occur as follows:

  - Early the deployment process, the minint and smstasksequence folders are copied to the existing C volume, which is encrypted.
  - Disk_1 is partitioned and formatted properly in preparation for deploying the target operating system.

- The target operating system is installed on the new partition and disk volume on Disk_1.
- During the State Restore Phase, the original C volume is not assigned a drive letter, so the task sequence steps in the State Restore Phase cannot access the minint and smstasksequence folders on the BitLocker-encrypted drive. The restoration of the user state migration data fails.

  The result is that the target operating system is installed, but the restoration of the user state migration data fails.

Overview of UDI

Typically, when deploying operating systems using the OSD feature in Configuration Manager and ZTI in MDT you must provide all the information necessary to deploy the operating system. Prior to performing the deployment, the information is configured in configuration files or in databases (such as the CustomSettings.ini file or the MDT DB). During the ZTI deployment process, ZTI converts the appropriate configuration settings to task sequence variables, which are consumed by the MDT task sequences for UDI. All of the configuration settings must be provided before you can initiate the deployment.

UDI provides a wizard driven interface that allows users to provide configuration information immediately prior to the deployment being performed. You can configure the user experience in the wizard, which lets you control the amount information the user completing the wizard must provide. This division of user roles allows IT pros to provide precise control over deployments while reducing the load on them by allowing other users to perform the deployments. The interface allows you to create generic OSD task sequences, and then provide computer specific information at the time of deployment, which provides greater flexibility in the deployment process.

> ⓘ **Note**
>
> If you are unfamiliar with UDI, review the UDI terms and terminology in "UDI Concepts" in the *Microsoft Deployment Toolkit Reference*. Familiarizing yourself with these terms and terminology will help you be more successful in applying the remainder of this guide to your organization.

Installing or Upgrading to MDT

To prepare for performing deployments using MDT, perform the following tasks:

1. Review the known issues, limitations, and recommendations for preparing disks on target computers in MDT as described in Reviewing Known Issues, Limitations, and Recommendations for Installing or Upgrading to MDT.

2. Prepare the prerequisite infrastructure required for the LTI, ZTI, and UDI deployments methods as described in Preparing the Prerequisite Infrastructure for All MDT Deployment Methods.

3. Perform any combination of the following steps to ensure that MDT is installed correctly:

    - Install a new instance of MDT on each computer where you want to manage MDT deployment shares as described in Installing a New Instance of MDT.

    - Upgrade an existing instance of MDT 2012 Update 1 as described in Upgrading to MDT.

4. Determine whether any updates are available for the components in the Deployment Workbench using the Check Updates Wizard as described in Upgrading to MDT.

> ⓘ **Note**
>
> Windows PowerShell™ version 2.0 or later must be installed on any computer on which MDT is installed for management of LTI or ZTI deployments.

Reviewing Known Issues, Limitations, and Recommendations for Installing or Upgrading to MDT

The following is a list of known issues, limitations, and recommendations for installing MDT:

- Ensure that the disk volume that contains the temporary folder that the Deployment Workbench uses has at least 20 GB of available disk space.

  The Deployment Workbench creates large images and requires temporary storage during the image-creation process. The Deployment Workbench determines the temporary folder to use by performing the following steps:

    1. Use the temporary folder specified in the **Temp_Dir** registry subkey, which is located at **HKEY_LOCAL_MACHINE\Software\Microsoft\Deployment 4**. Create the **Temp_Dir** registry subkey as a **REG_SZ** type that contains the fully qualified path to the folder to be used as the temporary folder.

2. If the **TEMP_DIR** registry subkey does not exist, then the Deployment Workbench uses the folder specified in the %TEMP% environment variable.

   Ensure that the disk volume specified in the **TEMP_DIR** registry subkey or in the %TEMP% environment variable has sufficient available disk space.

Preparing the Prerequisite Infrastructure for All MDT Deployment Methods

MDT requires installation of the following software for LTI, ZTI, and UDI:

- Microsoft .NET Framework version 3.5 with SP1
- Windows PowerShell version 2.0

For specifics about how to prepare your environment specifically for LTI, ZTI, or UDI, see the following sections:

- Preparing the LTI Deployment Environment
- Preparing the ZTI Deployment Environment for Configuration Manager
- Preparing the UDI Deployment Environment

Installing a New Instance of MDT

With all the prerequisite software installed, perform the following steps to install MDT (MicrosoftDeploymentToolkit_*platform*.msi, where *platform* is either x86 or x64):

1. Double-click **MicrosoftDeploymentToolkit2012_x64.msi** (for 64-bit operating systems) or **MicrosoftDeploymentToolkit2012_x86.msi** (for 32-bit operating systems).

   The Microsoft Deployment Toolkit 2013 Setup Wizard starts.

2. Complete the Microsoft Deployment Toolkit 2013 Setup Wizard using the information in Table 8.

Table 8. Information for Completing the Microsoft Deployment Toolkit 2013 Setup Wizard

Expand table

On this wizard page	Do the following
Welcome to the Microsoft Deployment Toolkit 2013 Setup Wizard	Select **Next**.
End-User License Agreement	- Review the license agreement. - Select the **I accept the terms in the License Agreement** check box, and then select **Next**.
Custom Setup	- Select the desired features. - Select the desired destination folder for installing MDT (which defaults to C:\Program Files\Microsoft Deployment Toolkit), and then select **Next**.
Ready to install Microsoft Deployment Toolkit 2013	Select **Install**.
Installing Microsoft Deployment Toolkit 2013	The progress for installing the Microsoft Deployment Toolkit 2013 is displayed.
Completed the Microsoft Deployment Toolkit 2013 Setup Wizard	Select **Finish**.

Upon completion, MDT is installed in the target folder you selected in the wizard.

Upgrading to MDT

MDT automatically uninstalls previous versions before installing, including the following versions:

- MDT 2012 Update 1

  In addition to upgrading the MDT installation, upgrade any existing deployment shares. For more information on this process, see Upgrade an Existing Deployment Share in the Deployment Workbench.

Performing LTI Deployments

You perform LTI deployments using only MDT and supporting components. You can perform LTI deployments over a network or from removable media. This flexibility makes LTI deployments appropriate for a wide range of organizations.

Perform LTI deployments by:

- Preparing the deployment environment as described in Preparing the LTI Deployment Environment
- Preparing for deployment to the reference computer as described in Preparing for LTI Deployment to the Reference Computer
- Deploying to and capture a reference computer image as described in Deploying To and Capturing an Image of the Reference Computer in LTI
- Preparing for deployment to target computers as described in Preparing for LTI Deployment to Target Computers
- Deploying captured images to target computers as described in Deploying Captured Images to Target Computers in LTI

Preparing the LTI Deployment Environment

After preparing the prerequisite infrastructure for MDT, prepare the LTI deployment environment.

To prepare the LTI deployment environment

1. Install the prerequisite LTI infrastructure as described in Prepare the Prerequisite LTI Infrastructure.
2. Install a new instance of MDT on the deployment server, or upgrade an existing instance of MDT as described in Install or Upgrade to MDT for LTI Deployments.
3. Install the components required by MDT and the LTI deployment process as described in Install Components That MDT and LTI Require.
4. Obtain the software that the LTI deployment process requires as described in Obtain the Software That the LTI Deployment Process Requires.

Prepare the Prerequisite LTI Infrastructure

LTI deployments require that a properly configured infrastructure exist prior to installing MDT and performing deployments. Ensure that your new or existing infrastructure is specifically optimized for the operating system deployments.

> ⓘ **Note**

Windows PowerShell version 2.0 or later must be installed on any computer on which MDT is installed for management of LTI deployments.

For more information about configuring your environment to support LTI deployments, see the following sections in the MDT document *Quick Start Guide for Lite Touch Installation*:

- "Prerequisites"
- "Step 1: Obtain the Required Software"

Install or Upgrade to MDT for LTI Deployments

To perform LTI deployments, you must have at least one instance of MDT running in your environment. If your existing environment has:

- No computers currently running MDT or a previous version of MDT, then install one or more new instances of MDT as described in Installing a New Instance of MDT
- One or more computers running a previous version of MDT, then upgrade those instances to MDT as described in Upgrading to MDT.

Install Components That MDT and LTI Require

The Deployment Workbench is the administration console for LTI. Most of the LTI management tasks are performed in the Deployment Workbench. MDT also includes a Windows PowerShell provider that allows for the automation of LTI management tasks through the Windows PowerShell command shell using MDT cmdlets.

> **Note**
>
> MDT supports Windows ADK for Windows 8.1, Windows PE 5.0, and System Center 2012 R2 Configuration Manager.

Table 10 lists the top-level nodes in the Deployment Workbench and the types of tasks performed in each node.

Table 10. Deployment Workbench Nodes

Expand table

Node	Description
Information Center	Provides access to documentation, displays breaking news about MDT, and lists the components required to use the Deployment Workbench.
Deployment Shares	Lists the deployment shares that this instance of the Deployment Workbench manages. Each deployment share includes operating systems, applications, operating system packages, task sequences, and out-of-box drivers populated in the deployment share.

The Deployment Workbench automates the download and installation of components used in LTI.

> ⓘ **Note**
>
> If the MDT computer has internet connectivity, the Deployment Workbench can automatically download the components.

To download and install Deployment Workbench components

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Information Center/Components.
3. In the Components pane, in the **Available for Download** section, select **component_name** (where *component_name* is the name of the component you want to download).
4. In the details pane, select **Download**.

   After downloading the component from the internet, the component is listed in the **Downloaded** section in the details pane.
5. In the details pane, in the **Download** section, select the downloaded component, and then select **Install**.

   > ⓘ **Note**
   >
   > If there is no **Install** button, the component cannot be installed or the installation is not necessary.

The installation process for the respective component is initiated. Complete the installation process for the component using the instructions provided for the component.

After the component is installed, the component appears in the **Installed** section of the details pane. If the component cannot be installed or you did not choose to install it, the component remains in the **Downloaded** section.

MDT uses the Windows ADK for Windows 8.1 which includes USMT.

Obtain the Software That the LTI Deployment Process Requires

Collect the software that LTI will deploy. LTI will import or add this software to deployment shares. The software that can be deployed includes:

- Operating system source files or image files for each operating system to be deployed to the reference and target computers
- Operating system packages for the operating systems, such as security updates, feature packs, or language packs
- Device drivers for the reference and target computers that are not included as a part of the operating system
- Applications that are to be installed as a part of the operating system image or during the deployment of the reference image

The Task Sequencer used in MDT deployments requires that the Create Global Object right be assigned to credentials used to access and run the Deployment Workbench and the deployment process. This right is typically available to accounts with Administrator-level permissions (unless explicitly removed). Also, the Specialized Security - Limited Functionality (SSLF) security profile, which is part of the Windows security baselines, removes the Create Global Object right and should not be applied to computers being deployed using MDT.

In addition, LTI requires that the:

- Local Administrator account on the target computer be enabled
- Local Administrator account be granted the Create Global Objects right
- Local Administrator account not have UAC enabled
- Ability to have an account automatically log on to the target computer be enabled and not blocked using Group Policy or a local security policy

- Logon banner for the target computers not be enabled using Group Policy or a local security policy

Preparing for LTI Deployment to the Reference Computer

For many of the LTI deployment scenarios, best practice is to create a reference computer as described in Choosing Thick, Thin, or Hybrid Images, and then capture an image of that computer. Later in the LTI deployment process, you will deploy the captured image of your reference computer to the appropriate target computers.

> ⓘ **Note**
>
> In some LTI deployment scenarios, you may want to deploy Windows operating systems without creating a reference image—for example, when you want to deploy thin images. If you are deploying thin images and you do not want to create a reference image, skip the steps that relate to the reference computer. For more information about determining which image type to use, see **Choosing Thick, Thin, or Hybrid Images**.

Create a reference computer for each image you want to create for deployment to the target computers. For more information about determining the number of images required in your organization and subsequently the number of reference computers required, see Estimate Storage Requirements for MDT Deployment Shares.

For more information about the use of reference computers in MDT-based deployments, see Using Reference Computers in MDT Deployments.

To prepare for deployment to the reference computer

1. Create a new deployment share, or upgrade an existing deployment share.For more information about:
    - Creating a new deployment share, see Create a New Deployment Share in the Deployment Workbench
    - Upgrading an existing deployment share, see Upgrade an Existing Deployment Share in the Deployment Workbench
2. Configure the deployment share to contain the appropriate software for deployment to the reference computer, including the following:

- Configuring operating systems as described in Configuring Operating Systems in the Deployment Workbench
- Configuring applications as described in View and Configure an Application in the Deployment Workbench
- Configuring operating system packages as described in Configuring Packages in the Deployment Workbench
- Configuring device drivers as described in Configuring Device Drivers in the Deployment Workbench

3. Create a new task sequence that will deploy the target operating system to the reference computer as described in:

   - Configuring Task Sequences in the Deployment Workbench
   - Configuring LTI Task Sequence Steps in the Deployment Workbench

     The LTI deployment process is unable to perform Sysprep operations on a target computer that is encrypted using BitLocker. Ensure that you do not enable BitLocker on the reference computer. Enable BitLocker on the target computers only after the target operating system is completely deployed.

   > **Tip**
   >
   > Create the task sequence for deploying to the reference computer based on the Standard Client Task Sequence template, included in MDT.

4. Customize the MDT configuration files to the needs of your organization as described in Configuring MDT Deployments.

5. Configure any Windows PE options for the deployment share as described in:

   - Configure the Deployment Share Properties Windows PE x86 Settings Tab
   - Configure the Deployment Share Properties Windows PE x86 Components Tab
   - Configure the Deployment Share Properties Windows PE x64 Settings Tab
   - Configure the Deployment Share Properties Windows PE x64 Components Tab

6. Update the deployment share to create the Windows PE images used to start LTI deployment as described in Update a Deployment Share in the Deployment Workbench.

Deploying To and Capturing an Image of the Reference Computer in LTI

After you have configured the deployment share, updated the deployment share, and created the Windows PE images that include the LTI scripts, start the reference computer with the Windows PE image, and complete the Deployment Wizard. The task sequence you created earlier in the process will deploy the target operating system, device drivers, operating system packages, and applications to the reference computer, and then capture an image of the reference computer.

To deploy to and capture an image of the reference computer

1. Create the LTI bootable media used to start the reference computer as described in Create the LTI Bootable Media.
2. Complete the Deployment Wizard to deploy and capture an image of the reference computer as described in Complete the Deployment Wizard.
3. Add the captured reference computer image to the Operating Systems node in the Deployment Workbench as described in Add the Captured Image of the Reference Computer to the Deployment Workbench.

Create the LTI Bootable Media

You must provide a method for starting the computer with the customized version of Windows PE you created when you updated the deployment share. The Deployment Workbench creates the LiteTouchPE_x86.iso and LiteTouchPE_x86.wim files (for 32-bit target computers) or the LiteTouchPE_x64.iso and LiteTouchPE_x64.wim files (for 64-bit target computers) in the *deployment_share*\Boot folder (where *deployment_share* is the network shared folder used as the deployment share). Create the appropriate LTI bootable media from one of these images.

To create the LTI bootable media

1. In Windows Explorer, go to *deployment_share*\Boot (where *deployment_share* is the network shared folder used as the deployment share).

> **Tip**
>
> To determine the location of the deployment share, view the properties of the share in the Deployment Workbench.

2. Based on the type of computer used for the reference computer, perform one of the following tasks:

   - If the reference computer is a physical computer, create a UFD, CD, or DVD of the ISO file.
   - If the reference computer is a VM, start the VM directly from the ISO file or from a CD or DVD of the ISO file.

Complete the Deployment Wizard

Start the reference computer with the LTI bootable media you created earlier in the process. The LTI bootable media starts Windows PE on the reference computer and initiates the deployment process. At the end of the process, the target operating system is deployed on the reference computer, and an image of the reference computer is captured.

> **Note**
>
> You could also initiate the process by starting the target computer from Windows Deployment Services. For more information, see Preparing Windows Deployment Services for LTI Deployments.

To complete the Deployment Wizard

1. Start the reference computer with the LTI bootable media you created earlier in the process.

   Windows PE starts, and then the Deployment Wizard starts.

2. Complete the Deployment Wizard as described in Running the Deployment Wizard, ensuring that you specifically follow the configuration settings on the wizard pages listed in Table 11 and selecting the values on the other wizard pages appropriate to your organization.

Table 11. Information for Completing the Deployment Wizard

Expand table

On this wizard page	Do this
Select a task sequence to execute on this computer	Select the task sequence you created for the reference computer deployment.
Join the computer to a domain or workgroup	Join a workgroup. Joining the reference computer to a domain causes problems when deploying the captured image of the reference computer to target computers. The most common symptom of trouble is that the deployment process halts, because the LTI process is not able to automatically log on.
Specify whether to capture an image	Select **Capture an image of this reference computer**, and provide the fully qualified Universal Naming Convention (UNC) path for the name of the captured Windows Imaging Format (WIM) image, including the WIM file name.

The wizard starts, and the operating system deployment starts. At the end of the deployment process, the **Deployment Summary** dialog box appears.

3. In the **Deployment Summary** dialog box, select **Details**.

   If any errors or warnings occur, review them, and record any diagnostic information. For more information about the errors or warnings, see the MDT document *Troubleshooting Reference*.

4. In the **Deployment Summary** dialog box, select **Finish**.

   The reference computer is now deployed, and the captured WIM file of the reference computer is stored in the location you specified on the **Specify whether to capture an image** wizard page.

Add the Captured Image of the Reference Computer to the Deployment Workbench

To deploy the captured image of the reference computer to the target computer, add the captured image to the list of operating systems in the Operating Systems node in

the Deployment Workbench. The Import Operating System Wizard copies the operating system files to the *deployment_share*\Operating Systems*operating_system* folder (where *deployment_share* is the deployment share folder created earlier in the process and *operating_system* is the name of the operating system added to the deployment share).

Add the captured image of the reference computer by completing the operating system import process as described in Import a Previously Captured Image of a Reference Computer, ensuring that you specifically follow the configuration settings on the wizard pages listed in Table 12 and selecting the values on the other wizard pages that are appropriate to your organization.

Table 12. Information for Completing the Importing a Previously Captured Image of a Reference Computer

Expand table

On this wizard page	Do this
Image	In **Source File**, specify the fully qualified path to the WIM file of the captured image of the reference computer.

The Import Operating System Wizard finishes. The captured image of the reference computer is added to the list of operating systems in the information pane and is copied to the deployment share.

Preparing for LTI Deployment to Target Computers

With the images of the reference computer captured, deploy the images to the target computers. In preparation, create one or more task sequences for deploying the captured images, ensure that the necessary deployment resources exist, and customize the deployment process.

To prepare for deployment to the target computers

1. Prepare network shares for storing migration data and MDT deployment logs as described in Preparing the MDT Migration Resources.
2. Create additional deployment shares to help in larger deployments as described in Create a New Deployment Share in the Deployment Workbench.

3. Optionally, prepare Windows Deployment Services to start the appropriate Windows PE images that will in turn start LTI deployment to the target computers as described in Preparing Windows Deployment Services for LTI Deployments (if you are using Windows Deployment Services to start the process).

4. Prepare the MDT task sequences, the MDT configuration files, and the MDT DB for each deployment scenario as described in:

    - Prepare for the New Computer Deployment Scenario to Target Computers Using LTI
    - Prepare for a Refresh Computer Deployment Scenario to Target Computers Using LTI
    - Prepare for a Replace Computer Deployment Scenario to Target Computers Using LTI

    Depending on the target computers in your organization, any combination of the deployments scenarios might be necessary. For more information about the MDT deployment scenarios, see Identifying Deployment Scenarios.

Prepare for the New Computer Deployment Scenario to Target Computers Using LTI

In the New Computer deployment scenario, a new installation of a Windows operating system is deployed to a new computer. There is no user migration information to save and restore and no existing file systems to preserve. Use the Standard Client Task Sequence or Standard Server Task Sequence templates to deploy the captured image of the reference computer to the target computer.

To prepare for the New Computer deployment scenario

1. Create a new task sequence that will deploy the captured image of the reference computer to the target computer as described in the following list, ensuring that you specifically follow the configuration settings on the wizard pages listed in Table 13 and select the values on the other wizard pages appropriate to your organization:

    - Configuring Task Sequences in the Deployment Workbench
    - Configuring LTI Task Sequence Steps in the Deployment Workbench

    > **Tip**

Create the task sequence for deploying to the target computer based on the Standard Client Task Sequence or Standard Server Task Sequence templates included in MDT.

Table 13. Information for Completing the New Task Sequence Wizard for Performing New Computer Deployment Scenario

Expand table

On this wizard page	Do this
Select OS	Select the captured image of the reference computer.

2. Customize the MDT configuration files to the needs of your organization as described in Configuring MDT Deployments.

3. Optionally, customize the MDT DB to the needs of your organization as described in Performing Deployments Using the MDT DB (if you are using the MDT DB to provide MDT configuration information).

4. Verify the Windows PE options for each deployment share as described in:

   - **Configure the Deployment Share Properties Windows PE x86 Settings Tab**
   - Configure the Deployment Share Properties Windows PE x86 Components Tab
   - Configure the Deployment Share Properties Windows PE x64 Settings Tab
   - Configure the Deployment Share Properties Windows PE x64 Components Tab

5. Update each deployment share, linked deployment share, and media to create the Windows PE images used to start LTI deployment as described in:

   - Update a Deployment Share in the Deployment Workbench
   - Replicate Linked Deployment Shares in the Deployment Workbench
   - Generate Media Images in the Deployment Workbench

Prepare for a Refresh Computer Deployment Scenario to Target Computers Using LTI

In the Refresh Computer deployment scenario, a computer is refreshed—that is, re-imaged for image standardization or to address a problem. You must save and restore the user migration information, because the existing file systems on the target computer are not preserved. Use the Standard Client Task Sequence or Standard Server Task Sequence templates to deploy the captured image of the reference computer to the target computer.

To prepare for the Refresh Computer deployment scenario

1. Create a new task sequence that will deploy the captured image of the reference computer to the target computer as described in the following list, ensuring that you specifically follow the configuration settings on the wizard pages listed in Table 14 and select the values on the other wizard pages that are appropriate to your organization:

   - Configuring Task Sequences in the Deployment Workbench
   - Configuring LTI Task Sequence Steps in the Deployment Workbench

   > **Tip**
   >
   > Create the task sequence for deploying images to the target computer based on the Standard Client Task Sequence or Standard Server Task Sequence templates included in MDT.

Table 14. Information for Completing the New Task Sequence Wizard for Performing a Refresh Computer Deployment Scenario

Expand table

On this wizard page	Do this
Select OS	Select the captured image of the reference computer.

2. Customize the MDT configuration files to the needs of your organization as described in Configuring MDT Deployments.

3. Optionally, customize the MDT DB to the needs of your organization as described in Performing Deployments Using the MDT DB (if you are using the MDT DB to provide MDT configuration information).

4. Verify the Windows PE options for each deployment share as described in:

    - Configure the Deployment Share Properties Windows PE x86 Settings Tab
    - Configure the Deployment Share Properties Windows PE x86 Components Tab
    - Configure the Deployment Share Properties Windows PE x64 Settings Tab
    - Configure the Deployment Share Properties Windows PE x64 Components Tab

5. Update each deployment share, linked deployment share, and media to create the Windows PE images used to start LTI deployment as described in:

    - Update a Deployment Share in the Deployment Workbench
    - Replicate Linked Deployment Shares in the Deployment Workbench
    - Generate Media Images in the Deployment Workbench

Prepare for a Replace Computer Deployment Scenario to Target Computers Using LTI

In the Replace Computer deployment scenario, one computer replaces another computer. The existing user state migration data is saved from the original computer to a network shared folder or removable media. Then, a new installation of Windows is deployed to a new computer. Finally, the user state data is restored to the new computer, because the file systems on the new computer are formatted as part of the new installation of Windows. Use the:

- Standard Client Replace Task Sequence template to save the user state migration data of the existing target computer

    > **Important**
    >
    > Run this task sequence on the existing target computer before running the task sequence based on the Standard Client Task Sequence template on the new target computer.

- Standard Client Task Sequence template to deploy the captured image of the reference computer to the new target computer and restore the user state migration data

  > **Important**
  >
  > Run this task sequence on the new target computer after running the task sequence based on the Standard Client Replace Task Sequence template on the existing target computer.

To prepare for the Replace Computer deployment scenario

1. Create a new task sequence that will save the user state migration data of the existing target computer as described in:

   - Configuring Task Sequences in the Deployment Workbench
   - Configuring LTI Task Sequence Steps in the Deployment Workbench

   > **Tip**
   >
   > Create the task sequence for capturing the user state of the existing target computer based on the Standard Client Task Replace Sequence template included in MDT.

2. Create a new task sequence that will deploy the captured image of the reference computer to the new target computer and restore the user state migration data saved by the task sequence in the previous step as described in the following list, ensuring that you specifically follow the configuration settings on the wizard pages listed in Table 15 and select the values on the other wizard pages that are appropriate to your organization:

   - Configuring Task Sequences in the Deployment Workbench
   - Configuring LTI Task Sequence Steps in the Deployment Workbench

   > **Tip**
   >
   > Create the task sequence for deploying to the new target computer based on the Standard Client Task Sequence template, included in MDT.

Table 15. Information for Completing the New Task Sequence Wizard for Performing the Refresh Computer Deployment Scenario

Expand table

On this wizard page	Do this
Select OS	Select the captured image of the reference computer.

3. Customize the MDT configuration files to the needs of your organization as described in Configuring MDT Deployments.
4. Optionally, customize the MDT DB to the needs of your organization as described in Performing Deployments Using the MDT DB (if you are using the MDT DB to provide MDT configuration information).
5. Verify the Windows PE options for each deployment share as described in:
   - Configure the Deployment Share Properties Windows PE x86 Settings Tab
   - Configure the Deployment Share Properties Windows PE x86 Components Tab
   - Configure the Deployment Share Properties Windows PE x64 Settings Tab
   - Configure the Deployment Share Properties Windows PE x64 Components Tab
6. Update each deployment share, linked deployment share, and media to create the Windows PE images used to start LTI deployment as described in:
   - Update a Deployment Share in the Deployment Workbench
   - Replicate Linked Deployment Shares in the Deployment Workbench
   - Generate Media Images in the Deployment Workbench

Deploying Captured Images to Target Computers in LTI

The deployment of the captured images to the target computers is slightly different for LTI. Deploy the captured image of the reference computer to target computers for each of the deployment scenarios in your organization as described in:

- Deploy Captured Images to Target Computers in the New Computer Deployment Scenario Using LTI
- Deploy Captured Images to Target Computers in a Refresh Computer Deployment Scenario Using LTI
- Deploy Captured Images to Target Computers in a Replace Computer Deployment Scenario Using LTI

  Depending on the target computers in your organization, any combination of the deployment scenarios might be necessary. For more information about the MDT deployment scenarios, see Identifying Deployment Scenarios.

Deploy Captured Images to Target Computers in the New Computer Deployment Scenario Using LTI

Start the target computer with the LTI bootable media you created earlier in the process or from Windows Deployment Services. The LTI bootable media starts Windows PE on the target computer and initiates deployment. At the end of the process, the captured image of the reference computer is deployed on the target computers.

To complete the Deployment Wizard

1. Start the reference computer with the LTI bootable media you created earlier in the process or from Windows Deployment Services.

   Windows PE starts, and then the Deployment Wizard starts.

2. Complete the Deployment Wizard as described in Running the Deployment Wizard, ensuring that you specifically follow the configuration settings on the wizard pages listed in Table 16 and select the values on the other wizard pages appropriate to your organization.

Table 16. Information for Completing the Deployment Wizard for the New Computer Deployment Scenario Using LTI

Expand table

On this wizard page	Do this
Select a task sequence to execute on this computer	Select the task sequence you created for the target computer deployment for the New Computer deployment scenario.

The wizard starts, and the operating system deployment starts. At the end of the process, the **Deployment Summary** dialog box appears.

3. In the **Deployment Summary** dialog box, select **Details**.

   If any errors or warnings occur, review them, and record any diagnostic information. For more information about the errors or warnings, see the MDT document *Troubleshooting Reference*.

4. In the **Deployment Summary** dialog box, select **Finish**.

   The target computers are now deployed.

Deploy Captured Images to Target Computers in a Refresh Computer Deployment Scenario Using LTI

Start the Deployment Wizard on the existing operating system on the target computer to start the **Standard Client** task sequence or **Standard Server** task sequence created earlier in the process. The Deployment Wizard saves the user state migration data of the existing target computer to the location you specify. Later in the task sequence, the user state migration data is restored to the target computer.

To complete the Deployment Wizard

1. Start the Deployment Wizard by connecting to the appropriate deployment share (for example, *server_name*\Distribution$\Scripts) and typing **cscript litetouch.vbs**.

   The Deployment Wizard starts.

2. Complete the Deployment Wizard as described in Running the Deployment Wizard, ensuring that you specifically follow the configuration settings on the wizard pages listed in Table 17 and select the values on the other wizard pages appropriate to your organization.

Table 17. Information for Completing the Deployment Wizard for the Refresh Computer Deployment Scenario Using LTI

Expand table

On this wizard page	Do this
Select a task sequence to execute on this computer	Select the task sequence you created for the target computer deployment for the Refresh Computer deployment scenario.
Choose a migration type.	Select **Refresh this computer**.
Specify where to save your data and settings.	Select one of the following options based on the requirements of your organization: **- Automatically determine the location** **- Specify a location**

The wizard starts, and the operating system deployment starts. At the end of the process, the **Deployment Summary** dialog box appears.

3. In the **Deployment Summary** dialog box, select **Details**.

   If any errors or warnings occur, review them, and record any diagnostic information. For more information about the errors or warnings, see the MDT document *Troubleshooting Reference*.

4. In the **Deployment Summary** dialog box, select **Finish**.

   The target computer is now deployed.

Deploy Captured Images to Target Computers in a Replace Computer Deployment Scenario Using LTI

The Replace Computer deployment scenario requires that you run the Deployment Wizard twice. Run the wizard the first time to capture the user state migration data from the existing target computer (old computer). Then, run it again to deploy the captured image of the reference computer to the new target computer (new computer) and restore the user state saved earlier in the process.

Ensure that the user state migration data is stored in a consistent and secure location so that the data can be readily restore later in the LTI process.

To deploy captured images of the reference computer

1. Save the user state migration data from the existing target computer as described in Save the User State Migration Data from the Existing Target Computer Using LTI.

2. Deploy the captured image of the reference computer to the new target computer as described in Deploy the Captured Image of the Reference Computer to the New Target Computer Using LTI.

Save the User State Migration Data from the Existing Target Computer Using LTI

Start the Deployment Wizard on the existing operating system on the target computer to start the Standard Client Replace Task Sequence template created earlier in the process. The Deployment Wizard saves the user state migration data of the existing target computer to the location you specify.

To complete the Deployment Wizard

1. Start the Deployment Wizard by connecting to the appropriate deployment share (for example, *server_name*\Distribution$\Scripts) and typing **cscript litetouch.vbs**.

   The Deployment Wizard starts.

2. Complete the Deployment Wizard as described in Running the Deployment Wizard, ensuring that you specifically follow the configuration settings on the wizard pages listed in Table 18 and select the values on the other wizard pages that are appropriate to your organization.

Table 18. Information for Completing the Deployment Wizard for the Replace Computer Deployment Scenario for Saving User State Migration Data Using LTI

Expand table

On this wizard page	Do this
Select a task sequence to execute on this computer	Select the task sequence you created for the target computer in the Replace Computer deployment scenario to save the user state migration data.

On this wizard page	Do this
Specify where to save your data and settings.	In the **Location** box, type the fully qualified path to the location in which you want to store the user state migration data. This location must be accessible to the new target computer.
Specify where to save a complete computer backup	Select **Specify a location**, and then type the fully qualified path to the location in which you want to store the backup. This backup is for use in restoring user state migration data that might have been missed, not as a rollback method for the target computer. If you want to have rollback capability for the target computer, perform a full backup using your organization's backup software.

The wizard starts, and the operating system deployment starts. At the end of the process, the **Deployment Summary** dialog box appears.

3. In the **Deployment Summary** dialog box, select **Details**.

   If any errors or warnings occur, review them, and record any diagnostic information. For more information about the errors or warnings, see the MDT document Troubleshooting Reference.

4. In the **Deployment Summary** dialog box, select **Finish**.

   The user state migration data of the existing target computer is saved.

Deploy the Captured Image of the Reference Computer to the New Target Computer Using LTI

Start the target computer with the LTI bootable media you created earlier in the process or from Windows Deployment Services. The LTI bootable media starts Windows PE on the target computer and initiates LTI deployment. At the end of the process, the captured image of the reference computer is deployed to the target computer.

To deploy the captured image of the reference computer

1. Start the reference computer with the LTI bootable media you created earlier in the process or from Windows Deployment Services.

   Windows PE starts, and then the Deployment Wizard starts.

2. Complete the Deployment Wizard as described in Running the Deployment Wizard, ensuring that you specifically follow the configuration settings on the

wizard pages listed in Table 19 and select the values on the other wizard pages that are appropriate to your organization.

Table 19. Information for Completing the Deployment Wizard for the Replace Computer Deployment Scenario for Deploying the Captured Image

Expand table

On this wizard page	Do this
Select a task sequence to execute on this computer	Select the task sequence you created for the target computer for the Replace Computer deployment scenario.
Specify whether to restore user data	Select the **Specify a location** option and type the location of the saved user state migration data in the **Location** box.

The wizard starts, and the operating system deployment starts. At the end of the process, the **Deployment Summary** dialog box appears.

3. In the **Deployment Summary** dialog box, select **Details**.

   If any errors or warnings occur, review them, and record any diagnostic information. For more information about the errors or warnings, see the MDT document *Troubleshooting Reference*.

4. In the **Deployment Summary** dialog box, select **Finish**.

Managing LTI Deployments in the Deployment Workbench

Deployment shares are the repository for all the deployment files used in LTI deployment. You can store a deployment share on a local drive, in a network shared folder, or in a stand-alone distributed file system (DFS); it does not have to reside on any specific computer. Deployment shares contain operating systems, applications, operating system packages, and device drivers.

Manage LTI deployments in the Deployment Workbench by:

- Managing shares as described in Managing Deployment Shares in the Deployment Workbench

- Configuring the operating systems as described in Configuring Operating Systems in the Deployment Workbench
- Configuring the applications to be deployed as described in Configuring Applications in the Deployment Workbench
- Configuring packages as described in Configuring Packages in the Deployment Workbench
- Configuring device drivers as described in Configuring Device Drivers in the Deployment Workbench
- Configuring task sequences as described in Configuring Task Sequences in the Deployment Workbench
- Performing common administrative tasks as described in Performing Common Management Tasks in the Deployment Workbench
- Performing advanced configuration tasks as described in Performing Advanced Configuration Tasks in the Deployment Workbench
- Performing common administrative tasks as described in Configuring LTI Task Sequence Steps in the Deployment Workbench

  In addtion to managing LTI deployments in the Deployment Workbench, you can manage LTI deployments using the MDT Windows PowerShell cmdlets. For more information on managing LTI deployments using the MDT Windows PowerShell cmdlets, see:

- The section, "MDT Windows PowerShell Cmdlets", in the MDT document, *Toolkit Reference*
- The section, "Managing MDT Using Windows PowerShell", in the MDT document, *Microsoft Deployment Toolkit Samples Guide*

Managing Deployment Shares in the Deployment Workbench

MDT uses the Deployment Workbench to manage the deployment shares in your organization. You configure the deployment shares by:

- Creating a new deployment share as described in Create a New Deployment Share in the Deployment Workbench

- Opening an existing deployment share as described in Open an Existing Deployment Share in the Deployment Workbench
- Upgrading an existing deployment share to MDT as described in Upgrade an Existing Deployment Share in the Deployment Workbench
- Configuring a deployment share as described in Configure a Deployment Share in the Deployment Workbench
- Copying a deployment share as described in Copy a Deployment Share
- Closing a deployment share as described in Close a Deployment Share in the Deployment Workbench
- Updating a deployment share as described in Update a Deployment Share in the Deployment Workbench
- Creating bootable devices to start LTI from the MDT boot images as described in Create Bootable Devices from MDT Boot Images

  In addtion to managing deployment shares in the Deployment Workbench, you can manage deployment shares using the MDT Windows PowerShell cmdlets. For more information on managing deployment shares using the MDT Windows PowerShell cmdlets, see the following sections beneath the section, "MDT Windows PowerShell Cmdlets", in the MDT document Toolkit Reference:

- **Add-MDTPersistentDrive**
- **Get-MDTDeploymentShareStatistics**
- **Get-MDTPersistentDrive**
- **Remove-MDTPersistentDrive**
- **Restore-MDTPersistentDrive**
- **Update-MDTDeploymentShare**

Create a New Deployment Share in the Deployment Workbench

To create a new deployment share, perform the following steps:

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.

2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares.

3. In the Actions pane, select **New Deployment Share**.

   The New Deployment Share Wizard starts.

4. Complete the New Deployment Share Wizard using the information in Table 20.

Table 20. Information for Completing the New Deployment Share Wizard

Expand table

On this wizard page	Do this
Path	In **Deployment share path**, type **path** (where *path* is the fully qualified path to an existing folder on a local drive or a network shared folder created earlier in the deployment process), and then select **Next**. Alternatively, select **Browse** to find the existing folder on a local drive or network shared folder.
Share	This page is displayed only if you entered a path to a folder on a local drive on the Path wizard page. - In **Share name**, type **share_name** (where *share_name* is the share name for the folder on a local drive specified on the Path wizard page). - Note the fully qualified to UNC path to the share being created listed immediately below the **Share name** box, and then select **Next**. The wizard grants the local group Everyone Full Control access at the share level. Based on your security requirements, you may want to restrict the security of the share.
Descriptive Name	In **Deployment share description**, type **description** (where *description* is a descriptive name that for the deployment share specified on previous wizard pages), and then select **Next**.
Allow Image Capture	Select or clear the **Ask if an image should be captured** check box based on requirements, and then select **Next**. This check box configures the Deployment Wizard to allow the user to optionally capture an image of the target computer, which is usually the reference computer. If the check box is:

On this wizard page	Do this
	- Selected, the path for storing the image and the image name can be configured in the Deployment Wizard - Cleared, an image is not capture or the image-capture information must be set in the MDT configuration file or database By default, this check box is selected.
Allow Admin Password	Select or clear the **Ask user to set the local Administrator Password** check box based on requirements, and then select **Next**. This check box configures the Deployment Wizard to allow the user to provide the password for the local Administrator account during the deployment process. If the check box is: - Selected, the password can be configured in the Deployment Wizard - Cleared, the password must be set in the MDT configuration file or database By default, this check box is cleared.
Allow Product Key	Select or clear the **Ask user for a product key** check box based on your requirements, and then select **Next**. This check box configures the Deployment Wizard to allow the user to provide a product key for the target operating system during the deployment process. If the check box is: - Selected, the product key can be configured in the Deployment Wizard - Cleared, the product key must be set in the MDT configuration file or database By default, this check box is cleared.
Summary	Review the information in the **Details** box, and then select **Next**.
Confirmation	Select **Save Output** to save the output of the wizard to a file, or select **View Script** to view the Windows PowerShell scripts used to perform the wizard tasks. Select **Close**.

Upon completion, the new deployment share is created in the target folder you selected in the wizard and appears in the Deployment Workbench.

Open an Existing Deployment Share in the Deployment Workbench

The Deployment Workbench can open an existing deployment share using the Open Deployment Share Wizard. Start the Open Deployment Share Wizard by:

- Right-clicking the Deployment Shares node, and then selecting**Open Deployment Share**
- Selecting the Deployment Shares node, and then, from the Action menu, selecting**Open Deployment Share**
- Selecting the Deployment Shares node, and then, in the Actions pane, selecting **Open Deployment Share**

To open an existing deployment share not already listed in the Deployment Workbench

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares.
3. In the Actions pane, select **Open Deployment Share**.

   The Open Deployment Share Wizard starts.
4. Complete the Open Deployment Share Wizard using the information in Table 21.

Table 21. Information for Completing the Open Deployment Share Wizard

Expand table

On this wizard page	Do this
Path	a. In **Deployment share path**, type **share_path** (where *share_path* is the fully qualified path to the existing deployment share). You can alternatively select Browse to find the local or network shared folder. b. Select the **Upgrade the content of the deployment share (if required)**check box. If the check box is:

On this wizard page	Do this
	- Selected, the Open Deployment Share Wizard upgrades the deployment share - Cleared, the Open Deployment Share Wizard will not upgrade the deployment share c. Select **Next**.
Summary	Review the information in the **Details** box, and then select **Next**.
Confirmation	You can select Save Output to save the output of the wizard to a file. You can also select View Script to view the Windows PowerShell scripts used to perform the wizard tasks. Select **Finish**.

Upgrade an Existing Deployment Share in the Deployment Workbench

MDT can upgrade an existing deployment share by:

- Opening an existing deployment share that is not already listed in the Deployment Workbench as described in Upgrade Deployment Shares Not Already Listed in the Deployment Workbench
- Upgrading an existing deployment share that is listed in the Deployment Workbench as described in Upgrade Deployment Shares Already Listed in the Deployment Workbench

Upgrade Deployment Shares Not Already Listed in the Deployment Workbench

Upgrade deployment shares not listed in the Deployment Workbench using the Open Deployment Share Wizard. Start the wizard by:

- Right-clicking the deployment share, and then selecting **Open Deployment Share**
- Selecting the deployment share, and then, from the **Actions** menu, selecting **Open Deployment Share**
- Selecting the deployment share, and then, in the Actions pane, selecting **Open Deployment Share**

To upgrade deployment shares that are not already listed in the Deployment Workbench, perform the following steps:

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares.
3. In the Actions pane, select **Open Deployment Share**.

   The Open Deployment Share Wizard starts.
4. Complete the Open Deployment Share Wizard using the information in Table 22.

Table 22. Information for Completing the Open Deployment Share Wizard

Expand table

On this wizard page	Do this
Path	- In **Deployment share path**, type **share_path** (where *share_path* is the fully qualified path to the existing deployment share). Alternatively, select Browse to find the local or network shared folder. - Select the **Upgrade the content of the deployment share (if required)** check box, and then select **Next**.
Summary	Review the information in the **Details** box, and then select **Next**.
Confirmation	Select **Save Output** to save the output of the wizard to a file, or select **View Script** to view the Windows PowerShell scripts used to perform the wizard tasks. Select Finish.

After the wizard finishes, the existing deployment share is upgraded (if required), and the **Upgrade the content of the deployment share (if required)** check box is selected. The deployment share is added to the details pane in the Deployment Workbench.

In addition to upgrading existing deployment shares, any existing installations of previous versions of MDT must be upgraded to MDT. For more information on

upgrading any previous installations to MDT, see Upgrading to MDT.

Upgrade Deployment Shares Already Listed in the Deployment Workbench

Upgrade existing deployment shares already listed in the Deployment Workbench using the Upgrade Deployment Share Wizard. Start the wizard by:

- Right-clicking the deployment share, and then selecting **Upgrade Deployment Share**
- Selecting the deployment share, and then, from the **Actions** menu, selecting **Upgrade Deployment Share**
- Selecting the deployment share, and then, in the Actions pane, selecting **Upgrade Deployment Share**

  To upgrade existing deployment shares already listed in the Deployment Workbench, perform the following steps:

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares.
3. In the Actions pane, select **Upgrade Deployment Share**.

   The Upgrade Deployment Share Wizard starts.
4. Complete the Upgrade Deployment Share Wizard using the information in Table 23.

Table 23. Information for Completing the Upgrade Deployment Share Wizard

Expand table

On this wizard page	Do this
Summary	Review the information in the **Details** box, and then select **Next**.
Confirmation	Select **Save Output** to save the output of the wizard to a file, or select **View Script** to view the Windows PowerShell scripts used to perform the

On this wizard page	Do this
	wizard tasks. Select **Finish**.

After the wizard finishes, the existing deployment share is upgraded and now can be accessed in the Deployment Workbench.

Configure a Deployment Share in the Deployment Workbench

You can view the properties of deployment shares beneath the Deployment Shares node in the Deployment Workbench by using the **Properties** actions as described in View Item Properties in the Deployment Workbench.

Configure an application in the Deployment Workbench by performing the following tasks in the **Application Properties** dialog box:

- Configure the settings on the **General** tab as described in Configure the Deployment Share Properties General Tab.
- Configure the settings on the **Rules** tab as described in Configure the Deployment Share Properties Rules Tab.
- Configure the settings on the **Windows PE x86 Settings** tab as described in Configure the Deployment Share Properties Windows PE x86 Settings Tab.
- Configure the settings on the **Windows PE x86 Components** tab as described in Configure the Deployment Share Properties Windows PE x86 Components Tab.
- Configure the settings on the **Windows PE x64 Settings** tab as described in Configure the Deployment Share Properties Windows PE x64 Settings Tab.
- Configure the settings on the **Windows PE x64 Components** tab as described in Configure the Deployment Share Properties Windows PE x64 Components Tab.

Configure the Deployment Share Properties General Tab

The deployment share properties stored on the **General** tab are mostly configured when you run the New Deployment Share Wizard. You can update the deployment share properties on the **General** tab through the **deployment_share Properties** dialog box (where *deployment_share* is the name of the deployment share in the Deployment Workbench).

To configure the General tab

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares.
3. In the details pane, select ***deployment_share*** (where *deployment_share* is the name of the deployment share).
4. In the Actions pane, select **Properties**.

   The ***deployment_share* Properties*** dialog box opens (where *deployment_share* is the name of the deployment share you want to configure).
5. On the **General** tab, configure the settings listed in Table 24 based on the requirements of your organization, and then select **OK**.

Table 24. Configuration Settings on the General Tab of Deployment Share Properties

Expand table

Setting	Description
Description	Contains the name of the deployment share displayed in the Deployment Workbench. The default value is **MDT Deployment Share**.
Comments	Provides information about the deployment share.
Network (UNC) path	Text box that contains fully qualified UNC path to the deployment share. This value is used only to enable multicast and is required if you want to do so by selecting the **Enable multicast for this deployment share** check box. If the deployment share was created from an existing network shared folder, this value is displayed in the details pane of the Deployment Workbench.
Local path	Contains the fully qualified path to the local folder in which the deployment share was created. This value is used only to enable multicast and is required if you want to do so by selecting the **Enable multicast for this deployment share** check box. If you created the deployment share from: - A local path, then this text box contains the local path used in the

Setting	Description
	creation process - An existing network shared folder, then this text box is empty If you created the deployment share from a local path, the local path rather than the UNC path is displayed in the details pane of the Deployment Workbench.
Platforms supported: x86	Select to configure the Update Deployment Share Wizard to create WIM files and bootable media for 32-bit target computers.
Platforms supported: x64	Select to configure the Update Deployment Share Wizard to create WIM files and bootable media for 64-bit target computers.
Enable multicast for this deployment share	Select to configure Windows Deployment Services to enable multicast deployment of images generated in this deployment share. If you select this check box, you must provide values for the **Network (UNC) path** and **Local path** boxes. For more information about enabling multicast-based deployments, see Enable Windows Deployment Services Multicast Deployment for LTI Deployments. The multicast features of MDT are available using Windows Deployment Services in Windows Server.

The deployment share configuration settings are saved, and the deployment share appears in the details pane of the Deployment Workbench.

Configure the Deployment Share Properties Rules Tab

The deployment share properties stored on the **Rules** tab are mostly configured when you run the New Deployment Share Wizard. These settings reside in CustomSettings.ini, which is in the deployment share's Control folder. For more information about the settings that you can configure on this tab, see the MDT document *Toolkit Reference*.

To configure the Rules tab

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares.
3. In the details pane, select ***deployment_share*** (where *deployment_share* is the name of the deployment share).

4. In the Actions pane, select **Properties**.

   The ***deployment_share*** **Properties** dialog box opens (where *deployment_share* is the name of the deployment share you want to configure).

5. On the **Rules** tab, configure the settings listed in Table 25 based on the requirements of your organization, and then select **OK**.

Table 25. Configuration Settings on the Rules Tab of Deployment Share Properties

Expand table

Setting	Description
CustomSettings.ini	Contains the current configuration of the CustomSetting.ini file for the deployment share.
Edit Bootstrap.ini	Select to modify the contents of the Bootstrap.ini file that the Deployment Workbench generates.

The deployment share configuration settings are saved, and the deployment share appears in the details pane of the Deployment Workbench.

Configure the Deployment Share Properties Windows PE x86 Settings Tab

The deployment share properties stored on the **Windows PE x86 Settings** tab are mostly configured when you run the New Deployment Share Wizard.

To configure the Windows PE x86 Settings tab

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.

2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares.

3. In the details pane, select ***deployment_share*** (where *deployment_share* is the name of the deployment share).

4. In the Actions pane, select **Properties**.

The **deployment_shareProperties* dialog box opens (where *deployment_share* is the name of the deployment share you want to configure).

5. On the **Windows PE x86 Settings** tab, configure the settings listed in REF _Ref304458789 \h Table 26 based on the requirements of your organization, and then select **OK**.

Table 26. Configuration Settings on the Windows PE x86 Settings Tab of Deployment Share Properties

Expand table

Setting	Description
Generate a Lite Touch Windows PE WIM file	Select to configure the Update Deployment Share Wizard to create a Windows PE WIM file that includes the LTI deployment scripts. If the check box is: - Selected, the Update Deployment Share Wizard creates the LiteTouchPE_x86.wim file with the image description specified in the **Image description** text box - Cleared, the Update Deployment Share Wizard will not create the WIM file
Image description	Contains the image description for the Windows PE WIM file that the Update Deployment Share Wizard creates. The default value is **Lite Touch Windows PE (x86)**.
Generate a Lite Touch bootable ISO image	Select to configure the Update Deployment Share Wizard to create a bootable Windows PE ISO file that includes the LTI deployment scripts. If the check box is: - Selected, the Update Deployment Share Wizard creates the ISO file with the name specified in the **ISO file name** text box - Cleared, the Update Deployment Share Wizard will not create the ISO file
ISO file name	Contains the file name for the Windows PE ISO file that the Update Deployment Share Wizard creates. The default value for this text box is **LiteTouchPE_x86.iso**. This text box is enabled only if you select **Generate a Lite Touch bootable ISO image**.
Generate a generic Windows	Select to configure the Update Deployment Share Wizard to create a Windows PE WIM file that does not include the LTI deployment scripts.

Setting	Description
PE WIM file	If the check box is: - Selected, the Update Deployment Share Wizard creates the GenericPE_x86.wim file with the image description specified in the Image description text box - Cleared, the Update Deployment Share Wizard will not create the WIM file
Image description	Contains the image description for the generic Windows PE WIM file that the Update Deployment Share Wizard creates. The default value for this text box is **Generic Windows PE (x86)**. This text box is enabled only if you select **Generate a generic Windows PE WIM file**.
Generate a generic bootable ISO image	Select to configure the Update Deployment Share Wizard to create a bootable Windows PE ISO file that does not include the LTI deployment scripts. If the check box is: - Selected, the Update Deployment Share Wizard creates the Generic_x86.iso file with the image description specified in the **ISO file name** text box - Cleared, the Update Deployment Share Wizard will not create the ISO file This box is enabled only if you select **Generate a generic Windows PE WIM file**.
ISO file name	Contains the file name for the generic Windows PE ISO file that the Update Deployment Share Wizard creates. The default value for this text box is **Generic_x86.iso**. This text box is enabled only if you select **Generate a generic bootable ISO image**.
Custom background bitmap file	Contains the fully qualified path to the BMP file that is to be used as the custom background bitmap. The default value for this text box is **%INSTALLDIR%\Samples\Background.bmp**.
Extra directories to add	Contains the fully qualified path to a folder structure to be included in the Windows PE images. The default value for the text box is empty.
Scratch space size	Configures the size of the scratch space in megabytes; you can select a value of **32**, **64**, **128**, **256**, or **512** MB. The default value is **32**.

The deployment share configuration settings are saved, and the deployment share appears in the details pane of the Deployment Workbench.

Configure the Deployment Share Properties Windows PE x86 Components Tab

The deployment share properties stored on the **Windows PE x86 Components** tab are mostly configured when you run the New Deployment Share Wizard.

To configure the Windows PE x86 Components tab

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares.
3. In the details pane, select ***deployment_share*** (where *deployment_share* is the name of the deployment share).
4. In the Actions pane, select **Properties**.

   The ***deployment_share*** Properties** dialog box opens (where *deployment_share* is the name of the deployment share you want to configure).
5. On the **Windows PE x86 Components** tab, configure the settings listed in Table 29 based on the requirements of your organization, and then select **OK**.

Table 27. Configuration Settings on the Windows PE x86 Components Tab of Deployment Share Properties

Expand table

Setting	Description
Selection profile	Use to select the device drivers and packages to be included in the Windows PE images based on the selection profile you choose. The default value is **All Drivers and Packages**. For more information about selection profiles, see Manage Selection Profiles.
Include all drivers from the selected driver group	Select to configure the Update Deployment Share Wizard to include all the device drivers in the selection profile specified in the **Selection profile** box.

Setting	Description
Include only drivers of the following types	Select to configure the Update Deployment Share Wizard to include only the device drivers in the chosen selection profile that are specified in the following check boxes: **- Include all network drivers in the selected group** **- Include all video drivers in the selected group** **- Include all mass storage drivers in the selected group** **- Include all system-class drivers in the selected group**
Include all network drivers in the selected group	Select to configure the Update Deployment Share Wizard to include all network drivers in the chosen selection profile. If the check box is: - Selected, the Update Deployment Share Wizard includes all network drivers in the selection profile specified in the **Selection profile** box in the Windows PE images - Cleared, the Update Deployment Share Wizard will not include all network drivers in the selection profile specified in the **Selection profile** box in the Windows PE images This check box is enabled only if you select **Include only drivers of the following types**.
Include all video drivers in the selected group	Select to configure the Update Deployment Share Wizard to include all video drivers in the chosen selection profile. If the check box is: - Selected, the Update Deployment Share Wizard includes all video drivers in the selection profile specified in the **Selection profile** box in the Windows PE images - Cleared, the Update Deployment Share Wizard will not include all video drivers in the selection profile specified in the **Selection profile** list box in the Windows PE images This check box is enabled only if you select **Include only drivers of the following types**.
Include all mass storage drivers in the selected group	Select to configure the Update Deployment Share Wizard to include all mass storage drivers in chosen the selection profile. If the check box is: - Selected, the Update Deployment Share Wizard includes all mass storage drivers in the selection profile specified in the **Selection profile** box in the Windows PE images

Setting	Description
	- Cleared, the Update Deployment Share Wizard will not include all mass storage drivers in the selection profile specified in the **Selection profile** box in the Windows PE images This check box is enabled only if you select **Include only drivers of the following types**.
Include all system-class drivers in the selected group	Select to configure the Update Deployment Share Wizard to include all system-class drivers in the chosen selection profile. If the check box is: - Selected, the Update Deployment Share Wizard includes all system-class drivers in the selection profile specified in the **Selection profile** box in the Windows PE images - Cleared, the Update Deployment Share Wizard will not include all system-class drivers in the selection profile specified in the **Selection profile** box in the Windows PE images This check box is enabled only if you select **Include only drivers of the following types**.
ADO	Select to add optional ADO components to the Windows PE images. These components are necessary for accessing Microsoft SQL Server® databases, such as the MDT DB. If the check box is: - Selected, the ADO components are added to the Windows PE images - Cleared, the ADO components are not added to the Windows PE images
Optional Fonts	Select to configure the Update Deployment Share Wizard to include the following fonts: - Chinese (ZH-CN) - Chinese (ZH-HK) - Chinese (ZH-TW) - Japanese (JA-JP) - Korean (KO-KR) Add these fonts when performing an LTI deployment of Windows Server images and the Setup files are Japanese, Korean, or Chinese. If the check box for a corresponding font is:

Setting	Description
	- Selected, the Update Deployment Share Wizard includes the font in the Windows PE images - Cleared, the Update Deployment Share Wizard will not include the font in the Windows PE images Adding fonts to Windows PE boot images increases the size of the images. Add fonts only if necessary.

The deployment share configuration settings are saved, and the deployment share appears in the details pane of the Deployment Workbench.

Configure the Deployment Share Properties Windows PE x64 Settings Tab

The deployment share properties stored on the Windows PE x64 Settings tab are mostly configured when you run the New Deployment Share Wizard.

To configure the Windows PE x64 Settings tab

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares.
3. In the details pane, select **deployment_share** (where *deployment_share* is the name of the deployment share).
4. In the Actions pane, select **Properties**.

   The ***deployment_share*** **Properties** dialog box opens (where *deployment_share* is the name of the deployment share you want to configure).
5. On the **Windows PE x64 Settings** tab, configure the settings listed in Table 28 based on the requirements of your organization, and then select **OK**.

Table 28. Configuration Settings on the Windows PE x64 Settings Tab of Deployment Share Properties

Expand table

Setting	Description
Generate a Lite Touch Windows PE WIM file	Select to configure the Update Deployment Share Wizard to create a Windows PE WIM file that includes the LTI deployment scripts. If the check box is: - Selected, the Update Deployment Share Wizard creates the LiteTouchPE_x86.wim file with the image description specified in the **Image description** box - Cleared, the Update Deployment Share Wizard will not create the WIM file
Image description	Contains the image description for the Windows PE WIM file that the Update Deployment Share Wizard creates. The default value is **Lite Touch Windows PE (x64)**.
Generate a Lite Touch bootable ISO image	Select to configure the Update Deployment Share Wizard to create a bootable Windows PE ISO file that includes the LTI deployment scripts. If the check box is: - Selected, the Update Deployment Share Wizard creates the ISO file with the name specified in the **ISO file name** box - Cleared, the Update Deployment Share Wizard will not create the ISO file
ISO file name	Contains the file name for the Windows PE ISO file that the Update Deployment Share Wizard creates. The default value for this text box is **LiteTouchPE_x64.iso**. This box is enabled only if you select **Generate a Lite Touch bootable ISO image**.
Generate a generic Windows PE WIM file	Select to configure the Update Deployment Share Wizard to create a Windows PE WIM file that does not include the LTI deployment scripts. If the check box is: - Selected, the Update Deployment Share Wizard creates the GenericPE_x64.wim file with the image description specified in the **Image description** box - Cleared, the Update Deployment Share Wizard will not create the WIM file
Image description	Contains the image description for the generic Windows PE WIM file that the Update Deployment Share Wizard creates. The default value for this text box is **Generic Windows PE (x64)**.

Setting	Description
	This box is enabled only if you select **Generate a generic bootable ISO Windows PE WIM file**.
Generate a generic bootable ISO image	Select to configure the Update Deployment Share Wizard to create a bootable Windows PE ISO file that does not include the LTI deployments scripts. If the check box is: - Selected, the Update Deployment Share Wizard creates the Generic_x64.iso file with the image description specified in the **ISO file name** box - Cleared, the Update Deployment Share Wizard will not create the ISO file This box is enabled only if you select **Generate a generic Windows PE WIM file**.
ISO file name	Contains the file name for the generic Windows PE ISO file that the Update Deployment Share Wizard creates. The default value for this text box is **Generic_x64.iso**. This text box is enabled only if the **Generate a generic bootable ISO image** check box is selected.
Custom background bitmap file	Contains the fully qualified path to the BMP file that is to be used as the custom background bitmap. The default value for this text box is **%INSTALLDIR%\Samples\Background.bmp**.
Custom background bitmap file	Contains the fully qualified path to a folder structure to be included in the Windows PE images. The default value for the text box is empty.
Custom background bitmap file	Configures the size of the scratch space in megabytes; you can select a value of **32**, **64**, **128**, **256**, or **512** MB. The default value is **32**.

The deployment share configuration settings are saved, and the deployment share appears in the details pane of the Deployment Workbench.

Configure the Deployment Share Properties Windows PE x64 Components Tab

The deployment share properties stored on the **Windows PE x64 Components** tab are mostly configured when you run the New Deployment Share Wizard.

To configure the Windows PE x64 Components tab

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares.
3. In the details pane, select ***deployment_share*** (where *deployment_share* is the name of the deployment share).
4. In the Actions pane, select **Properties**.

   The ***deployment_share*** **Properties** dialog box opens (where *deployment_share* is the name of the deployment share you want to configure).
5. On the **Windows PE x86 Components** tab, configure the settings listed in Table 29 based on the requirements of your organization, and then select **OK**.

Table 29. Configuration Settings on the Windows PE x64 Components Tab of Deployment Share Properties

Expand table

Setting	Description
Selection profile	Selects the device drivers and packages to be included in the Windows PE images based on the selection profile you choose. The default value is **All Drivers and Packages**. For more information about selection profiles, see Manage Selection Profiles.
Include all drivers from the selected driver group	Select to configure the Update Deployment Share Wizard to include all the device drivers in the selection profile specified in the **Selection profile** box.
Include only drivers of the following types	Select to configure the Update Deployment Share Wizard to include only the device drivers in the chosen selection profile that are specified in the following check boxes: **- Include all network drivers in the selected group** **- Include all video drivers in the selected group** **- Include all mass storage drivers in the selected group** **- Include all system-class drivers in the selected group**
Include all network drivers in the	Select to configure the Update Deployment Share Wizard to include all network drivers in the chosen selection profile. If the check box is:

Setting	Description
selected group	- Selected, the Update Deployment Share Wizard includes all network drivers in the selection profile specified in the **Selection profile** box in the Windows PE images - Cleared, the Update Deployment Share Wizard will not include all network drivers in the selection profile specified in the **Selection profile** box in the Windows PE images This check box is enabled only if you select **Include only drivers of the following types**.
Include all video drivers in the selected group	Select to configure the Update Deployment Share Wizard to include all video drivers in the chosen selection profile. If the check box is: - Selected, the Update Deployment Share Wizard includes all video drivers in the selection profile specified in the **Selection profile** box in the Windows PE images - Cleared, the Update Deployment Share Wizard will not include all video drivers in the selection profile specified in the **Selection profile** box in the Windows PE images This check box is enabled only if you select **include only drivers of the following types**.
Include all mass storage drivers in the selected group	Select to configure the Update Deployment Share Wizard to include all mass storage drivers in the chosen selection profile. If the check box is: - Selected, the Update Deployment Share Wizard includes all mass storage drivers in the selection profile specified in the **Selection profile** box in the Windows PE images - Cleared, the Update Deployment Share Wizard will not include all mass storage drivers in the selection profile specified in the **Selection profile** box in the Windows PE images This check box is enabled only if you select **include only drivers of the following types**.
Include all system-class drivers in the selected group	Select to configure the Update Deployment Share Wizard to include all system-class drivers in chosen the selection profile. If the check box is: - Selected, the Update Deployment Share Wizard includes all system-class drivers in the selection profile specified in the **Selection profile** box in the Windows PE images

Setting	Description
	- Cleared, the Update Deployment Share Wizard will not include all system-class drivers in the selection profile specified in the **Selection profile** box in the Windows PE images This check box is enabled only if you select **Include only drivers of the following types**.
ADO	Select to add the optional ADO components to the Windows PE images. These components are necessary for accessing SQL Server databases, such as the MDT DB. If this check box is: - Selected, the ADO components are added to the Windows PE images - Cleared, the ADO components are not added to the Windows PE images
Optional Fonts	Use to configure the Update Deployment Share Wizard to include the following fonts: - Chinese (ZH-CN) - Chinese (ZH-HK) - Chinese (ZH-TW) - Japanese (JA-JP) - Korean (KO-KR) Add these fonts when performing an LTI deployment of Windows Server images when the Setup files are Japanese, Korean, or Chinese. If the check box for a corresponding font is: - Selected, the Update Deployment Share Wizard includes the font in the Windows PE images - Cleared, the Update Deployment Share Wizard will not include the font in the Windows PE images Adding fonts to Windows PE boot images increases the size of the images. Add fonts only if necessary.

The deployment share configuration settings are saved, and the deployment share appears in the details pane of the Deployment Workbench.

Copy a Deployment Share

Deployment shares are local or network shared folders. You can make a copy of a deployment share using any file-copy process, such as in Windows Explorer. When copying a deployment share to another computer, ensure that you share the folder with the appropriate permissions.

After you copy the deployment share, open it in the Deployment Workbench. For more information about opening deployment shares, see Open an Existing Deployment Share in the Deployment Workbench.

Close a Deployment Share in the Deployment Workbench

> **Note**
>
> Closing a deployment share does not remove the local or network shared folder or delete the contents of the local or network shared folder: It only removes the deployment share from the list of deployment shares in the Deployment Shares node in the Deployment Workbench.

Close existing deployment shares in the Deployment Workbench using the **Close Deployment Share** action. Start the **Close Deployment Share** action by performing one of the following tasks:

- Right-click the deployment share, and then select **Close Deployment Share**.
- Select the deployment share, and then, from the **Action** menu, select **Close Deployment Share**.
- Select the deployment share, and then, in the Actions pane, select **Close Deployment Share**.

Update a Deployment Share in the Deployment Workbench

Updating a deployment share creates the Windows PE boot images (WIM and ISO files) necessary to start LTI deployment.

To update a deployment share in the Deployment Workbench

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.

2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares.

3. In the details pane, select ***deployment_share***, and then in the Actions pane, select **Update Deployment Share** (where *deployment_share* is the name of the deployment share you want to update).

   The Update Deployment Share Wizard starts.

4. Complete the Update Deployment Share Wizard using the information in Table 30. Accept the default values unless otherwise specified.

Table 30. Information for Completing the Update Deployment Share Wizard

Expand table

On this wizard page	Do this
Options	a. Select one of the following options based on your requirements: - **Optimize the boot image updating process**. This option configures the Update Deployment Share Wizard to update existing versions of the image files. Select this option when you want to reduce the amount of time required to update the boot images. Typically, this process takes less time than the **Completely regenerate the boot images** option. You can optionally select the **Compress the boot image contents to recover space used by removed or modified content** check box to reduce the size of the boot images. Over time, the process of adding and removing content (such as drivers, components, and packages) can increase the size of the Windows PE image. Selecting this option reduces the image size to the minimum for the current content. However, it may also increase the time required to generate the images. - **Completely regenerate the boot images**. This option configures the Update Deployment Share Wizard to create a new version of all the image files. Select this option when you want to force the creation of new images. This process can take longer than the **Optimize the boot image updating process** option. By default, the **Optimize the boot image updating process** option is selected and the **Compress the boot image contents to recover space used by removed or modified content** check box is cleared.

On this wizard page	Do this
	b. Select **Next**.
Summary	Review the information in the **Details** box, and then select **Next**.
Confirmation	You can select **Save Output** to save the output of the wizard to a file. You can also select **View Script** to view the Windows PowerShell scripts used to perform the wizard tasks. Select **Close**.

The Deployment Workbench starts updating the deployment share and creates the LiteTouchPE_x86.iso and LiteTouchPE_x86.wim files (for 32-bit target computers) or LiteTouchPE_x64.iso and LiteTouchPE_x64.wim files (for 64-bit target computers) in the *deployment_share*\Boot folder (where *deployment_share* is the network shared folder used as the deployment share) based on the configuration settings on the **Windows PE x86 Settings** and **Windows PE x64 Settings** tabs.

> ⓘ **Note**
>
> Optionally, create a bootable device, such as a UFD or USB hard disk, from the ISO file so that you can start the target computer from the device as described in Create Bootable Devices from MDT Boot Images.

Create Bootable Devices from MDT Boot Images

Starting destination computers using a bootable device (such as a UFD or a USB hard disk) is often quicker and easier than starting computers using Windows Deployment Services or CDs.

> ⓘ **Note**
>
> The target computer must support booting from the device to use this method of starting target computers.

To create a bootable UFD

1. On a computer running Windows 7 or later operating system, insert the UFD or USB hard disk.

2. Run **Diskpart.exe**, and type the command **list disk** to determine the disk number associated with the device.

3. Input the following commands, where N is the disk number identified in the previous step:

    - **select disk N**
    - **clean**
    - **create partition primary**
    - **select partition 1**
    - **active**
    - **format fs=fat32**
    - **assign**
    - **exit**

4. Copy the contents of LiteTouchPE_x86.iso (for 32-bit target computers) or LiteTouchPE_x64.iso (for 64-bit target computers) to the device by performing one of the following tasks:

    - Burn the ISO file to a CD, and then copy its contents to the device using the command:

        Windows Command Prompt

        ```
        xcopy <d>:\*.* <e>:\*.* /s /e /f
        ```

        Where *d* is the driver letter of the CD and *e* is the drive letter of the device.

    - Alternatively, mount the ISO file using a virtual CD program, and then copy its contents to the device using the command:

        Windows Command Prompt

        ```
        xcopy <d>:\*.* <e>:\*.* /s /e /f
        ```

        where *d* is the driver letter of the CD and *e* is the drive letter of the device.

Configuring Operating Systems in the Deployment Workbench

MDT uses the Deployment Workbench to manage the operating systems that you can deploy to the reference and target computers in your organization. Configure the operating systems in the Deployment Workbench by:

- Importing an operating system as described in Import an Operating System into the Deployment Workbench
- Viewing an operating system's properties as described in View Operating System Properties in the Deployment Workbench
- Copying an operating system as described in Copy an Operating System in the Deployment Workbench
- Moving an operating system as described in Move an Operating System in the Deployment Workbench
- Renaming an operating system as described in Rename an Operating System in the Deployment Workbench
- Deleting an operating system as described in Delete an Operating System from the Deployment Workbench
- Managing folders for operating systems as described in Manage Folders for Operating Systems in the Deployment Workbench

  In addtion to configuring operating systems in the Deployment Workbench, you can configure operating systems using the MDT Windows PowerShell cmdlets. For more information on configuring operating systems using the MDT Windows PowerShell cmdlets, see the following sections beneath the section, "MDT Windows PowerShell Cmdlets", in the MDT document *Toolkit Reference*:

- **Get-MDTOperatingSystemCatalog**
- **Import-MDTOperatingSystem**

Import an Operating System into the Deployment Workbench

You can import operating systems into the Deployment Workbench using the options listed in Table 31. You manage this import in the Import Operating System Wizard in the Deployment Workbench.

Table 31. Options for Importing Operating Systems into the Deployment Workbench

Expand table

Option	Select this option to import an operating system from
Full set of source files	Windows distribution media, such as a DVD, CD, or equivalent media source.
Custom image file	A WIM image that was previously captured for deployment, typically from a reference computer.
Windows Deployment Services images	Images that exist on computers running Windows Deployment Services.

> **Note**
>
> Always import operating systems from operating system sources that have the most recent updates. Doing so helps reduces the management effort and network traffic when applying the updates after the target operating system has been deployed.

Start the Import Operating System Wizard using one of the following methods:

- In the Deployment Workbench console tree, select the Operating System node or a folder beneath the Operating System node. Then, in the Actions pane, select **Import Operating System**.
- In the Deployment Workbench console tree, select the Operating System node or a folder beneath the Operating System node. Then, from the **Action** menu, select **Import Operating System**.
- In the Deployment Workbench console tree, select the Operating System node or a folder beneath the Operating System node. Then, select **Import Operating System**.

  You can also import operating systems into the Deployment Workbench from:

- Windows distribution media as described in Import an Operating System from Windows Distribution Media
- WIM images previously captured from reference computers as described in Import a Previously Captured Image of a Reference Computer

- Existing WIM images in Windows Deployment Services as described in Import an Operating System from an Existing Windows Deployment Services Image

Import an Operating System from Windows Distribution Media

MDT allows you to import operating systems into the Deployment Workbench from Windows distribution media, which includes product DVDs, CDs, or folders containing the distribution files. Import the operating system using the Import Operating System Wizard in the Deployment Workbench.

To import an operating system from Windows distribution media

1. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/Operating Systems (where *deployment_share* is the name of the deployment share to which the operating system will be added).

2. In the Actions pane, select **Import Operating System**.

   The Import Operating System Wizard starts.

3. Complete the Import Operating System Wizard using the information in Table 32.

Expand table

On this wizard page	Do this
OS Type	Select **Full set of source files**, and then select **Next**.
Source	In **Source directory**, type ***source_folder*** (where *source_folder* is the fully qualified path to the folder containing the operating system source files), and then select **Next**. Alternatively, select Browse to find the source folder.
Destination	In **Destination directory name**, type ***destination_folder*** (where *destination_folder* is the name of the folder in the deployment share that will contain the operating system source files), and then select **Next**.
Summary	Select **Next**.
Confirmation	Select **Save Output** to save the output of the wizard to a file, or select **View Script** to view the Windows PowerShell scripts used to perform the wizard tasks.

On this wizard page	Do this
	Select **Finish**.

The Import Operating System Wizard finishes. The operating system is added to the list of operating systems in the details pane of the Deployment Workbench.

Import a Previously Captured Image of a Reference Computer

MDT allows you to import previously captured images of reference computers or other custom images into the Deployment Workbench. Import the operating system using the Import Operating System Wizard in the Deployment Workbench.

To import an operating system from a previously captured image of a reference computer

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/Operating Systems (where *deployment_share* is the name of the deployment share to which the operating system will be added).
3. In the Actions pane, select **Import Operating System**.

   The Import Operating System Wizard starts.
4. Complete the Import Operating System Wizard using the information in Table 33.

Expand table

On this wizard page	Do this
OS Type	Select **Custom image file**, and then select **Next**.
Source	In **Source file**, type ***source_file*** (where *source_file* is the fully qualified path to the WIM image file containing the operating system source files), and then select **Next**. Alternatively, select **Browse** to find the source WIM image.

On this wizard page	Do this
Setup	Select one of the following options based on your requirements, and then select **Next**: - **Setup files are not needed**. Select this option when no Setup files are needed for an image. - **Copy Windows 7, Windows Server 2008 R2, or later setup files from the specified path**. Select this option to copy the Setup files from a folder containing the Windows setup files when those files are not available in another operating system in the Deployment Workbench. In **Setup source directory**, type or select **Browse** to find the folder containing the Setup files. The default option is **Setup and Sysprep files are not needed**.
Destination	In **Destination directory name**, type ***destination_folder*** (where *destination_folder* is the name of the folder in the deployment share that will contain the operating system source files), and then select **Next**.
Summary	Select **Next**.
Confirmation	Select **Save Output** to save the output of the wizard to a file, or select **View Script** to view the Windows PowerShell scripts used to perform the wizard tasks. Select **Finish**.

The Import Operating System Wizard finishes. The operating system is added to the list of operating systems in the details pane of the Deployment Workbench.

If you attempt to import a custom image that does not have an Edition ID, the wizard fails with an error similar to the following:

Console

```
Setup failed applying image F:\Deploy\Operating
Systems\W2K8R2RTM\W2K8R2RTM.wim, rc = 31
ZTI ERROR - Non-zero return code by LTIApply, rc = 31.
```

To resolve this issue, add an Edition ID to the image by running the following command (where *edition_id* is the appropriate SKU ID as defined in the original factory image or in the Windows ADK, *wim_file* is the name of the WIM file, *new_image_name* is the new image name, and *new_image_description* is the new description for the image):

Windows Command Prompt

```
imagex /flags <edition_id> /info <wim_file> 1 <new_image_name>
<new_image_description>
```

Import an Operating System from an Existing Windows Deployment Services Image

MDT allows you to import existing WIM images in Windows Deployment Services into the Deployment Workbench. Import the operating system using the Import Operating System Wizard in the Deployment Workbench.

To import an operating system from an existing image in Windows Deployment Services

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/Operating Systems (where *deployment_share* is the name of the deployment share to which the operating system will be added).
3. In the Actions pane, select **Import Operating System**.

   The Import Operating System Wizard starts.
4. Complete the Import Operating System Wizard using the information in Table 34.

Table 34. Information for Completing the Import Operating System Wizard

Expand table

On this wizard page	Do this
OS Type	Select **Custom image file**, and then select **Next**.
WDS Server	In **Server name**, type ***server_name*** (where *server_name* is the name of the Windows Deployment Services server that contains the existing WIM images), and then select Next.
Summary	Select **Next**.

On this wizard page	Do this
Confirmation	Select **Save Output** to save the output of the wizard to a file, or select View Script to view the Windows PowerShell scripts used to perform the wizard tasks. Select **Finish**.

The Import Operating System Wizard finishes. The operating system is added to the list of operating systems in the details pane but is not is copied to the deployment share. The Deployment Workbench leaves the operating system image on the Windows Deployment Services server, but the image is now available to LTI.

View Operating System Properties in the Deployment Workbench

You view operating system properties beneath the Operating System node in the Deployment Workbench using the **Properties** actions as described in View Item Properties in the Deployment Workbench.

Table 35 lists the configuration settings on the **General** tab of the operating system **Properties** dialog box and provides a description of each setting.

> **Note**
>
> The configuration settings on the **General** tab are populated when you import the operating system, and only the **Operating system name** can be modified. No other settings can be modified.

Expand table

Setting	Description
Operating system name	Contains the name of the operating system—for example, Windows 8 ENTERPRISE in Windows 8 x64 install.wim.
Description	Contains the description of the operating system—for example, Windows 8 ENTERPRISE.
OS type	Contains the type of operating system—for example, Windows Image-based Setup.
Platform	Contains the processor architecture of the operating system—for example, x64.

Setting	Description
Languages(s)	Contains the languages included in the operating system—for example, en-US.
Includes Setup	Contains **True** or **False**, which indicates whether the operating system includes the files necessary to perform setup.
Path	Contains the path to the operating system relative to the root of the deployment share.
Image file	Contains the path and file name of the operating system relative to the root of the deployment share.
Image index	Contains a numeric value that indicates the image index for the image—for example, 1.
Image name	Includes the image name—for example, Windows 8 ENTERPRISE.
Image size (MB)	Contains the size of the image in megabytes—for example, 7921 indicates 7,921 MB, or 7.921 GB.
HAL	Contains the HAL type for the image—for example, acpiapic.

Copy an Operating System in the Deployment Workbench

Copy and paste operating systems and folders beneath the Operating System node in the Deployment Workbench using the **Copy** and **Paste** actions as described in Copy Items in the Deployment Workbench.

Move an Operating System in the Deployment Workbench

Move operating systems and folders beneath the Operating System node in the Deployment Workbench using the **Cut** and **Paste** actions as described in Move Items in the Deployment Workbench.

Rename an Operating System in the Deployment Workbench

Rename operating systems and folders beneath the Operating System node in the Deployment Workbench using the **Rename** action as described in Rename Items in the Deployment Workbench.

Delete an Operating System from the Deployment Workbench

Delete operating systems and folders beneath the Operating System node in the Deployment Workbench using the Delete Selected Items Wizard as described in Delete Items from the Deployment Workbench. The Delete Selected Items Wizard allows deletion of individual operating system files or entire folder structures.

Manage Folders for Operating Systems in the Deployment Workbench

You can manage folders beneath the Operating Systems node in the Deployment Workbench to create hierarchical groupings of operating systems. For more information on:

- Managing folders, see Manage Folders in the Deployment Workbench
- Selection profiles, see Manage Selection Profiles

Configuring Applications in the Deployment Workbench

MDT uses the Deployment Workbench to manage the applications deployed to the reference and target computers in your organization. Configure the applications in the Deployment Workbench by:

- Creating a new application as described in Create a New Application in the Deployment Workbench
- Viewing and configuring an application as described in View and Configure an Application in the Deployment Workbench
- Copying an application as described in Copy an Application in the Deployment Workbench
- Moving an application as described in Move an Application in the Deployment Workbench
- Renaming an application as described in Rename an Application in the Deployment Workbench
- Deleting an application as described in Delete an Application from the Deployment Workbench
- Managing folders for applications as described in Manage Folders for Applications in the Deployment Workbench

- Enabling or disabling an application as described in Enable or Disable an Application in the Deployment Workbench
- Preventing an application from being visible as described in Prevent an Application from Being Visible in the Deployment Wizard
- Configuring the computer to restart as described in Configure the Computer to Restart After Application Installation
- Customizing application installation as described in Customize Application Installation in Task Sequences

  In addtion to managing applications in the Deployment Workbench, you can manage applications using the MDT Windows PowerShell cmdlets. For more information on managing applications using the MDT Windows PowerShell cmdlets, see the following sections beneath the section, "MDT Windows PowerShell Cmdlets", in the MDT document *Toolkit Reference*:

- **Get-MDTDeploymentShareStatistics**
- **Import-MDTApplication**

Create a New Application in the Deployment Workbench

You can create new applications in the Deployment Workbench using one of the options listed in Table 36. You import operating systems into the Deployment Workbench using the New Application Wizard.

Table 36. Options for Creating a New Application

Expand table

Option	Select this option to create an application when
Application with source files	The application source files are not available to the destination computer when installing the operating system build or when they must be customized. When you select this option, the application files are copied to the deployment share, and the deployment process installs the application from the deployment share.
Application without source files or elsewhere on the network	The application source files are available during installation of the operating system build or to run a command that requires no application source files. When you select this option, no application source files are copied to the deployment share. Instead, the deployment process installs the application from another location on the network.

Option	Select this option to create an application when
Application bundle	A list of applications that you must install in a particular order. This list is specified as dependencies for the application bundle, but the bundle itself does not install anything.

Start the New Application Wizard using one of the following methods:

- In the Deployment Workbench console tree, select the Applications node or a folder beneath the Applications node. Then, in the Actions pane, select **New Application**.
- In the Deployment Workbench console tree, select the Applications node or a folder beneath the Applications node. Then, from the **Action** menu, select **New Application**.
- In the Deployment Workbench console tree, select the Applications node or a folder beneath the Applications node. Then, select **New Application**.

  You can create a new application in the Deployment Workbench for:

- Applications to be deployed from the deployment share as described in Create a New Application That Is Deployed from the Deployment Share
- Applications to be deployed from another network shared folder as described in Create a New Application That Is Deployed from Another Network Shared Folder
- Deploying application dependencies as described in Create a New Application for Deploying Application Dependencies

Create a New Application That Is Deployed from the Deployment Share

MDT allows you to use the New Application Wizard in the Deployment Workbench to create new applications that are deployed from the deployment share. The New Application Wizard copies source files for this type of applications to the deployment share.

To create a new application that is deployed from the deployment share

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.

2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/Applications (where *deployment_share* is the name of the deployment share to which the application will be added).

3. In the Actions pane, select **New Application**.

   The New Application Wizard starts.

4. Complete the New Application Wizard using the information in Table 37.

Table 37. Information for Completing the New Application Wizard

Expand table

On this wizard page	Do this
Application Type	Select **Application with source files**, and then select **Next**.
Details	- In **Publisher**, type ***publisher_name*** (where *publisher_name* is the name of the application's publisher). - In **Application Name**, type ***application_name*** (where *application_name* is the descriptive name of the application). - In **Version**, type **version** (where *version* is the version of the application). - In **Language**, type **language** (where *language* is the language of the application). - Select **Next**.
Source	a. In **Source directory**, type ***source_folder*** (where *source_folder* is the fully qualified path to the folder containing the application source files). Alternatively, select **Browse** to find the source folder. b. Select or clear the **Move the files to the deployment share instead of copying them** check box based on your requirements, and then select **Next**. This check box determines whether the wizard copies or moves the application source files to the deployment share. If the check box is:

On this wizard page	Do this
	- Selected, the wizard moves the source files to the deployment share - Cleared, the wizard copies the source files to the deployment share By default, this check box is cleared.
Destination	In **Specify the name of the directory that should be created**, type ***destination_folder*** (where *destination_folder* is the name of the folder in the deployment share that will contain the application source files), and then select **Next**.
Command Details	- In **Command line**, type ***command_line*** (where *command_line* is the command line to be run to start the installation of the application, including any command-line parameters). - In **Working directory**, type ***working_directory*** (where *working_directory* is the fully qualified or relative path for the folder designated as the working directory for the application). - Select **Next**.
Summary	Select **Next**.
Confirmation	Select **Save Output** to save the output of the wizard to a file, or select **View Script** to view the Windows PowerShell scripts used to perform the wizard tasks. Select **Finish**.

The New Application Wizard finishes. The application is added to the list of operating systems in the details pane in the Deployment Workbench.

Create a New Application That Is Deployed from Another Network Shared Folder

MDT allows for the creation of new applications that you deploy from a network shared folder other than the deployment share. Create a new application using the New Application Wizard in the Deployment Workbench. The New Application Wizard does not copy the source files for this type of application.

To create a new application that is deployed from a network shared folder other than the deployment share

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.

2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/Applications (where *deployment_share* is the name of the deployment share to which the application will be added).

3. In the Actions pane, select **New Application**.

   The New Application Wizard starts.

4. Complete the New Application Wizard using the information in Table 38.

Expand table

On this wizard page	Do this
Application Type	Select **Application without source files or elsewhere on the network**, and then select **Next**.
Details	- In **Publisher**, type ***publisher_name*** (where *publisher_name* is the name of the application's publisher). - In **Application Name**, type ***application_name*** (where *application_name* is the descriptive name of the application). - In **Version**, type ***version*** (where *version* is the version of the application). - In **Language**, type ***language*** (where *language* is the language of the application). - Select **Next**.
Command Details	- In **Command line**, type ***command_line*** (where *command_line* is the command line to be run to start the installation of the application, including any command-line parameters). - In **Working directory**, type ***working_directory*** (where *working_directory* is the fully qualified or relative path for the folder designated as the working directory for the application). - Select **Next**.
Summary	Select **Next**.
Confirmation	Select **Save Output** to save the output of the wizard to a file, or select **View Script** to view the Windows PowerShell scripts used to perform the

On this wizard page	Do this
	wizard tasks. Select **Finish**.

The New Application Wizard finishes. The application is added to the list of operating systems in the details pane of the Deployment Workbench.

Create a New Application for Deploying Application Dependencies

MDT allows for the creation of new applications used to deploy only the dependencies for an application instead of installing the application itself. Create a new application using the New Application Wizard in the Deployment Workbench. The New Application Wizard does not copy source files to the deployment share.

To create a new application for deploying application dependencies

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/Applications (where *deployment_share* is the name of the deployment share to which the application will be added).
3. In the Actions pane, select **New Application**.

   The New Application Wizard starts.
4. Complete the New Application Wizard by using the information in Table 39.

Table 39. Information for Completing the New Application Wizard

Expand table

On this wizard page	Do this
Application Type	Select **Application bundle**, and then select **Next**.

On this wizard page	Do this
Details	- In **Publisher**, type ***publisher_name*** (where *publisher_name* is the name of the application's publisher). - In **Application Name**, type ***application_name*** (where *application_name* is the descriptive name of the application). - In **Version**, type ***version*** (where *version* is the version of the application). - In **Language**, type ***language*** (where *language* is the language of the application). - Select **Next**.
Summary	Select **Next**.
Confirmation	Select **Save Output** to save the output of the wizard to a file, or select **View Script** to view the Windows PowerShell scripts used to perform the wizard tasks. Select **Finish**.

The New Application Wizard finishes. The application is added to the list of operating systems in the details pane of the Deployment Workbench.

View and Configure an Application in the Deployment Workbench

View the properties of applications beneath the Applications node in the Deployment Workbench using the **Properties** actions as described in View Item Properties in the Deployment Workbench. Configure an application in the Deployment Workbench by performing the following steps in the **Application Properties** dialog box:

- Configure properties on the **General** tab as described in Configure the Application Properties General Tab.
- Configure properties on the **Details** tab as described in Configure the Application Properties Details Tab.
- Configure properties on the **Dependencies** tab as described in Configure the Application Properties Dependencies Tab.
- Configure the properties on the **Office Products** tab as described in Configure the Application Properties Office Products Tab.

Configure the Application Properties General Tab

The application properties stored on the **General** tab are mostly configured when the New Application Wizard runs. Update the application properties on the **General** tab through the ***application_name*** **Properties** dialog box (where *application_name* is the name of the application in the Deployment Workbench).

To configure the General tab for application properties

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/Applications (where *deployment_share* is the name of the deployment share to which the application will be added).
3. In the details pane, select ***application_name*** (where *application_name* is the name of the application to configure).
4. In the Actions pane, select **Properties**.

   The ***application_name*** **Properties** dialog box opens (where *application_name* is the name of the application to configure).
5. On the **General** tab, configure the settings listed in Table 40 based on the requirements of your organization, and then select **OK**.

Table 40. Configuration Settings on the General Tab of Application Properties

Expand table

Setting	Description
Name	Contains the name of the application displayed in the Deployment Workbench. If **Display name** is not configured, this value is also displayed in the Deployment Wizard.
Comments	Provides information about the application.
Display name	(Optional) Contains the name displayed in the Deployment Wizard instead of the value in **Name**. If no value is specified, the value in **Name** is displayed in the Deployment Wizard.

Setting	Description
Short name	Contains the name of the folder in which the application resides.
Version	Contains the version number of the application. Enter the version number in this box; it is not validated against the actual application version number but is provided for informational purposes.
Publisher	Contains the name of the application's publisher. Enter the publisher in this box; it is not validated against the actual application version number but is provided for informational purposes.
Language	Contains the language of the application. Enter the language in this box; it is not validated against the actual application language but is provided for informational purposes.
Source directory	Configures the folder in which the source of the application files resides.
Application GUID	Contains the GUID for the application.
Hide this application in the Deployment Wizard	Select to control when this application appears in the Deployment Wizard. If the check box is: - Selected, the Deployment Wizard will not display this application. - Cleared, the Deployment Wizard displays this application. This check box is cleared by default. This setting is also shown in the **Hide** column in the details pane of the Deployment Workbench.
Enable this application	Select to control when this application is available to other wizards and dialog boxes in the Deployment Workbench. If the check box is: - Selected, other wizards and dialog boxes in the Deployment Workbench are able to select this application - Cleared, other wizards and dialog boxes in the Deployment Workbench are unable to select this application

Setting	Description
	This check box is selected by default.

The application configuration settings are saved, and the modifications are displayed in the details pane of the Deployment Workbench.

Configure the Application Properties Details Tab

The application configuration settings stored on the **Details** tab are initially configured when the New Application Wizard runs. Update the application properties on the **Details** tab through the ***application_name*** **Properties** dialog box (where *application_name* is the name of the application in the Deployment Workbench).

To configure the Details tab for application properties

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/Applications (where *deployment_share* is the name of the deployment share to which the application will be added).
3. In the details pane, select ***application_name*** (where *application_name* is the name of the application to configure).
4. In the Actions pane, select **Properties**.

   The ***application_name*** **Properties** dialog box opens (where *application_name* is the name of the application to configure).
5. On the **Details** tab, configure the settings listed in Table 41 based on the requirements of your organization, and then select **OK**.

Table 41. Configuration Settings on the Details Tab of Application Properties

Expand table

Setting	Description
Application bundle	Select to configure the application to install application dependencies but not the application. The other available option is **Standard application**.
Standard application	Select to configure the application to be a standard application that has source files, a command line, and other options listed on this tab. The other available option is **Application bundle**.
Quiet install command	Configures the command line to run for performing an unattended, or quiet, installation of the application. This text box is enabled only when you select the **Standard application** option.
Working directory	Configures the working directory of the application and is enabled only when you select **Standard application**.
Uninstall registry key name	Configures the registry subkey **HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Uninstall**, used to determine whether the application is already installed on the target computer. If MDT detects the presence of the subkey, it assumes that the application is already installed and skips the installation of the application and any dependencies. This text box is enabled only when you select **Standard application**.
Reboot the computer after installing this application	Select to configure the MDT deployment process to restart the target computer after installing this application. If the check box is: - Selected, the target computer restarts after installing the application - Cleared, the target computer will not restart after installing the application This check box is cleared by default.
This can run on any platform	Select to configure the application to run on any supported 32-bit or 64-bit Windows operating system. The other available option is **This can run only on the specified client platforms**.
This can run only on the specified client platforms	Select to configure the application to run on any supported 32-bit or 64-bit Windows operating system. The other available option is **This can run on any platform**.

The application configuration settings are saved, and the modifications are displayed in the details pane of the Deployment Workbench.

Configure the Application Properties Dependencies Tab

MDT checks the dependencies of an application before installing the application. Similarly, MDT ensures that all application dependencies are installed before installing the application.

> **Note**
>
> Application dependencies are installed even if you do not select the dependencies separately from the application. Also, application dependencies override any rules defined in CustomSettings.ini or in the MDT DB.

When you define more than one application dependency, you can configure the order in which the dependencies are installed, thereby ensuring that the dependencies are installed in a specific order. Update the application properties on the **Dependencies** tab through the ***application_name*** **Properties** dialog box (where *application_name* is the name of the application in the Deployment Workbench).

To configure the Dependencies tab for application properties

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/Applications (where *deployment_share* is the name of the deployment share to which you will add the application).
3. In the details pane, select ***application_name*** (where *application_name* is the name of the application you are configuring).
4. In the Actions pane, select **Properties**.

   The ***application_name*** **Properties** dialog box opens (where *application_name* is the name of the application to configure).
5. On the **Dependencies** tab, configure the settings by performing one of the actions in Table 42 based on the requirements of your organization, and then select **OK**.

Table 42. Actions on the Dependencies Tab of Application Properties

Expand table

Action	Description
Add	Adds a new application dependency to the list of dependencies using the **Select an item** dialog box. You can add any applications that already exist in the deployment share.
Remove	Removes an application dependency from the list of dependencies.
Up	Moves an application dependency higher in the sequence of installed dependencies. Application dependencies are installed from the top of the list to the bottom.
Down	Moves an application dependency lower in the sequence of installed dependencies. Application dependencies are installed from the top of the list to the bottom.

The application configuration settings are saved, and the modifications are displayed in the details pane of the Deployment Workbench.

Configure the Application Properties Office Products Tab

The application properties stored on the **Office Products** tab are mostly configured when the New Application Wizard runs. Update the application properties on the **Office Products** tab through the ***application_name*** **Properties** dialog box (where *application_name* is the name of the application in the Deployment Workbench).

> ⓘ **Note**
>
> This tab is displayed when you create an application for Microsoft Office. For all other applications, this tab is not displayed.

To configure the Office Products tab for application properties

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/Applications (where *deployment_share* is the name of the deployment share to which the application will be added).

3. In the details pane, select ***application_name*** (where *application_name* is the name of the application to configure).

4. In the Actions pane, select **Properties**.

   The ***application_name*** **Properties** dialog box opens (where *application_name* is the name of the application to configure).

5. On the **Office Products** tab, configure the settings listed in Table 43 based on the requirements of your organization, and then select **OK**.

Table 43. Configuration Settings on the Office Products Tab of Application Properties

Expand table

Setting	Description
Office product to install	Selects the Microsoft Office products to be installed.
Office languages	Use to select each language included in the source image. By default, Microsoft Office Setup installs the same language as the target operating system. Use these check boxes to force the installation of specific language packs.
Product key	Select to determine whether the Deployment Workbench configures the Config.xml file for Microsoft Office Setup to provide a product key. If the check box is: - Selected, enter the product key in the associated box to automatically configure the Config.xml file - Cleared, the product key is provided during Microsoft Office Setup or in a Windows Installer (MSP) configuration file
Customer name	Select to determine whether the Deployment Workbench configures the Config.xml for Microsoft Office Setup to provide the customer name. If the check box is: - Selected, enter the customer name in the associated box to automatically configure the Config.xml file - Cleared, the customer name is provided during Microsoft Office Setup or in an MSP configuration file

Setting	Description
Display level	Select to determine whether the Deployment Workbench configures Config.xml for Microsoft Office Setup to configure the display level of the setup process. If the check box is: - Selected, select the display level in the associated box to automatically configure the Config.xml file - Cleared, the display level is provided during Microsoft Office Setup or in an MSP configuration file
Accept EULA	Select to determine whether the Deployment Workbench configures Config.xml for Microsoft Office Setup to automatically accept the end user license agreement (EULA) during the Setup process. If the check box is: - Selected, the Config.xml file is configured to automatically accept the EULA - Cleared, EULA acceptance is provided during Microsoft Office Setup or in an MSP configuration file
Cache only	Select to determine whether the Deployment Workbench configures Config.xml for Microsoft Office Setup to install the local installation source (LIS) cache to the target computer during the setup process but not install Microsoft Office. If the check box is: - Selected, the Config.xml file is configured to copy the LIS cache during Microsoft Office Setup but not install Microsoft Office products - Cleared, the LIS cache is copied and Microsoft Office products are installed during Microsoft Office Setup
Always suppress reboot	Select to determine whether the Deployment Workbench configures Config.xml for Microsoft Office Setup to prevent restarting the target computer during the setup process. If the check box is: - Selected, the Config.xml file is configured to prevent a restart of the target computer during Microsoft Office Setup - Cleared, the target computer can be restarted during Microsoft Office Setup
Add	Select to add Microsoft Office language packs.
Edit Config.xml	Select to modify the contents of the Config.xml file that the Deployment Workbench generates.

The application configuration settings are saved, and the modifications are displayed in the details pane of the Deployment Workbench.

Copy an Application in the Deployment Workbench

Copy and paste applications and folders beneath the Applications node in the Deployment Workbench by using the **Copy** and **Paste** actions as described in Copy Items in the Deployment Workbench.

Move an Application in the Deployment Workbench

Move applications and folders beneath the Applications node in the Deployment Workbench by using the **Cut** and **Paste** actions as described in Move Items in the Deployment Workbench.

Rename an Application in the Deployment Workbench

Rename applications and folders beneath the Applications node in the Deployment Workbench by using the **Rename** action as described in Rename Items in the Deployment Workbench.

Delete an Application from the Deployment Workbench

Delete applications and folders beneath the Applications node in the Deployment Workbench using the Delete Selected Items Wizard as described in Delete Items from the Deployment Workbench. The Delete Selected Items Wizard allows deletion of individual applications or entire folder structures.

> ⓘ **Note**
>
> You should not delete an application when other applications are dependent on it. However, the Deployment Workbench does not enforce this recommendation.

Manage Folders for Applications in the Deployment Workbench

You can manage folders beneath the Applications node in the Deployment Workbench to create hierarchical groupings of applications. For more information on:

- Managing folders, see Manage Folders in the Deployment Workbench
- Selection profiles, see Manage Selection Profiles

Enable or Disable an Application in the Deployment Workbench

Control whether applications are available to other wizards and dialog boxes in the Deployment Workbench by using the **Enable this application** check box on the **General** tab of the application **Properties** dialog box.

> **Tip**
>
> To configure an application so that it can only be installed during a task sequence step, disable the application. Doing so allows the application to be installed during the task sequence but prevents the application from appearing in the list of available applications.

For more information on enabling or disabling applications in the Deployment Workbench, see Configure the Application Properties General Tab.

Prevent an Application from Being Visible in the Deployment Wizard

Prevent an application from being visible in the Deployment Wizard by selecting the **Hide this application in the Deployment Wizard** check box on the **General** tab of the application **Properties** dialog box, as described in Configure the Application Properties General Tab.

> **Note**
>
> The status of the **Hide this application in the Deployment Wizard** check box is shown in the **Hide** column in the details pane of the **Application** node.

Configure the Computer to Restart After Application Installation

Restart the target computer after installing an application by selecting the **Restart the computer after installing this application** check box on the **Details** tab of the application **Properties** dialog box. Selecting this check box causes the Deployment Wizard to restart the target computer after installing the application, and then continue with the next step in the task sequence.

> **Caution**
>
> Do not allow the application to restart the target computer. MDT must control restarts, or the task sequence will fail. For example, use the command **REBOOT=REALLYSUPPRESS** to prevent some Windows Installer-based applications

from restarting. To prevent Microsoft Office from restarting the computer, add the property **SETUP_REBOOT=NEVER** to the Config.xml file or the MST file created by using the Office Customization Tool.

For more information on how to configure MDT to restart the target computer after installing an application, see Configure the Application Properties Details Tab.

Customize Application Installation in Task Sequences

Adding applications in the Applications node in a deployment share through the Deployment Workbench is the simplest method of deploying most applications. MDT task sequences deploy applications by using the **Install Application** task sequence type. Some of task sequence templates included in MDT have the **Install Applications** task sequence step in the **State Restore** group, which is based on the **Install Application** task sequence type.

The **Install Application** task sequence type allows for installation of one or more applications in a single task sequence step using one of the configuration options listed in Table 44.

Table 44. Configuration Settings on the Properties Tab of the Install Application Task Sequence

Expand table

Setting	Description
Install multiple applications	Select to install one or more applications in a single task sequence step. This configuration option allows installation of any applications that you: - Select in the Deployment Wizard - Specify in the **Applications** property in CustomSettings.ini or the MDT DB - Specify in the **MandatoryApplications** property in CustomSettings.ini or the MDT DB You use the **Success codes** box in conjunction with this option to identify the application installation return codes that indicate a successful application deployment. The default values in this box are **0** and **3010** for the task sequence step in the templates included in MDT. This configuration option is the default selection for the **Install Applications** task sequence step.

Setting	Description
	For more information on the: - **Applications** property, see the section, "Applications", in the MDT document *Toolkit Reference*. - **MandatoryApplications** property, see the section, "MandatoryApplications", in the MDT document *Toolkit Reference*.
Install single application	Select to install one or more applications in a single task sequence step. You use the **Application to install** box in conjunction with this option to select the application to install, including any application dependencies for the selected application.

Customize the application-deployment process in the task sequences by:

- Configuring the existing **Install Applications** task sequence step in the **State Restore** group as described in Configure an Existing Install Applications Task Sequence Step
- Creating a new task sequence step based on the **Install Application** task sequence type as described in Create a New Task Sequence Step for Installing Applications

Configure an Existing Install Applications Task Sequence Step

Configure an existing **Install Applications** task sequence step by modifying the configuration settings on the **Properties** tab of the task sequence step.

To configure an existing Install Applications task sequence step

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/Task Sequences (where *deployment_share* is the name of the deployment share to which the application will be added).
3. In the details pane, select ***task_sequence*** (where *task_sequence* is the name of the task sequence to configure).
4. In the Actions pane, select **Properties**.

The ***task_sequence*** **Properties** dialog box opens (where *task_sequence* is the name of the application to configure).

5. In the ***task_sequence*** **Properties** dialog box (where *task_sequence* is the name of the application you want to configure), select the **Task Sequence** tab.
6. In the hierarchy of the task sequence, go to State Restore/Install Applications.
7. On the **Properties** tab, configure the settings listed in Table 45 based on the requirements of your organization, and then select **OK**.

Table 45. Configuration Settings on the Properties Tab of the Install Applications Task Sequence

Expand table

Setting	Description
Name	Configures the name of the task sequence step displayed in the task sequence hierarchy.
Description	Configures the description text for the task sequence step.
Install multiple applications	Select to configure the task sequence step to install any applications that you: - Select in the Deployment Wizard - Specify in the **Applications** property in CustomSettings.ini or the MDT DB - Specify in the **MandatoryApplications** property in CustomSettings.ini or the MDT DB
Success codes	Configures the list of success codes for the application-installation software. Each success code is separated by a space. This text box is only enabled when you select the **Install multiple applications** option.
Install a single application	Configures the task sequence step to install only the application listed in the **Application to install** box.
Application to install	Configures the application to be installed when you select the **Install a single application** option. Select the application to install by selecting **Browse**. This text box is enabled only when you select **Install a single application**.

The updated task sequence appears in the details pane of the Deployment Workbench.

Create a New Task Sequence Step for Installing Applications

In most instances, the existing **Install Applications** task sequence step is sufficient for installing applications to target computers. However, there are instances in which the existing **Install Applications** task sequence step may not be sufficient for your organization's requirements, or you may need to install an application at a different sequence in the task sequence.

For example, the installation process for some device drivers is performed more like an application installation than the typical installation process for a traditional device driver. You can install these device drivers by creating a new task sequence step based on the **Install Application** task sequence type.

> **Tip**
>
> Disable the existing **Install Applications** step in the task sequence, and add all applications manually using the task sequence controls. The benefits of this approach are that you can easily select and insert applications into the task sequence in any order necessary, simplifying management of a large number of applications.

To create a new task sequence step for installing applications

1. Create a new task sequences step based on the **Install Application** type at the appropriate place in the task sequence hierarchy as described in Configure the Task Sequence Properties Task Sequence Tab.
2. Configure the new task sequence step to deploy one or more applications as described in Configure an Existing Install Applications Task Sequence Step.

Configuring Packages in the Deployment Workbench

Packages in MDT are operating system software installed on the target computers and stored in CAB or MSU files, such as security updates, service packs, feature packs, or language packs. Manage the packages to be deployed to the reference and target computers in your organization using the Deployment Workbench. You configure packages in the Deployment Workbench in a deployment share's Packages node by:

- Importing a new package as described in Import a New Package into the Deployment Workbench
- Modifying an existing package as described in Modify an Existing Package in the Deployment Workbench
- Copying a package as described in Copy a Package in the Deployment Workbench
- Moving a package as described in Move a Package in the Deployment Workbench
- Renaming a package as described in Rename a Package in the Deployment Workbench
- Deleting a package as described in Delete a Package from the Deployment Workbench
- Managing folders for packages as described in Manage Folders for Packages in the Deployment Workbench
- Enabling or disabling a package Enable or Disable a Package in the Deployment Workbench
- Preventing a package from being visible as described in Prevent a Package from Being Visible in the Deployment Wizard
- Customizing package installation as described in Customize Package Installation in Task Sequences

  In addtion to managing operating system packages in the Deployment Workbench, you can manage operating system packages using the MDT Windows PowerShell cmdlets. For more information on managing operating system packages using the MDT Windows PowerShell cmdlets, see the following sections beneath the section, "MDT Windows PowerShell Cmdlets", in the MDT document *Toolkit Reference*:

- **Get-MDTDeploymentShareStatistics**
- **Import-MDTPackage**

Import a New Package into the Deployment Workbench

Import packages into the Deployment Workbench by using the Import OS Packages Wizard. Start the Import OS Packages Wizard using one of the following methods:

- In the console tree, select the Packages node or a folder beneath the Packages node. Then, in the Actions pane, select **Import OS Packages**.
- In the console tree, select the Packages node or a folder beneath the Packages node. Then, from the **Action** menu, select **Import OS Packages**.
- In the console tree, select the Packages node or a folder beneath the Packages node, and then select **Import OS Packages**.

To import a new package

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/Packages (where *deployment_share* is the name of the deployment share to which you will add the application).
3. In the Actions pane, select **Import OS Packages**.The Import OS Packages Wizard starts.
4. Complete the Import OS Packages Wizard using the information in Table 46.

Table 46. Information for Completing the Import OS Packages Wizard

Expand table

On this wizard page	Do this
Specify Directory	In **Package source directory**, type ***path*** (where *path* is the fully qualified path to the folder that contains the package you want to import), and then select **Next**. You can alternatively select **Browse** to find the folder on a local drive or network shared folder.
Summary	View the information in the **Details** box, and then select **Next**.
Confirmation	You can select **Save Output** to save the output of the wizard to a file. You can also select **View Script** to view the Windows PowerShell scripts used to perform the wizard tasks.

On this wizard page	Do this
	Select Finish.

The Import OS Packages Wizard finishes. The package is added to the list of packages in the details pane of the Deployment Workbench.

Modify an Existing Package in the Deployment Workbench

Modify packages in the Packages node in the Deployment Workbench using the **Properties** actions as described in View Item Properties in the Deployment Workbench. The package properties are mostly configured when you run the Import OS Packages Wizard. Update the package properties on the **General** tab through the ***package_name*** **Properties** dialog box (where *package_name* is the name of the application in the Deployment Workbench).

To modify an existing package

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/Packages (where *deployment_share* is the name of the deployment share in which you will configure the package).
3. In the details pane, select ***package_name*** (where *package_name* is the name of the package you want to configure).
4. In the Actions pane, select **Properties**.

   The ***package_name*** **Properties** dialog box opens (where *package_name* is the name of the package you want to configure).
5. On the **General** tab, configure the settings listed in Table 47 based on the requirements of your organization, and then select **OK**.

Table 47. Configuration Settings on the General Tab of the Package Properties

Expand table

Setting	Description
Name	Contains the name of the package displayed in the Deployment Workbench. If **Display name** is not configured, this value is also displayed in the Deployment Wizard.
Comments	Provides information about the package.
Display name	(Optional) Contains the name displayed in the Deployment Wizard instead of the value in **Name**. If no value is specified, the value in **Name** is displayed in the Deployment Wizard.
Type	Type of package, which typically includes the following high-level types of packages: - Language packs - Hotfix patches - Feature packs The package type in this text box is automatically determined by the Deployment Workbench and cannot be modified.
Processor architecture	Target processor architecture for the package; it can be x86, amd64, or ia64. The processor architecture in this box is automatically determined by the Deployment Workbench and cannot be modified.
Language	Contains the language of the application. The language in this box is automatically determined by the Deployment Workbench and cannot be modified.
Keyword	Used to identify the version of the language pack. The keyword in this text box is automatically determined by the Deployment Workbench and cannot be modified.
Public key token	Contains the public key token that MDT uses to updated the unattended.xml file. The public key token in this text box is automatically determined by the Deployment Workbench and cannot be modified.
Version	Contains the version number of the package.

Setting	Description
	The version number in this text box is automatically determined by the Deployment Workbench and cannot be modified.
Product name	Contains the name of the product for which the package is intended. The product name in this text box is automatically determined by the Deployment Workbench and cannot be modified.
Product version	Contains the version number of the product for which the package is intended. The product version number in this text box is automatically determined by the Deployment Workbench and cannot be modified.
Package path	Contains the path of the package relative to the root of the deployment share. The path in this text box is automatically determined by the Deployment Workbench and cannot be modified.
Hide this package in the Deployment Wizard	Select to control when this package appears in the Deployment Wizard. If the check box is: - Selected, the Deployment Wizard will not display this package - Cleared, the Deployment Wizard displays this package The check box is cleared by default.
Enable (approve) this package	Select to control when this package is available to other wizards and dialog boxes in the Deployment Workbench. If the check box is: - Selected, other wizards and dialog boxes in the Deployment Workbench are able to select this package - Cleared, other wizards and dialog boxes in the Deployment Workbench are unable to select this package The check box is selected by default.

The package configuration settings are saved, and the modifications are displayed in the details pane of the Deployment Workbench.

Copy a Package in the Deployment Workbench

You can copy and paste packages and folders beneath the Packages node in the Deployment Workbench using the **Copy** and **Paste** actions as described in Copy Items in the Deployment Workbench.

Move a Package in the Deployment Workbench

You can move packages and folders beneath the Packages node in the Deployment Workbench using the **Cut** and **Paste** actions as described in Move Items in the Deployment Workbench.

Rename a Package in the Deployment Workbench

You can rename packages and folders beneath the Packages node in the Deployment Workbench using the **Rename** action as described in Rename Items in the Deployment Workbench.

Delete a Package from the Deployment Workbench

You can delete packages and folders beneath the Applications node in the Deployment Workbench using the Delete Selected Items Wizard as described in Delete Items from the Deployment Workbench. The Delete Selected Items Wizard allows you to delete individual package or entire folder structures.

Manage Folders for Packages in the Deployment Workbench

You can manage folders beneath the Packages node in the Deployment Workbench to create hierarchical groupings of operating system packages. For more information on:

- Managing folders, see Manage Folders in the Deployment Workbench
- Selection profiles, see Manage Selection Profiles

Enable or Disable a Package in the Deployment Workbench

You can control whether packages are available to other wizards and dialog boxes in the Deployment Workbench by selecting the **Enable (approve) this package** check box on the **General** tab of the package **Properties** dialog box.

> **Tip**

If you want to configure a package so that it can only be installed during a task sequence step, disable the application. Doing so allows the package to be installed during the task sequence but prevents the application from appearing in the list of available package.

For more information on enabling or disabling packages in the Deployment Workbench, see Configuring Packages in the Deployment Workbench.

Prevent a Package from Being Visible in the Deployment Wizard

You can prevent a package from being visible in the Deployment Wizard by selecting the **Hide this application in the Deployment Wizard** check box on the **General** tab of the application **Properties** dialog box. For more information on preventing packages from appearing in the Deployment Wizard, see Configuring Packages in the Deployment Workbench.

Customize Package Installation in Task Sequences

Adding packages in a deployment share's Packages node through the Deployment Workbench is the simplest method for deploying most packages. MDT task sequences deploy packages using the **Install Updates Offline** task sequence type. Some of task sequence templates included in MDT have the **Apply Patches** task sequence step in the **Preinstall/Refresh only**group, which is based on the **Install Updates Offline** task sequence type.

The **Install Updates Offline** task sequence type allows you to install one or more packages in a single task sequence step using selection profiles, which allow one or more packages to be selected and deployed as a unit. For more information managing selection profiles, see Manage Selection Profiles.

Customize the package deployment process in your task sequences by:

- Configuring the existing **Apply Patches** task sequence step in the **Preinstall** group as described in Configure an Existing Apply Patches Task Sequence Step
- Creating a new task sequence step based on the **Install Updates Offline** task sequence type as described in Create a New Task Sequence Step for Installing Packages
- Adding language packs to task sequence steps as described in Add Language Packs to Task Sequence Steps

Configure an Existing Apply Patches Task Sequence Step

You configure an existing **Apply Patches** task sequence step by modifying the configuration settings on the **Properties** tab of the task sequence step.

To configure an existing Apply Patches task sequence step

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/Task Sequences (where *deployment_share* is the name of the deployment share to which you will add the application).
3. In the details pane, select ***task_sequence*** (where *task_sequence* is the name of the task sequence you want to configure).
4. In the Actions pane, select **Properties**.

   The ***task_sequence*** **Properties** dialog box opens (where *task_sequence* is the name of the application you want to configure).
5. In the ***task_sequence*** **Properties** dialog box, select the **Task Sequence** tab (where *task_sequence* is the name of the application you want to configure).
6. In the hierarchy of the task sequence, go to Preinstall/Apply Patches.
7. On the **Properties** tab, configure the settings listed in Table 48 based on the requirements of your organization, and then select **OK**.

Table 48. Configuration Settings on the Properties Tab of the Install Applications Task Sequence Step

Expand table

Setting	Description
Name	Configures the name of the task sequence step displayed in the task sequence hierarchy.
Description	Configures the description text for the task sequence step.
Selection profile	Used to select the selection profile that contains the updates you want to deploy in this task sequence step. The selection profile can contain one or

Setting	Description
	more packages to be deployed.

The updated task sequence appears in the details pane of the Deployment Workbench.

Create a New Task Sequence Step for Installing Packages

In most instances, the existing **Apply Patches** task sequence step is sufficient for installing packages to target computers. However, there are instances in which the existing **Apply Patches** task sequence step may not be sufficient for your requirements or you may need to install a package at a different place in the task sequence.

For example, the packages may need to be installed in a specific order or may have dependencies, such as installing a service pack before installing hotfixes. First, create folders and selection profiles for each grouping of packages that you wanted to install separately. Then, install the groups of packages by creating a new task sequence step for each group based on the **Install Updates Offline**-type task sequence step.

> **Tip**
>
> You can disable the existing **Apply Patches** step in the task sequence and add all packages manually using the task sequence controls. The benefit of this approach is that you easily select and insert packages into the task sequence in any order necessary. This simplifies management of a large number of packages.

To create a new task sequence step for installing packages

1. Create a new task sequences step based on the **Install Updates Offline** type at the appropriate place in the task sequence hierarchy as described in Configure the Task Sequence Properties Task Sequence Tab.
2. Configure the new task sequence step to deploy one or more packages by selecting the appropriate selection profile containing the packages to be installed as described in Configure an Existing Install Applications Task Sequence Step.

Add Language Packs to Task Sequence Steps

Language packs are one of the types of packages available in MDT and enable a multilingual Windows environment. Windows is now language neutral, and all language and locale resources are added to Windows through language packs (Lp.cab files). By

adding one or more language packs to Windows those languages can be enabled when installing the operating system. As a result, the same Windows image can be deployed to regions with different language and locale settings, reducing development and deployment time.

See the following references for additional information about language packs in Windows:

- For instructions on installing language packs during deployment, see Running the Deployment Wizard.
- For the configuration properties for installing language packs automatically, see the MDT document *Toolkit Reference*.
- For more information about Windows language packs, see "Manage Language Packs for Windows" in the Windows ADK.

Configuring Device Drivers in the Deployment Workbench

Integrate device drivers for the reference and target computers into Windows PE and the target operating system unless these components are included in Windows PE or the target operating system. The Deployment Workbench helps centralize and automate device driver management and integration for LTI by providing a centralized repository of device drivers, ensuring that the proper device drivers are deployed. The Deployment Workbench also automates the injection of the appropriate device drivers into Windows PE images that the Deployment Workbench generates. MDT supports different strategies for device driver management. For more information about device driver management strategies, see Managing Device Drivers.

Configure device drivers in the Deployment Workbench in a deployment share's Out-of-Box node by:

- Importing device drivers as described in Import Device Drivers into the Deployment Workbench
- Modifying existing device drivers as described in Modify Existing Device Drivers in the Deployment Workbench
- Copying device drivers as described in Copy Device Drivers in the Deployment Workbench
- Moving device drivers as described in Move Device Drivers in the Deployment Workbench

- Renaming device drivers as described in Rename Device Drivers in the Deployment Workbench
- Deleting device drivers as described in Delete Device Drivers from the Deployment Workbench
- Managing folders for device drivers as described in Manage Folders for Device Drivers in the Deployment Workbench
- Enabling or disabling device drivers as described in Enable or Disable Device Drivers in the Deployment Workbench
- Deploy specific device drivers to target computers for LTI deployments as described in Deploy Specific Device Drivers to Target Computers in LTI

  In addtion to managing device drivers in the Deployment Workbench, you can manage device drivers using the MDT Windows PowerShell cmdlets. For more information on managing device drivers using the MDT Windows PowerShell cmdlets, see the following sections beneath the section, "MDT Windows PowerShell Cmdlets", in the MDT document *Toolkit Reference*:

- **Import-MDTDriver**
- **Get-MDTDeploymentShareStatistics**

Import Device Drivers into the Deployment Workbench

Import device drivers into the Deployment Workbench using the Import Drivers Wizard. Start the Import Drivers Wizard using one of the following methods:

- In the console tree, select the Out-of-Box Drivers node or a folder beneath the Out-of-Box Drivers node. Then, in the Actions pane, select **Import Drivers**.
- In the console tree, select the Out-of-Box Drivers node or a folder beneath the Out-of-Box Drivers node. Then, from the **Action** menu, select **Import Drivers**.
- In the console tree, select the Out-of-Box Drivers node or a folder beneath the Out-of-Box Drivers node, and then select **Import Drivers**.

To import device drivers

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.

2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/Out-of-box Drivers node (where *deployment_share* is the name of the deployment share to which you will add the device drivers) or a folder beneath that node.

3. In the Actions pane, select **Import Drivers**.

   The Import Driver Wizard starts.

4. Complete the Import Driver Wizard using the information in Table 49.

Table 49. Information for Completing the Import Driver Wizard

Expand table

On this wizard page	Do this
Specify Directory	a. In **Driver source directory**, type **path** (where *path* is the fully qualified path to the folder that contains the device drivers you want to import). You can alternatively select **Browse** to find the folder on a local drive or network shared folder. b. Select or clear the **Import drivers even if they are duplicates of an existing driver** check box based on the requirements of your organization. If the check box is: - Selected, the wizard will import the drivers even if the same drivers already exist - Cleared, the wizard will not import the drivers if the same drivers already exist In most instances, do not select this check box, as doing so increases the size of the deployment share and makes driver management more complex. c. Select **Next**.
Summary	Select **Next**.
Confirmation	You can select **Save Output** to save the output of the wizard to a file. You can also select **View Script** to view the Windows PowerShell scripts used

On this wizard page	Do this
	to perform the wizard tasks. Select **Finish**.

The Import Drivers Wizard finishes. The device drivers are added to the list of device drivers in the details pane of the Deployment Workbench.

Modify Existing Device Drivers in the Deployment Workbench

Modify device drivers in the Out-of-Box Drivers node in the Deployment Workbench using the **Properties** action as described in View Item Properties in the Deployment Workbench. Configure device drivers in the Deployment Workbench by performing the following steps in the device driver **Properties** dialog box:

1. Configure properties on the **General** tab as described in Configure the Device Driver Properties General Tab.
2. View properties on the **Details** tab as described in View the Device Driver Properties Details Tab.

Configure the Device Driver Properties General Tab

The device driver properties stored on the **General** tab are mostly configured when you run the Import Device Drivers Wizard. Update the device driver properties on the **General** tab through the ***driver_name*** **Properties** dialog box (where *driver_name* is the name of the device driver in the Deployment Workbench).

To modify existing device drivers properties on the General tab

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/Out-of-Box Drivers (where *deployment_share* is the name of the deployment share in which you will configure the device driver).
3. In the details pane, select ***driver_name*** (where *driver_name* is the name of the device driver you want to configure).
4. In the Actions pane, select **Properties**.

The ***driver_name*** **Properties** dialog box opens (where *driver_name* is the name of the device driver you want to configure).

5. On the **General** tab, configure the settings listed in Table 50 based on the requirements of your organization, and then select **OK**.

Table 50. Configuration Settings on the General Tab of the Device Driver Properties

Expand table

Setting	Description
Driver name	Contains the name of the device driver displayed in the Deployment Workbench and the Deployment Wizard.
Comments	Provides information about the device driver.
Platforms: x86	Select to control whether this device driver is for 32-bit operating system. If the check box is: - Selected, the device driver is available for deployment to 32-bit operating systems - Cleared, the device driver is unavailable for deployment to 32-bit operating systems If the Deployment Workbench incorrectly detects the platforms that the device driver supports, you can clear the platform selection. For example, if the Deployment Workbench incorrectly detects 32-bit and 64-bit device drivers, clear the x64 selection; the driver will then only be used for 32-bit deployments.
Platforms: x64	Select to control whether this device driver is for 64-bit operating system. If the check box is: - Selected, the device driver is available for deployment to 64-bit operating systems - Cleared, the device driver is unavailable for deployment to 64-bit operating systems If the Deployment Workbench incorrectly detects the platforms that the device driver supports, you can clear the platform selection. For example, if the Deployment Workbench incorrectly detects 32-bit and 64-bit device drivers, clear the x64 selection; the driver will then only be used for 32-bit deployments.

Setting	Description
Enable this driver	Select to control whether this device driver is available to other wizards and dialog boxes in the Deployment Workbench. If the check box is: - Selected, the device driver is available to other wizards and dialog boxes in the Deployment Workbench - Cleared, the device driver is unavailable to other wizards and dialog boxes in the Deployment Workbench The check box is selected by default.

The device driver configuration settings are saved, and the modifications are displayed in the details pane of the Deployment Workbench.

View the Device Driver Properties Details Tab

The device driver properties stored on the **Details** tab are configured when you run the Import Device Drivers Wizard. All the information on the **Details** tab is read only and cannot be modified. View the device driver properties on the **Details** tab through the ***driver_name*** **Properties** dialog box (where *driver_name* is the name of the device driver in the Deployment Workbench).

To view existing device drivers properties on the Details tab

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/Out-of-Box Drivers (where *deployment_share* is the name of the deployment share in which you will configure the device driver).
3. In the details pane, select ***driver_name*** (where *driver_name* is the name of the device driver you want to configure).
4. In the Actions pane, select **Properties**.

   The ***driver_name*** **Properties** dialog box opens (where *driver_name* is the name of the device driver you want to configure).
5. On the **Details** tab, view the settings listed in Table 51, and then select **OK**.

> **Note**
>
> The configuration settings on the **Details** tab are automatically determined by the Deployment Workbench and cannot be modified.

Table 51. Configuration Settings on the Details Tab of the Device Driver Properties

Expand table

Setting	Description
Manufacturer	Contains the name of the device driver manufacturer.
Version	Contains the version number of the device driver.
Driver date	Contains the date of the device driver.
Driver type (class)	Contains the class of the device driver, such as system or boot.
INF path	Contains the path to the device drive file relative to the root of the deployment share.
Supported OS versions	Contains a comma-separated list of the Windows operating system versions that the device driver supports.
Hash	Contains the encrypted hash of every file that is part of the driver package.
Supported PnP IDs	Contains a list of the plug-and-play IDs that the device driver supports.
This driver is WHQL signed	Select to indicate whether the device driver is signed by the Windows Hardware Quality Labs (WHQL). For device drivers that pass the WHQL tests, Microsoft creates a digitally signed certification file that allows installation on 64-bit versions of Windows and prevents 32-bit versions of Windows from displaying a warning message that the driver has not been certified by Microsoft.If the check box is: - Selected, the device driver has been signed by the WHQL - Cleared, the device driver has not been signed by the WHQL

Copy Device Drivers in the Deployment Workbench

You can copy and paste device drivers and folders beneath the Out-of-Box Drivers node in the Deployment Workbench using the **Copy** and **Paste** actions as described in Copy Items in the Deployment Workbench.

Move Device Drivers in the Deployment Workbench

You can move device drivers and folders beneath the Out-of-Box Drivers node in the Deployment Workbench using the **Cut** and **Paste** actions as described in Move Items in the Deployment Workbench.

Rename Device Drivers in the Deployment Workbench

You can rename device drivers and folders beneath the Out-of-Box Drivers node in the Deployment Workbench using the **Rename** action as described in Rename Items in the Deployment Workbench.

Delete Device Drivers from the Deployment Workbench

You can delete device drivers and folders beneath the Out-of-Box Drivers node in the Deployment Workbench using the Delete Selected Items Wizard as described in Delete Items from the Deployment Workbench. The Delete Selected Items Wizard allows you to delete individual packages or entire folder structures.

Manage Folders for Device Drivers in the Deployment Workbench

You can manage folders beneath the Out-of-Box Drivers node in the Deployment Workbench to create hierarchical groupings of device drivers. For more information on:

- Managing folders, see Manage Folders in the Deployment Workbench
- Selection profiles, see Manage Selection Profiles

Enable or Disable Device Drivers in the Deployment Workbench

You can control whether device drivers are available to other wizards and dialog boxes in the Deployment Workbench by selecting the **Enable this driver** check box on the **General** tab of the device driver **Properties** dialog box.

> ⊗ **Caution**

If you disable a device driver, the driver is never installed.

For more information on enabling or disabling device drivers in the Deployment Workbench, see Modify Existing Device Drivers in the Deployment Workbench.

Deploy Specific Device Drivers to Target Computers in LTI

By default, LTI deployments include all device drivers in Windows PE and deploy them to the target computers. Then, the target operating system uses Plug-and-Play IDs to identify the device drivers that are needed for the devices on the target computers.

To change this default behavior, configure the LTI deployment process to install specific drivers to target computers as described in Control Device Driver Deployments for LTI. For more information about strategies for device driver management, see Select the Device Driver Management Strategy.

Configuring Task Sequences in the Deployment Workbench

Task sequences in MDT contain the steps to be performed during LTI. Task sequences in MDT use the same task sequence engine as Configuration Manager; however, Configuration Manager is not required to perform LTI deployments. Use the Deployment Workbench to manage the task sequences used to perform deployments to the reference and target computers in your organization.

Configure task sequences in the Deployment Workbench in a deployment share's Packages node by:

- Creating a new task sequence as described in Create a New Task Sequence in the Deployment Workbench
- Modifying an existing task sequence as described in Modify an Existing Task Sequence in the Deployment Workbench
- Copying task sequences as described in Copy Task Sequences in the Deployment Workbench
- Moving task sequences as described in Move Task Sequences in the Deployment Workbench
- Renaming task sequences as described in Rename Task Sequences in the Deployment Workbench

- Deleting task sequences as described in Delete Task Sequences from the Deployment Workbench
- Managing folders for task sequences as described in Manage Folders for Task Sequences in the Deployment Workbench
- Enabling or disabling a task sequence as described in Enable or Disable a Task Sequence in the Deployment Workbench
- Preventing task sequences from being visible in the Deployment Wizard as described in Prevent a Task Sequence from Being Visible in the Deployment Wizard
- Modifying the unattended setup answer file for a task sequence as described in Modify the Unattended Setup Answer File Associated with the Task Sequence

  In addtion to managing task sequences in the Deployment Workbench, you can manage task sequences using the MDT Windows PowerShell cmdlets. For more information on managing task sequences using the MDT Windows PowerShell cmdlets, see the following sections beneath the section, "MDT Windows PowerShell Cmdlets", in the MDT document *Toolkit Reference*:

- **Import-MDTTaskSequence**
- **Get-MDTDeploymentShareStatistics**

Create a New Task Sequence in the Deployment Workbench

Using the New Task Sequence Wizard in the Deployment Workbench to create new task sequences. Start the New task Sequence Wizard using one of the following methods:

- In the console tree, select the Task Sequences node or a folder beneath the Task Sequences node, and then, in the Actions pane, select **New Task Sequence**.
- In the console tree, select the Task Sequences node or a folder beneath the Task Sequences node, and then, from the **Action** menu, select **New Task Sequence**.
- In the console tree, select the Task Sequences node or a folder beneath the Task Sequences node, and then select **New Task Sequence**.

  MDT includes task sequence templates that you can use for common deployment scenarios. In many instances, you can use the templates without any modification to the task sequence. However, you can modify task sequences created from the templates to meet the requirements of your organization.

Table 52 lists the task sequence templates in MDT.

Expand table

Template	Description
Sysprep and Capture	Performs a Sysprep operation and captures an image of a reference computer.
Standard Client Task Sequence	Creates the default task sequence for deploying operating system images to client computers, including desktop and portable computers
Standard Client Replace Task Sequence	Backs up the system entirely, backs up the user state, and wipes the disk
Custom Task Sequence	Creates a customized task sequence that does not install an operating system
Standard Server Task Sequence	Creates the default task sequence for deploying operating system images to server computers.
Litetouch OEM Task Sequence	Pre-loads operating systems images on computers in a staging environment prior to deploying the target computers in the production environment (typically by a computer OEM).
Post OS Installation Task Sequence	Performs installation tasks after the operating system has been deployed to the target computer
Deploy to VHD Client Task Sequence	Deploys client operating system images to a virtual hard disk (VHD) file on the target computer
Deploy to VHD Server Task Sequence	Deploys server operating system images to a VHD file on the target computer

> **Note**
>
> Select the **Litetouch OEM** task sequence only when performing deployments using removable media-based deployments you create in the Media node in the Deployment Workbench. Although you can select the Litetouch OEM Task Sequence template from other deployment shares, the task sequence will not finish successfully.

To create a new task sequence

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.

2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/Task Sequences (where *deployment_share* is the name of the deployment share to which you will add the task sequence).

3. In the Actions pane, select **New Task Sequence**.

   The New Task Sequence Wizard starts.

4. Complete the New Task Sequence Wizard using the information in Table 53.

Table 53. Information for Completing the New Task Sequence Wizard

Expand table

On this wizard page	Do this
General Settings	- In **Task sequence ID**, type ***task_sequence_id*** (where *task_sequence_id* is a unique identifier for the task sequence you are creating). Although you can change a task sequence's name and comments later, you cannot change a task sequence's ID. Before creating task sequences, create a naming scheme to use in creating task sequence IDs that will provide meaningful information about each task sequence. An example naming scheme is *Version-Edition-Level-Label*, where *Version* is the operating system version (Win8, Win2012), *Edition* is the operating system edition (Enterprise, Standard, Ultimate), *Level* is the service pack level (SP1, SP2), and *Label* is a descriptive label that identifies the customizations. - In **Task sequence name**, type ***task_sequence_name*** (where *task_sequence_name* is a descriptive name for the task sequence you are creating). - In **Task sequence comments**, type ***task_sequence_comment*** (where *task_sequence_comment* is text that describes the purpose or usage of the task sequence). - Select **Next**.
Select Template	In **The following task sequence templates are available. Select the one you would like to use as a starting point**, select ***task_sequence***, and then select **Next**.
Select OS	In **The following operating system images are available to be deployed with this task sequence. Select one to use**, select ***operating_system*** (where

On this wizard page	Do this
	operating_system is the operating system in the Operating Systems node in the Deployment Workbench that you want to deploy to the resource or target computer), and then select **Next**.
Specify Product Key	a. Select one of the following options based on the requirements of your organization: - **Do not specify a product key at this time**. Select this option when a product key is not required when deploying Windows; when the product key will be provided in the Deployment Wizard; or when using volume licenses that are activated using KMS. - **Specify a multiple activation key (MAK) for activating this operating system**. Select this option when deploying Windows using MAK product keys in the deployment. MAK product keys are used by Microsoft Volume Licensing customers. - **Specify the product key for this operating system**. Select this option when deploying retail product key. For more information about Volume Activation and product keys in MDT, see Volume Activation Overview. b. Select **Next**.
OS Settings	- In **Full Name**, type ***user_full_name*** (where *user_full_name* is the name of the user of the target computer). - In **Organization**, type ***organization_name*** (where *organization_name* is the name of the organization). - In **Internet Explorer Home Page**, type ***home_url*** (where *home_url* is the Uniform Resource Locator [URL] of the website to be the default site when starting Internet Explorer). - Select **Next**.
Admin Password	In **Administrator Password** and **Please confirm Administrator Password**, type ***password*** (where *password* is the password to be assigned to the built-in Administrator account on the reference or target computer), and then select **Next**.
Summary	Select **Next**.
Confirmation	Select **Finish**.

The New Task Sequence Wizard finishes. The package is added to the list of packages in the details pane of the Deployment Workbench.

Modify an Existing Task Sequence in the Deployment Workbench

Modify task sequences in the Task Sequences node in the Deployment Workbench using the **Properties** actions as described in View Item Properties in the Deployment Workbench. Configure task sequences in the Deployment Workbench by performing the following steps in the task sequence **Properties** dialog box:

1. Configure properties on the **General** tab as described in Configure the Task Sequence Properties General Tab.
2. Configure properties on the **Task Sequence** tab as described in Configure the Task Sequence Properties Task Sequence Tab.
3. Configure properties on the **OS Info** tab as described in Configure the Task Sequence Properties OS Info Tab.

Configure the Task Sequence Properties General Tab

The task sequence properties stored on the **General** tab are mostly configured when you run the New Task Sequence Wizard. Update the task sequence properties on the **General** tab through the ***task_sequence_name*** **Properties** dialog box (where *task_sequence_name* is the name of the task sequence in the Deployment Workbench).

To modify existing task sequence properties on the General tab

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/Task Sequences (where *deployment_share* is the name of the deployment share in which you will configure the task sequence).
3. In the details pane, select ***task_sequence_name*** (where *task_sequence_name* is the name of the task sequence you want to configure).
4. In the Actions pane, select **Properties**.

   The ***task_sequence_name*** **Properties** dialog box opens (where *task_sequence_name* is the name of the task sequence you want to configure).

5. On the **General** tab, configure the settings listed in Table 54 based on the requirements of your organization, and then select OK.

Table 54. Configuration Settings on the General Tab of Task Sequence Properties

Expand table

Setting	Description
Task sequence ID	Contains the task sequence identifier that the New Task Sequence Wizard provided. The information in this text box is automatically generated by Deployment Workbench and cannot be modified.
Task sequence name	Contains the name of the task sequence displayed in the Deployment Workbench and the Deployment Wizard.
Comments	Provides information about the task sequence.
Task sequence version	Contains the version number of the task sequence. You can type any version number that is appropriate for your organization's versioning standards.
This can run on any platform	Select to configure the task sequence to run on any supported 32-bit or 64-bit Windows operating system. The other available option is **This can run only on the specified client platforms**.
This can run only on the specified client platforms	Select to configure the task sequence to run on any supported 32-bit or 64-bit Windows operating system. The other available option is **This can run only any platform**.
Hide this task sequence in the Deployment Wizard	Select to control when this task sequence appears in the Deployment Wizard. If the check box is: - Selected, the Deployment Wizard will not display this task sequence - Cleared, the Deployment Wizard displays this task sequence This check box is cleared by default.
Enable this task sequence	Select to control when this task sequence is available to other wizards and dialog boxes in the Deployment Workbench. If the check box is: - Selected, other wizards and dialog boxes in the Deployment Workbench can select this task sequence

Setting	Description
	- Cleared, other wizards and dialog boxes in the Deployment Workbench cannot select this task sequence This check box is selected by default.

The task sequence configuration settings are saved, and the modifications are displayed in the details pane of the Deployment Workbench.

Configure the Task Sequence Properties Task Sequence Tab

The task sequence properties stored on the **Task Sequence** tab are mostly configured when you run the New Task Sequence Wizard. However, you can update the task sequence properties on the **Task Sequence** tab through the ***task_sequence_name* Properties** dialog box (where *task_sequence_name* is the name of the task sequence in the Deployment Workbench).

The **Task Sequence** tab contains areas and other controls that you use to:

- Configure steps and sequences as described in Configure the Task Sequence Steps and Step Sequence
- Configure step properties as described in Configure the Task Sequence Step Properties
- Configure step options as described in Configure the Task Sequence Step Options

Configure the Task Sequence Steps and Step Sequence

The **Task Sequence** tab contains a hierarchical representation of the task sequence steps and their sequence. Task sequence steps are organized into a hierarchical folder structure based on deployment phases.

You can organize one or more task sequence steps by creating a group. You can organize multiple groups and task sequence steps to create a hierarchy of groups and task sequence steps. You use task sequence step groups to control the processing of one or more task sequence steps as a unit.

Configure the task sequence steps and step sequence by selecting one of the following options from the menu bar at the top of the hierarchical representation:

- **Add**. Select to add a task sequence step group or step to the task sequence. The categories of task sequence steps that you can add are listed in Table 56 along

with the task sequence step types in each category. For more information about each of the task sequence step types listed in Table 55, see the corresponding section in the MDT document *Toolkit Reference*.

Table 55. Task Sequence Step Categories and Types

Expand table

Category	Task sequence step types in this category
General	- Run Command Line - Run PowerShell Script - Set Task Sequence Variable - Restart computer - Gather - Install Updates Offline - Validate - Install Application - Inject drivers - Execute Orchestrator Runbook
Disks	- Format and Partition Disk - Enable BitLocker - Create Virtual Hard Disk (VHD)
Images	- Install Operating System
Settings	- Apply Network Settings - Capture Network Settings - Recover From Domain
Roles	- Install Roles and Features - Uninstall Roles and Features

Category	Task sequence step types in this category
	- Configure DHCP - Configure DNS - Configure ADDS - Authorize DHCP

- **Remove**. Select to remove the currently highlighted task sequence step or group.

  > **Important**
  >
  > If you remove a task sequence group, you also remove all the task sequence steps in that group.

- **Up**. Select to configure a task sequence step to be processed earlier in the deployment process. The move is reflected in the task sequence hierarchy.

  > **Note**
  >
  > If you move the first task sequence step in a task sequence group up, the task sequence step will be performed before the entire group and will be removed from the group. If another task sequence group immediately precedes the group, the task sequence step will become the last step in the preceding group.

- **Down**. Select to configure a task sequence step to be processed earlier in the deployment process.

  > **Note**
  >
  > If you move the last task sequence step in a task sequence group down, the task sequence step will be performed after the entire group and will be removed from the group. If another task sequence group immediately follows the group, the task sequence step will become the first step in the following group.

To modify existing task sequence steps and step sequence

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.

2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/Task Sequences (where *deployment_share* is the name of the deployment share in which you will configure the task sequence).

3. In the details pane, select ***task_sequence_name*** (where *task_sequence_name* is the name of the task sequence you want to configure).

4. In the Actions pane, select **Properties**.

   The ***task_sequence_name*** **Properties** dialog box opens (where *task_sequence_name* is the name of the task sequence you want to configure).

5. On the **Task Sequence** tab, in the task sequence hierarchy, configure the task sequence steps and step sequences based on the requirements of your organization, and then select **OK**.

   The task sequence configuration settings are saved, and the modifications are displayed in the details pane of the Deployment Workbench.

   For more information about customizing task sequence steps for installing:

- Applications, see Customize Application Installation in Task Sequences
- Packages, see Customize Package Installation in Task Sequences

Configure the Task Sequence Step Properties

On the **Properties** tab, you configure the properties for task sequence groups or individual task sequence steps. The configuration settings for:

- Task sequence groups are the same for all groups
- Task sequence steps are different for each task sequence step type

  Table 56 lists the properties common to task sequence groups and steps. In addition to these properties, most task sequence steps have properties that are specific to the task sequence type.

Table 56. Properties Common to Task Sequence Groups and Steps

Expand table

Setting	Description
Type	Contains the task sequence type, which is always set to **Group** for task sequence groups or to the types listed in REF _Ref304894666 \h Table 55
Name	Contains the name of the task sequence group or step displayed in the task sequence hierarchy
Description	Provides descriptive information about the task sequence group or step

For more information about:

- Configuring specific task sequence step types, see the corresponding section in Configuring Task Sequences in the Deployment Workbench
- The properties for each task sequence type, see the section for corresponding task sequence steps in the MDT document *Toolkit Reference*

To modify existing task sequence group and individual step properties

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/Task Sequences (where *deployment_share* is the name of the deployment share in which you will configure the task sequence).
3. In the details pane, select ***task_sequence_name*** (where *task_sequence_name* is the name of the task sequence you want to configure).
4. In the Actions pane, select **Properties**.

   The ***task_sequence_name*** **Properties** dialog box opens (where *task_sequence_name* is the name of the task sequence you want to configure).
5. On the **Task Sequence** tab, in the task sequence hierarchy, select the **Properties** tab.
6. On the **Properties** tab, configure the task sequence group or individual step based on the requirements of your organization, and then select **OK**.

Configure the Task Sequence Step Options

On the **Options** tab, you configure settings that control how the task sequence step runs. These settings allow you to disable the step, specify the return codes for the step that indicate success, determine whether the step should continue in the event of an error, and any conditions for running the step.

The configuration settings on the **Options** tab for:

1. A task sequence group affect all the steps with the group
2. An individual task sequence step affect only that step

   For example, if you configure a condition for a task sequence group, that condition affects all the task sequence steps within the group.

To modify existing task sequence group and individual step options

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/Task Sequences (where *deployment_share* is the name of the deployment share in which you will configure the task sequence).
3. In the details pane, select ***task_sequence_name*** (where *task_sequence_name* is the name of the task sequence you want to configure).
4. In the Actions pane, select **Properties**.

   The ***task_sequence_name*** **Properties** dialog box opens (where *task_sequence_name* is the name of the task sequence you want to configure).
5. On the **Task Sequence** tab, in the task sequence hierarchy, select the **Options** tab.
6. On the **Options** tab, configure the task sequence group or individual step sequences based on the requirements of your organization, and then select **OK**.
7. On the **Task Sequence** tab, configure the settings listed in Table 57 based on the requirements of your organization, and then select **OK**.

Table 57. Configuration Settings on the Task Sequence Tab of Task Sequence Properties

Expand table

Setting	Description
Disable this step	Select to control whether the task sequence step runs during the task sequence. If the check box is: - Selected, the task sequence group or step is not run during the task sequence - Cleared, the task sequence group or step runs during the task sequence This check box is cleared by default.
Success codes	Contains the list of numeric codes that indicate whether the task sequence step finished successfully. Success codes are not available in task sequence step groups.
Continue on error	Select to control whether the task sequence should continue when the task sequence group or step encounters an error. If the check box is: - Selected, the task sequence continues if the group or step encounters an error - Cleared, the task sequence will not continue if the group or step encounters an error This check box is selected by default.
Condition list box	Contains any conditional criteria for running this step. If no criteria are specified, the step runs.Add criteria for determining when the group of tasks should (or should not) run. Use the **Add**, **Remove**, and **Edit** buttons to modify the conditions under which the group of tasks runs. The criteria can be based on: - An **IF** statement - A task sequence variable - The version of the target operating system. - A Windows Management Instrumentation (WMI) Query Language (WQL) query within a WMI namespace Any conditions configured for a group affect all the tasks within a group. For more information about conditions in task sequence steps, see Configure Task Sequence Step Conditions.

Configure the Task Sequence Properties OS Info Tab

The task sequence properties stored on the **OS Info** tab are mostly configured when you run the New Task Sequence Wizard. You update the task sequence properties on the **OS Info** tab through the ***task_sequence_name* Properties** dialog box (where *task_sequence_name* is the name of the task sequence in the Deployment Workbench).

To modify existing task sequence properties on the OS Info tab

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/Task Sequences (where *deployment_share* is the name of the deployment share in which you will configure the task sequence).
3. In the details pane, select ***task_sequence_name*** (where *task_sequence_name* is the name of the task sequence you want to configure).
4. In the Actions pane, select **Properties**.

   The ***task_sequence_name* Properties** dialog box opens (where *task_sequence_name* is the name of the task sequence you want to configure).
5. On the **OS Info** tab, configure the settings listed in Table 58 based on the requirements of your organization, and then select **OK**.

Table 58. Configuration Settings on the OS Info Tab of Task Sequence Properties

Expand table

Setting	Description
Operating system description	Contains the name of the operating system that you provided when creating the task sequence—for example Windows 8 ENTERPRISE. The information in this text box is automatically generated by the Deployment Workbench and cannot be modified.
Build	Contains the build number of the operating system. The information in this text box is automatically generated by the Deployment Workbench and cannot be modified.

Setting	Description
Platform	Contains the processor architecture of the operating system—for example, x86. The information in this text box is automatically generated by the Deployment Workbench and cannot be modified.
Edit Unattend.xml	Select to modify the contents of the Unattended.xml file that the Deployment Workbench generates for Windows. The Deployment Workbench cannot create catalog files for some Windows images of different architecture types. The following list describes the architecture types running the Deployment Workbench and catalogs that you can create for each architecture: - **Deployment Workbench running on x86**. Creates catalogs for x86 and x64 Windows images - **Deployment Workbench running on x64**. Creates catalogs only for x64 Windows images The Get Operating System Catalog Wizard may appear if an operating system does not yet have a catalog. You will see a progress bar in the Get Operating System Catalog Wizard, but no user interaction is required. The wizard may take a few minutes to finish.

The task sequence configuration settings are saved, and the modifications are displayed in the details pane of the Deployment Workbench.

Copy Task Sequences in the Deployment Workbench

You can copy and paste task sequences and folders beneath the Task Sequences node in the Deployment Workbench using the **Copy** and **Paste** actions as described in Copy Items in the Deployment Workbench.

Move Task Sequences in the Deployment Workbench

Move task sequences and folders beneath the Task Sequences node in the Deployment Workbench by using the **Cut** and **Paste** actions as described in Move Items in the Deployment Workbench.

Rename Task Sequences in the Deployment Workbench

Rename task sequences and folders beneath the Task Sequences node in the Deployment Workbench by using the **Rename** action as described in Rename Items in the Deployment Workbench.

Delete Task Sequences from the Deployment Workbench

Delete task sequences and folders beneath the Task Sequences node in the Deployment Workbench using the Delete Selected Items Wizard as described in Delete Items from the Deployment Workbench. The Delete Selected Items Wizard allows you to delete individual task sequences or entire folder structures.

Manage Folders for Task Sequences in the Deployment Workbench

You can manage folders beneath the Task Sequences node in the Deployment Workbench to create hierarchical groupings of task sequences. For more information on:

1. Managing folders, see Manage Folders in the Deployment Workbench
2. Selection profiles, see Manage Selection Profiles

Enable or Disable a Task Sequence in the Deployment Workbench

You can control whether task sequences are available to other wizards and dialog boxes in the Deployment Workbench using the **Enable this task sequence** check box on the **General** tab of the package **Properties** dialog box, as described in Configuring Task Sequences in the Deployment Workbench.

Prevent a Task Sequence from Being Visible in the Deployment Wizard

You can prevent a task sequence from being visible in the Deployment Wizard using the **Hide this task sequence in the Deployment Wizard** check box on the **General** tab of the application **Properties** dialog box, as described in Configuring Task Sequences in the Deployment Workbench.

Modify the Unattended Setup Answer File Associated with the Task Sequence

MDT automatically updates the unattended setup answer file (Unattend.xml) for a task sequence based on the configuration settings you provide in the Deployment Workbench and in the Deployment Wizard. However, there are instances in which you

may need to modify the unattended setup answer file for a task sequence directly, such as when you modify a configuration parameter that is not exposed in the Deployment Workbench or in the Deployment Wizard. Directly modify the unattended setup answer file for a task sequence by selecting **Edit Unattend.xml** on the **OS Info** tab of the task sequence **Properties** dialog box.

For more information about:

- Modifying the unattended setup answer file in the Deployment Workbench, see Configure the Task Sequence Properties OS Info Tab
- Unattend.xml, see the *Windows Assessment and Deployment Kit User's Guide* in the Windows ADK

Performing Common Management Tasks in the Deployment Workbench

You use the Deployment Workbench to perform many of the common management tasks. Although some management is unique to each type of item, the following tasks are common to all items in the Deployment Workbench:

- Managing folders as described in Manage Folders in the Deployment Workbench
- Viewing item properties as described in View Item Properties in the Deployment Workbench
- Copying items as described in Copy Items in the Deployment Workbench
- Moving items as described in Move Items in the Deployment Workbench
- Renaming items as described in Rename Items in the Deployment Workbench
- Deleting items as described in Delete Items from the Deployment Workbench

Manage Folders in the Deployment Workbench

You use folders to organize applications, operating systems, device drivers, and other items in the Deployment Workbench. Folders allow you to create hierarchies for organizing items as well as subsets of items that you can include in selection profiles.

> ⓘ **Note**

Folders are similar in concept to the groups that existed in previous versions of MDT, such as device driver groups.

Management tasks for folders include:

- Creating a new folder as described in Create a New Folder in the Deployment Workbench
- Modifying an existing folder as described in Modify an Existing Folder in the Deployment Workbench
- Copying a folder as described in Copy a Folder in the Deployment Workbench
- Moving a folder as described in Move a Folder in the Deployment Workbench
- Renaming a folder as described in Rename a Folder in the Deployment Workbench
- Deleting a folder as described in Delete a Folder from the Deployment Workbench
- Enabling or disabling a folder as described in Enable or Disable a Folder in the Deployment Workbench

Create a New Folder in the Deployment Workbench

Create folders in the Deployment Workbench using the New Folder Wizard. Start the New Folder Wizard using one of the following methods:

- In the console tree, select a node or a folder. Then, in the Actions pane, select **New Folder**.
- In the console tree, select a node or a folder. Then, from the **Action** menu, select **New Folder**.
- In the console tree, right-click a node or a folder. Then, select **New Folder**.

To create a new folder

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share/node_or_folder* (where *deployment_share* is the name of the deployment share in which you will create the

folder and *node_or_folder* is the name of the node or folder in which you will create the folder).

3. In the Actions pane, select **New Folder**.

   The New Folder Wizard starts.

4. Complete the New Folder Wizard using the information in Table 59.

Table 59. Information for Completing the New Folder Wizard

Expand table

On this wizard page	Do this
General Settings	a. In **Folder name**, type ***folder_name*** (where *folder_name* is the name of the folder you want to create). b. In **Folder comment**, type ***folder_comment*** (where *folder_comment* is text that describes the user of the folder in the deployment share). c. Select or clear the **Enable this folder** check box based on the needs of your organization. If this check box is: - Selected, the folder, subfolders, and content can be included in selection profiles - Cleared, the folder, subfolders, and content cannot be included in selection profiles d. Select **Next**.
Summary	Review the information, then select **Next**.
Confirmation	You can select **Save Output** to save the output of the wizard to a file. You can also select **View Script** to view the Windows PowerShell scripts used to perform the wizard tasks. Select **Finish**.

After the New Folder Wizard finishes, the new folder appears in the deployment share in the Deployment Workbench.

Modify an Existing Folder in the Deployment Workbench

Modify existing folders in the Deployment Workbench using the **Properties** actions as described in View Item Properties in the Deployment Workbench. The folder properties are mostly configured when you run the New Folder Wizard. Update the folder properties on the **General** tab through the ***folder_name*** **Properties** dialog box (where *folder_name* is the name of the folder in the Deployment Workbench).

To modify an existing folder

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/*node_or_folder* (where *deployment_share* is the name of the deployment share in which you will modify the folder and *node_or_folder* is the name of the node or folder that contains the folder to be modified).
3. In the details pane, select ***folder_name*** (where *folder_name* is the name of the folder you want to modify).
4. In the Actions pane, select **Properties**.

   The ***folder_name*** **Properties** dialog box opens (where *folder_name* is the name of the folder to be modified).
5. On the **General** tab, configure the settings listed in Table 60 based on the requirements of your organization, and then select **OK**.

Table 60. Configuration Settings on the General Tab of Folder Properties

Expand table

Setting	Description
Name	Contains the name of the folder that is displayed in the Deployment Workbench.
Comments	Provides information about the folder.
Enable this folder	Select to enable or disable the folder. If the check box is: - Selected, you can select the folder in selection profiles

Setting	Description
	- Cleared, you cannot select the folder in selection profiles

The folder configuration settings are saved, and the modifications are displayed in the details pane of the Deployment Workbench.

Copy a Folder in the Deployment Workbench

You can copy and paste folders in the Deployment Workbench using the **Copy** and **Paste** actions as described in Copy Items in the Deployment Workbench.

Move a Folder in the Deployment Workbench

You can move folders in the Deployment Workbench using the **Cut** and **Paste** actions as described in Move Items in the Deployment Workbench.

Rename a Folder in the Deployment Workbench

You can rename folders in Deployment Workbench using the **Rename** action as described in Rename Items in the Deployment Workbench.

Delete a Folder from the Deployment Workbench

You can delete a folder in the Deployment Workbench using the Delete Selected Items Wizard as described in Delete Items from the Deployment Workbench. The Delete Selected Items Wizard allows you to delete individual folders or an entire hierarchy of folders.

Enable or Disable a Folder in the Deployment Workbench

You can control whether folders are available to other wizards and dialog boxes in the Deployment Workbench using the **Enable this folder** check box on the **General** tab of the folder **Properties** dialog box. For more information on enabling or disabling folders in the Deployment Workbench, see Modify an Existing Folder in the Deployment Workbench.

View Item Properties in the Deployment Workbench

You can view the properties of operating systems, device drivers, and other items from the Deployment Workbench using one of the following methods:

- In the details pane, select an item. Then, in the Actions pane, select **Properties**.
- In the details pane, select an item. Then, from the **Action** menu, select **Properties**.
- In the details pane, right-click an item, and then select **Properties**.
- In the details pane, double-click an item.

  To set the properties of an item in a deployment share

To set the properties of an item in a deployment share

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/*item_type* (where *deployment_share* is the name of the deployment share in which you will copy the item and *item_type* is the type of item you will copy, such as an operating system or device driver).
3. In the details pane, select ***item*** (where *item* is the name of the item to be renamed).
4. In the Actions pane, select **Properties**.

   The ***item*** **Properties** dialog box is displayed (where *item* is the name of the item you selected).

Copy Items in the Deployment Workbench

Use the Deployment Workbench to copy operating systems, device drivers, and other items within a deployment share or between two deployments shares. When you copy an item, the Deployment Workbench creates a link to the original item instead of creating a separate copy of the item. This reduces the size of the deployment share. If you want to create a duplicate of an item, import the item again in the target folder.

When you copy an item between deployment shares and an item with the same:

- GUID already exists in the target deployment share, the configuration settings for the source item will be applied to the target item, including the name (if the items do not already have the same name)

- Name already exists in the target deployment share, an error is generated, because two items of the same type cannot have the same name

  You can copy items by using:

1. **Cut** and **Paste** actions as described in Copy Items Using the Cut and Paste Actions
2. Drag-and-drop functionality as described in Copy Items Using Drag-and-Drop Functionality

Copy Items Using the Cut and Paste Actions

You can copy an item using the **Cut** and **Paste** actions in the Deployment Workbench. Copy the item from the source location using one of the following methods:

- In the details pane, select an item. Then, in the Actions pane, select **Copy**.
- In the details pane, select an item. Then, from the **Action** menu, select **Copy**.
- In the details pane, right-click an item, and then select **Copy**.

  Paste the item that you have copied using one of the following methods:

- In the details pane, select the target location. Then, in the Actions pane, select **Paste**.
- In the details pane, select the target location. Then, from the **Action** menu, select **Paste**.
- In the details pane, right-click the target location, and then select **Paste**.

To copy and paste items in a deployment share

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/*item_type* (where *deployment_share* is the name of the deployment share in which you will copy the item and *item_type* is the type of item you will copy, such as an operating system or device driver).
3. In the details pane, select ***item*** (where *item* is the name of the item to be copied).
4. In the Actions pane, select **Copy**.

5. In the details pane, go to ***target_folder*** (where *target_folder* is the name of the folder where you want to copy the item).

6. In the Actions pane, select **Paste**.

   The new copy of the item appears in the details pane of the Deployment Workbench.

Copy Items Using Drag-and-Drop Functionality

You can copy items by dragging an item from the source location to the target location.

To copy items in the Deployment Workbench using drag-and-drop functionality

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.

2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share/item_type* (where *deployment_share* is the name of the deployment share to which you will copy the item and *item_type* is the type of item you will copy, such as an operating system or device driver).

3. In the details pane, drag ***item*** (where *item* is the name of the item to be copied) to the target location, press CTRL, and then release the mouse button.

   The item is copied to the target location in the details pane of the Deployment Workbench.

Move Items in the Deployment Workbench

Use the Deployment Workbench to move operating systems, device drivers, and other items within a deployment share or between two deployments shares. You can move items by using:

- **Cut** and **Paste** actions as described in Move Items Using the Cut and Paste Actions
- Drag-and-drop functionality as described in Move Items Using Drag-and-Drop Functionality

Move Items Using the Cut and Paste Actions

You can move an item using the **Cut** and **Paste** in the Deployment Workbench. Cut the item from the source location using one of the following methods:

- In the details pane, select an item. Then, in the Actions pane, select **Properties**.
- In the details pane, select an item. Then, from the **Action** menu, select **Properties**.
- In the details pane, right-click an item, and then select **Properties**.
- In the details pane, double-click an item.

To set the properties of an item in a deployment share

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/*item_type* (where *deployment_share* is the name of the deployment share in which you will copy the item and *item_type* is the type of item you will copy, such as an operating system or device driver).
3. In the details pane, select item (where *item* is the name of the item to be renamed).
4. In the Actions pane, select **Properties**.

   The **item Properties** dialog box is displayed (where *item* is the name of the item you selected).

Move Items Using Drag-and-Drop Functionality

You can move items by dragging them from the source location to the target location.

To move items in the Deployment Workbench using drag-and-drop functionality

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/*item_type* (where *deployment_share* is the name of the deployment share to which you will move the item and *item_type* is the type of item you will move, such as an operating system or device driver).

3. In the details pane, drag ***item*** (where *item* is the name of the item to be moved) to the target location.

   The item is moved to the target location.

Rename Items in the Deployment Workbench

You can rename operating systems, device drivers, and other items in the Deployment Workbench by using one of the following methods:

- In the details pane, select an item. Then, in the Actions pane, select **Rename**.
- In the details pane, select an item. Then, from the **Action** menu, select **Rename**.
- In the details pane, right-click an item, and then select **Rename**.

  Because the **Copy** and **Paste** actions in the Deployment Workbench create a link to the original item rather than a separate copy. So, when you rename an item, the Deployment Workbench automatically renames any copies of an item in other folders.

To rename an item in a deployment share

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/*item_type* (where *deployment_share* is the name of the deployment share in which you will rename the item and *item_type* is the type of item you will rename, such as an operating system or device driver).
3. In the details pane, select ***item*** (where *item* is the name of the item to be renamed). Then, in the Actions pane, select **Rename**.
4. In the details pane, type ***new_item_name*** (where *new_item_name* is the new name of the item), and then press ENTER.

   The new name of the item appears in the details pane of the Deployment Workbench.

Delete Items from the Deployment Workbench

You can delete operating systems, device drivers, and other items from the Deployment Workbench using the Delete Selected Items Wizard. Start the Delete Selected Items Wizard using one of the following methods:

- In the details pane, select an item. Then, in the Actions pane, select **Delete**.
- In the details pane, select an item. Then, from the **Action** menu, select **Delete**.
- In the details pane, right-click an item, and then select **Delete**.

  You can delete individual items or folders that contain one or more items or subfolders. You can also delete items that have copies in multiple folders. The Delete Selected Items Wizard's **Options** page includes the check boxes shown in Table 61.

Table 61. Information for Completing the Delete Selected Items Wizard

Expand table

Check box	Description
Completely delete these items, even if there are copies in other folders	Select to delete an item, including all copies of an item that might exist in other folders. If this check box is: - Selected, the selected item and all copies in other folders are deleted - Cleared, only the selected item is deleted; all copies in other folders are unaffected
Recursively delete the contents of folders, as well as multiple items that have the same source file	This check box allows you to delete: - Not only the immediate contents of a folder but also the content from subfolders - Multiple items that have the same source file—for example, if you have an operating system image file that contains multiple operating system editions, such as Server-Core or Server-Enterprise If this check box is: - Selected and the selected item is a folder, then the folder, subfolders, and all the contents of all subfolders are deleted - Selected and the selected item not a folder, then the item

Check box	Description
	and all items that have the same source file are deleted - Cleared, only the selected item is deleted; all subfolders or other items that have the same source file are unaffected

> **Note**
>
> When you delete an item from the Deployment Workbench, the corresponding file or folder is also deleted in the *deployment_share**item_type**item_subfolder* (where *deployment_share* is the name of the deployment share and *item_type* is the type of item you are deleting, such as an operating system or device driver) if no remaining items reference the folder.

To delete an item from a deployment share

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/*item_type* (where *deployment_share* is the name of the deployment share to which you will add the operating system and *item_type* is the type of item you are deleting, such as an operating system or device driver).
3. In the details pane, select ***item*** (where *item* is the name of the item to be deleted, such as a folder, an operating system, or a device driver).
4. In the Actions pane, select **Delete**.

   The Delete Selected Items Wizard starts.
5. Complete the Delete Selected Items Wizard using the information in Table 62.

Table 62. Information for Completing the Delete Selected Items Wizard

Expand table

On this wizard page	Do this
Options	- Select or clear the **Completely delete these items, even if there are copies in other folders** check box based on your requirements. By default, this check box is cleared.
Summary	Select Next.
Confirmation	You can select **Save Output** to save the output of the wizard to a file. You can also select **View Script** to view the Windows PowerShell scripts used to perform the wizard tasks. Select **Finish**.

After the Delete Selected Items Wizard finishes, the item and other affected items are removed from the Deployment Workbench and from the deployment share.

Performing Advanced Configuration Tasks in the Deployment Workbench

The Deployment Workbench includes advanced configuration options that extend the features provided in basic LTI deployments. These configuration options provide more granular selection of the content you want to include in the deployment, support deployments in larger organizations, and support deployments from stand-alone media without the need to connect to a deployment share.

Advanced configuration tasks that you can perform include:

- Managing selection profiles as described in Manage Selection Profiles
- Managing linked deployment shares as described in Manage Linked Deployment Shares
- Managing deployment media as described in Manage LTI Deployment Media
- Managing the MDT DB as described in Manage the MDT DB

Manage Selection Profiles

Selection profiles allow you to select one or more folders in the Deployment Workbench that contain one or more items in the Deployment Workbench, including applications, device drivers, operating systems, operating system packages, and task sequences.

se selection profiles to group items, and then use those groupings of items:

- To include the appropriate device drivers and packages for Windows PE.
- To include the appropriate device drivers for the target operating system in the **Inject Drivers** task sequence step type.
- To identify the operating system packages to deploy in the **Install Updates Offline** task sequence step type.
- As the basis for creating linked deployment shares.
- As the basis for creating MDT deployment media.

Table 63 lists the default selection profiles in the Deployment Workbench.

Table 63. Default Selection Profiles in Deployment Workbench

Expand table

Selection profile	Description
Everything	Holds all folders from all nodes in the Deployment Workbench, including all applications, operating systems, device drivers, operating system packages, and task sequences.
All Drivers	Holds all folders from the Out-of-Box Drivers node in the Deployment Workbench, including all device drivers.
All Drivers and Packages	Holds all folders from the Applications and Out-of-Box Drivers nodes in the Deployment Workbench, including all applications and device drivers.
All Packages	Holds all folders from the Applications node in the Deployment Workbench, including all applications and device drivers.
Nothing	Includes no folders or items in the Deployment Workbench.
Sample	A sample selection profile that shows how to select a subset of the items and include all folders from the Packages and Task Sequences nodes in the Deployment Workbench. This selection profile includes all operating system packages and task sequences.

Manage selection profiles by completing the following tasks in the Deployment Workbench:

- Create a new selection profile as described in Create a New Selection Profile in the Deployment Workbench.
- Modify an existing selection profile as described in Modify an Existing Selection Profile in the Deployment Workbench.
- Copy a selection profile as described in Copy a Selection Profile in the Deployment Workbench.
- Move a selection profile as described in Move a Selection Profile in the Deployment Workbench.
- Rename a selection profile as described in Rename a Selection Profile in the Deployment Workbench.
- Delete a selection profile as described in Delete a Selection Profile from the Deployment Workbench.
- Identify the differences between selection provides and groups as described in Identify the Relationship Between Selection Profiles and Groups.

Create a New Selection Profile in the Deployment Workbench

Create selection profiles in the Deployment Workbench using the New Selection Profile Wizard. Start the New Selection Profile Wizard using one of the following methods:

- In the console tree, select the Selection Profiles node. Then, in the Actions pane, select **New Selection Profile**.
- In the console tree, select the Selection Profiles node. Then, from the **Action** menu, select **New Selection Profile**.
- In the console tree, right-click the Selection Profiles node, and then select **New Selection Profile**.

To create a new selection profile

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/Advanced Configuration/Selection Profiles (where *deployment_share* is the name of the deployment share to which you will add the application).

3. In the Actions pane, select **New Selection Profile**.

   The New Selection Profile Wizard starts.

4. Complete the New Selection Profile Wizard using the information in Table 64.

Table 64. Information for Completing the New Selection Profile Wizard

Expand table

On this wizard page	Do this
General Settings	- In **Selection profile name**, type ***profile_name*** (where *profile_name* is the name of the new profile). - In **Selection profile comments**, type ***comment*** (where *comment* is descriptive text that provides information about the selection profile). - Select Next.
Folders	In **Select the folders that should be included in this selection profile**, select ***folders*** (where *folders* is the name of the folders that contain the Deployment Workbench items you want to include in this selection profile), and then select **Next**.
Summary	Review in the information in **Details**, and then select **Next**.
Confirmation	You can select **Save Output** to save the output of the wizard to a file. You can also select **View Script** to view the Windows PowerShell scripts used to perform the wizard tasks. Select **Finish**.

   The New Selection Profile Wizard finishes. The selection profile is added to the list of selection profiles in the details pane of the Deployment Workbench.

Modify an Existing Selection Profile in the Deployment Workbench

Modify existing selection profiles in the Deployment Workbench's Selection Profiles node using the **Properties** actions as described in View Item Properties in the Deployment Workbench. The selection profile properties are mostly configured when you run the New Selection Profile Wizard. However, you can update the selection profile

properties on the **General** tab of the ***profile_name*** **Properties** dialog box (where *profile_name* is the name of the selection profile in the Deployment Workbench).

To configure the General tab for package properties

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/Advanced Configuration/Selection Profiles (where *deployment_share* is the name of the deployment share where you will configure the package).
3. In the details pane, select ***profile_name*** (where *profile_name* is the name of the selection profile you want to configure).
4. In the Actions pane, select **Properties**.

   The ***profile_name*** **Properties** dialog box opens (where *profile_name* is the name of the selection profile you want to configure).
5. On the **General** tab, configure the settings listed in Table 65 based on the requirements of your organization, and then select **OK**.

Table 65. Configuration Settings on the General Tab of Package Properties

Expand table

Setting	Description
Name	Contains the name of the selection displayed in the Deployment Workbench and the Deployment Wizard.
Comments	Provides information about the selection profile.
Folders	Hierarchical list of the folders and their selection status.

The selection profile configuration settings are saved, the modifications are displayed in the details pane of the Deployment Workbench, and the *deployment_share*\Control\SelectionProfiles.xml file (where *deployment_share* is the name of the deployment share) is updated with the selection profile configuration settings.

Copy a Selection Profile in the Deployment Workbench

You can copy and paste selection profiles in the Deployment Workbench using the **Copy** and **Paste** actions as described in Copy Items in the Deployment Workbench.

Move a Selection Profile in the Deployment Workbench

You can move selection profiles in the Deployment Workbench using the **Cut** and **Paste** actions as described in Move Items in the Deployment Workbench.

Rename a Selection Profile in the Deployment Workbench

You can rename selection profiles in the Deployment Workbench using the **Rename** action as described in Rename Items in the Deployment Workbench.

Delete a Selection Profile from the Deployment Workbench

You can delete a selection profile in the Deployment Workbench using the Delete Selected Items Wizard as described in Delete Items from the Deployment Workbench. The Delete Selected Items Wizard allows you to delete individual selection profiles.

Identify the Relationship Between Selection Profiles and Groups

Use selection profiles to create groups of Deployment Workbench items, such as operating systems, device drivers, or applications. Use the selection profiles to specify device drivers, define content to include in a linked deployment share, define the content to include for media deployments, and other tasks.

The relationship between items and folders in a selection profile is stored in the following files in the *deployment_share*\Control folder (where *deployment_share* is the location of the deployment share):

- **itemGroups.xml**. There is a separate file for each type of item, including:
  - ApplicationGroups.xml
  - DriverGroups.xml
  - LinkedDeploymentShareGroups.xml
  - MediaGroups.xml
  - OperatingSystemGroups.xml

- PackageGroups.xml
- SelectionProfileGroups.xml
- TaskSequenceGroups.xml

  For example, consider a selection profile for device drivers called *WinPEAndFullOS* that are stored in a folder created immediately beneath the Out-of-Box Drivers node. The following code is an excerpt from the DriverGroups.xml file generated when you created the selection profile:

```
XML

<groups>
...
<group_quid="{e5143c1c-24e4-466d-9b56-b0db693c8619}" enable="True">
   <Name>WinPEAndFullOS</Name>
...
   <Member>{1eca45a5-d7ef-475a-bb0d-7f7747f16b3a}</Member>
```

- **SelectionProfiles.xml**. This file contains the definitions for all the selection profiles defined for the deployment share. The following code is an excerpt from the SelectionProfile.xml file generated when you created the WinPEAndFullOS selection profile:

```
XML

<selectionProfile quid="{46a3e6a2-694c-4c2f-afd8-a2986e6e252e}"
enable="True">
  <Name>Drivers Safe For WinPE</Name>
  <Comments>Include Driver packages safe for WinPE.</Comments>
  <ReadOnly>True</ReadOnly>
  <Definition><SelectionProfile><Include path="Out-of-Box
Drivers\WinPEAndFullOS" /><Include path="Out-of-Box Drivers\WinPEOnly"
/></SelectionProfile></Definition>
</selectionProfile>
```

  By default, if you do not specify a selection profile or group in the CustomSettings.ini file or in the MDT DB, LTI uses all items. If you specify both selection profiles and groups in the CustomSettings.ini file or the MDT DB, LTI uses all the items from both the selection profile and the group.

  For example, if you specify a selection profile and use the default group (which includes all items), the end result is that LTI uses all items, because the default group includes all items, regardless of what you specify in the selection profile. To

restrict the items to a selection profile, specify a group that contains no items (that is, is empty). The reverse is true if you want to use a group.

Because of the introduction of folders in MDT, groups include all folders and subfolders by default. You can override this behavior using the **SkipGroupSubFolders**property. For more information on this property, see the corresponding section in the MDT document *Toolkit Reference*.

In most instances, you can use selection profiles and groups to perform most deployments. However, the following properties are available for more advanced scenarios—such as if you want to exclude a parent folder but include a child folder:

- **CustomDriverSelectionProfile**
- **CustomPackageSelectionProfile**
- **CustomWizardSelectionProfile**

  For more information on these properties, see the corresponding sections in the MDT document *Toolkit Reference*.

Manage Linked Deployment Shares

Linked deployment shares in MDT allow you to provide a logical connection between two deployment shares: a source and a target deployment share. A selection profile determines the items to be linked. When creating the link between the deployment shares, you can choose whether to merge or replace content in the target deployment share.

Using linked deployment shares, you can easily replicate an entire deployment share or portions of a deployment share to another deployment share. In this way, you can make changes to one deployment share, and then easily update other deployment shares based on the selection profiles you chose when creating the linked deployment shares.

Manage linked deployment shares by performing the following tasks in the Deployment Workbench:

- Create a new linked deployment share as described in Create a New Linked Deployment Share in the Deployment Workbench.
- Modify an existing linked deployment share as described in Modify an Existing Linked Deployment Share in the Deployment Workbench.
- Copy a linked deployment share as described in Copy a Linked Deployment Share in the Deployment Workbench.

- Move a linked deployment share as described in Move a Linked Deployment Share in the Deployment Workbench.
- Rename a linked deployment share as described in Rename a Linked Deployment Share in the Deployment Workbench.
- Delete a linked deployment share as described in Delete a Linked Deployment Share from the Deployment Workbench.
- Replicate linked deployment shares as described in Replicate Linked Deployment Shares in the Deployment Workbench.

  In addtion to managing linked deployement shares in the Deployment Workbench, you can manage linked deployment shares using the MDT Windows PowerShell cmdlets. For more information on managing linked deployment shares using the MDT Windows PowerShell cmdlets, see the following sections beneath the section, "MDT Windows PowerShell Cmdlets", in the MDT document *Toolkit Reference*:

- **Update-MDTLinkedDS**
- **Get-MDTDeploymentShareStatistics**

Create a New Linked Deployment Share in the Deployment Workbench

Create new linked deployment shares in the Deployment Workbench using the New Linked Deployment Share Wizard. Start the New Linked Deployment Share Wizard using one of the following methods:

- In the console tree, select the Linked Deployment Share node. Then, in the Actions pane, select **New Linked Deployment Share**.
- In the console tree, select the Linked Deployment Share node. Then, from the **Action** menu, select **New Linked Deployment Share**.
- In the console tree, right-click the Linked Deployment Share node, and then select **New Linked Deployment Share**.

To create a new linked deployment share

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.

2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/Advanced Configuration/Linked Deployment Share (where *deployment_share* is the name of the deployment share you want to configure).

3. In the Actions pane, select **New Linked Deployment Share**.

   The New Linked Deployment Share Wizard starts.

4. Complete the New Linked Deployment Share Wizard using the information in Table 66.

Table 66. Information for Completing the New Linked Deployment Share Wizard

Expand table

On this wizard page	Do this
General Settings	a. In **Linked deployment share UNC path**, type ***unc_path*** (where *unc_path* is the fully qualified UNC path to the target deployment share). You can alternatively select Browse to find the network shared folder. b. In **Comments**, type ***comment*** (where *comment* is descriptive text that provides information about the linked deployment share). c. In **Selection profile**, select ***profile*** (where *profile* is the name of the selection profile that will be used to establish the items to be linked between the source and target deployment shares). d. Under **Selection profile**, select one of the following options based on your requirements: - **Merge the selected content into the target deployment share**. Select to configure the wizard to copy the content in the selection profile into an existing target deployment share without deleting or overwriting any folders or items in the target deployment share. Selecting this option also copies the standard folders from the source deployment share, including the Scripts, Tools, USMT, and OEM folders. - **Replace the contents of the target deployment share folders with those selected**. Select to configure the wizard to copy the content in the selection profile into an existing target deployment share and overwrite any existing folders or items in the target deployment share. Selecting this option also copies the standard folders from the source deployment share,

On this wizard page	Do this
	including the Scripts, Tools, USMT, and OEM folders. By default, the **Merge the selected content into the target deployment share** option is selected. e. Select **Next**.
Summary	Review in the information in **Details**, and then select **Next**.
Confirmation	You can select **Save Output** to save the output of the wizard to a file. You can also select **View Script** to view the Windows PowerShell scripts used to perform the wizard tasks. Select **Finish**.

The New Linked Deployment Share Wizard finishes, and the linked deployment share is added to the list of linked deployments shares in the details pane of the Deployment Workbench.

Modify an Existing Linked Deployment Share in the Deployment Workbench

Modify existing linked deployment share in the Deployment Workbench's Linked Deployment Shares node using the **Properties** actions as described in View Item Properties in the Deployment Workbench. The linked deployment share properties are configured when you run the New Linked Deployment Share Wizard. However, you can update the linked deployment share properties on the **General** tab of the ***linked_deployment_share*** **Properties** dialog box (where *linked_deployment_share* is the name of the linked deployment share in the Deployment Workbench).

To modify an existing linked deployment share

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/Advanced Configuration/Linked Deployment Share (where *deployment_share* is the name of the deployment share in which you will configure the package).
3. In the details pane, select ***linked_deployment_share*** (where *linked_deployment_share* is the name of the linked deployment share you want to

configure).

4. In the Actions pane, select **Properties**.

   The ***linked_deployment_share*** **Properties** dialog box opens (where *linked_deployment_share* is the name of the selection profile you want to configure).

5. On the **General** tab, configure the settings listed in Table 67 based on the requirements of your organization, and then select **OK**.

Table 67. Configuration Settings on the General Tab of Linked Deployment Share Properties

Expand table

Setting	Description
Link identifier	Contains the identifier of the linked deployment share. The identifier in this text box is automatically generated by the Deployment Workbench and cannot be modified.
Comments	Provides information about the linked deployment share.
Linked deployment share UNC path	Contains the fully qualified UNC path to the target deployment share.
Choose a selection profile	Contains the selection profile that identifies the content to be replicated between the source and target deployment shares.
Merge the selected content into the target deployment share	Select to configure the wizard to copy the content in the selection profile into an existing target deployment share without deleting or overwriting any folders or items in the target deployment share. Selecting this option also copies the standard folders from the source deployment share, including the Scripts, Tools, USMT, and OEM folders.
Replace the contents of the target deployment share folders with those selected	Select to configure the wizard to copy the content in the selection profile into an existing target deployment share and overwrite any existing folders or items in the target deployment share. Selecting this option also copies the standard folders from the source deployment share, including the Scripts, Tools, USMT, and OEM folders.
Copy standard folders (Scripts, Tools, USMT,	Select to configure the Replicate to Linked Deployment Share Wizard to share. If this check box is:

Setting	Description
OEM) to this linked deployment share	- Selected, the standard folders are copied to the linked deployment share - Cleared, the standard folders are not copied to the linked deployment share This check box is cleared by default.
Automatically update boot images after replicating content to this linked deployment share	Select to configure the Replicate to Linked Deployment Share Wizard to automatically update any boot images in the linked deployment share after the content is replicated from the source deployment share. If this check box is: - Selected, the boot images in the linked deployment share are automatically updated when replication is complete - Cleared, the boot images in the linked deployment share are not automatically updated when replication is complete This check box is cleared by default. By default, the linked deployment share is configured to generate 32-bit and 64-bit boot images. Open the linked deployment share in the Deployment Workbench to change this default behavior as described in Open an Existing Deployment Share in the Deployment Workbench.
Access the linked deployment share in single-user mode in order to improve replication performance	Select to configure the Replicate to Linked Deployment Share Wizard to open the linked deployment share in *single-user mode* while replicating the content to the linked deployment share. Single-user mode improves replication performance, If this check box is: - Selected, the linked deployment share is opened in single-user mode as replication is performed and replication performance is improved If you select this check box, changes that other users make in the linked deployment share may be overwritten and lost during the replication process. - Cleared, the linked deployment share is not opened in single-user mode as replication is performed and replication performance is unimproved This check box is cleared by default.

The linked deployment share configuration settings are saved. The modifications are displayed in the details pane in the Deployment Workbench.

Copy a Linked Deployment Share in the Deployment Workbench

You can copy and paste linked deployment shares in the Deployment Workbench using the **Copy** and **Paste** actions as described in Copy Items in the Deployment Workbench.

Move a Linked Deployment Share in the Deployment Workbench

You can move linked deployment shares in the Deployment Workbench using the **Cut** and **Paste** actions as described in Move Items in the Deployment Workbench.

Rename a Linked Deployment Share in the Deployment Workbench

You can rename linked deployment shares in the Deployment Workbench using the **Rename** action as described in Rename Items in the Deployment Workbench.

Delete a Linked Deployment Share from the Deployment Workbench

You can delete a linked deployment shares in the Deployment Workbench using the Delete Selected Items Wizard as described in Delete Items from the Deployment Workbench. The Delete Selected Items Wizard allows you to delete individual linked deployment shares.

Replicate Linked Deployment Shares in the Deployment Workbench

You can replicate the content from the source deployment share to the linked deployment shares in the Deployment Workbench using the Replicate to Linked Deployments Share Wizard. Ensure that sufficient storage exists for the linked deployment share prior to running the Replicate to Linked Deployments Share Wizard, as the wizard does not verify that sufficient storage exists prior to replicating the content.

> **Note**
>
> By default, the linked deployment share is configured to generate 32-bit and 64-bit boot images. Open the linked deployment share in the Deployment Workbench to

change this default behavior as described in Open an Existing Deployment Share in the Deployment Workbench.

To replicate content to a linked deployment share

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/Advanced Configuration/Linked Deployment Share (where *deployment_share* is the name of the deployment share where you will add the application).
3. In the details pane, select ***linked_deployment_share*** (where *linked_deployment_share* is the name of the linked deployment share you want to configure).
4. In the Actions pane, select **Replicate Content**.

   The Replicate to Linked Deployment Share Wizard starts. The replication process starts automatically and is displayed on the **Progress** wizard page.
5. Complete the Replicate to Linked Deployment Share Wizard using the information in Table 68.

Table 68. Information for Completing the Replicate to Linked Deployment Share Wizard

Expand table

On this wizard page	Do this
Progress	View the progress of the replication process.
Confirmation	You can select **Save Output** to save the output of the wizard to a file. You can also select **View Script** to view the Windows PowerShell scripts used to perform the wizard tasks. Select **Finish**.

> **Note**

If you view the output of the wizard, the replication appears to have occurred twice. However, the replication is actually performed in two passes: The first pass copies new items into the linked deployment share, and the second pass deletes any items that are no longer needed in the linked deployment share.

The Replicate to Linked Deployments Share wizard finishes. The folders and the content you specified in the selection profile in the linked deployment share are replicated from the source deployment share to the target deployment share. Depending on the configuration of the linked deployment share, the folders and content on the target deployment share are merged or replaced.

Manage LTI Deployment Media

Media in LTI allows you to perform LTI deployments solely from local media, without connecting to a deployment share. You can store the media on a DVD, USB hard disk, or other portable device. After you create the media, generate bootable WIM images that allow the deployment to be performed from portable media devices locally available on the target computer.

You determine the items to be included on the media in a selection profile you specify when you create the media. The Deployment Workbench automatically includes Windows PE in the media WIM image so that Windows PE is started from the media available to the target computer. When Windows PE starts, the Deployment Wizard is automatically started, as well.

Manage deployment media by performing the following tasks in the Deployment Workbench:

- Create new deployment media as described in Create New Deployment Media in the Deployment Workbench.
- Modify existing media as described in Modify Existing Media in the Deployment Workbench.
- Copy media as described in Copy Media in the Deployment Workbench.
- Move media as described in Move Media in the Deployment Workbench.
- Delete media as described in Delete Media from the Deployment Workbench.
- Generate media images as described in Generate Media Images in the Deployment Workbench.

- Create bootable devices from deployment media as described in Create Bootable Devices from Deployment Media.

  In addtion to managing deployment media in the Deployment Workbench, you can manage deployment media using the MDT Windows PowerShell cmdlets. For more information on managing deployment media using the MDT Windows PowerShell cmdlets, see the following sections beneath the section, "MDT Windows PowerShell Cmdlets", in the MDT document *Toolkit Reference*:

- **Update-MDTMedia**
- **Get-MDTDeploymentShareStatistics**

Create New Deployment Media in the Deployment Workbench

Create new deployment media in the Deployment Workbench using the New Media Wizard. Start the New Media Wizard using one of the following methods:

- In the console tree, select the Media node. Then, in the Actions pane, select **New Media**.
- In the console tree, select the Media node. Then, from the **Action** menu, select **New Media**.
- In the console tree, right-click the Media node, and then select **New Media**.

To create new deployment media

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/Advanced Configuration/Media (where *deployment_share* is the name of the deployment share to which you will add the application).
3. In the Actions pane, select **New Media**.

   The New Media Wizard starts.
4. Complete the New Media Wizard using the information in Table 69.

Table 69. Information for Completing the New Media Wizard

Expand table

On this wizard page	Do this
General Settings	- In **Media path**, type ***media_path*** (where *media_path* is the fully qualified path to an empty local or network shared folder that is the source folder for creating the media). You can alternatively select **Browse** to find the folder on a local drive or network shared folder. Do not use a subfolder of an existing deployment share as the media path. This will result in the following error when updating the media: "Invalid top level folder" - In **Comments**, type ***comment*** (where *comment* is descriptive text that provides information about the media). - In **Selection profile**, select ***profile*** (where *profile* is the name of the selection profile that will be used to establish the items to be stored on the media). - Select **Next**.
Summary	Review in the information in **Details**, and then select **Next**.
Confirmation	You can select **Save Output** to save the output of the wizard to a file. You can also select **View Script** to view the Windows PowerShell scripts used to perform the wizard tasks. Select **Finish**.

The New Media Wizard finishes. The media are added to the list of media in the details pane of the Deployment Workbench. The *media_path*\Content\Deploy folder is created (where *media_path* is the name of the media path you specified in the wizard), and some base folders are created. The folders and content you specified in the selection profile are copied to the Deploy folder when the Update Media Content Wizard runs.

Modify Existing Media in the Deployment Workbench

Modify existing media in the Media node in the Deployment Workbench using the **Properties** actions as described in View Item Properties in the Deployment Workbench. Configure media in the Deployment Workbench by performing the following steps in the media **Properties** dialog box:

- Configure properties on the **General** tab as described in Configure the Media Properties General Tab.
- Configure properties on the **Rules** tab as described in Configure the Media Properties Rules Tab.
- Configure the settings on the **Windows PE x86 Settings** tab as described in Configure the Media Properties Windows PE x86 Settings Tab.
- Configure the settings on the **Windows PE x86 Components** tab as described in Configure the Media Properties Windows PE x86 Components Tab.
- Configure the settings on the **Windows PE x64 Settings** tab as described in Configure the Media Properties Windows PE x64 Settings Tab.
- Configure the settings on the **Windows PE x64 Components** tab as described in Configure the Media Properties Windows PE x64 Components Tab.

Configure the Media Properties General Tab

The media properties on the **General** tab are configured when you run the New Media Wizard. However, you can update the linked deployment share properties on the **General** tab of the ***media*** **Properties** dialog box (where *media* is the name of the media in the Deployment Workbench).

To modify existing media properties on the General tab

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/Advanced Configuration/Media (where *deployment_share* is the name of the deployment share in which you will configure the media).
3. In the details pane, select ***media*** (where *media* is the name of the media you want to configure).
4. In the Actions pane, select **Properties**.

   The ***media*** **Properties** dialog box opens (where *media* is the name of the media you want to configure).

5. On the **General** tab, configure the settings listed in Table 70 based on the requirements of your organization, and then select **OK**.

Table 70. Configuration Settings on the General Tab of Media Properties

Expand table

Setting	Description
Media identifier	Contains the identifier of the media. The identifier in this text box is automatically generated by the Deployment Workbench and cannot be modified.
Comments	Provides information about the media.
Media path	Contains the fully qualified UNC path to the target folder for the media source files and generated images.
Selection profile	Contains the selection profile that identifies the content to be included in the WIM and ISO image files that the Deployment Workbench generates.
Platforms Supported: Generate x86 boot image	Select to configure the Update Media Content Wizard to create WIM files and bootable media for 32-bit target computers. Selecting both the **x86** and **x64** check boxes generates a dual-boot ISO file. When started, a menu appears from which you can specify which boot image to use.
Platforms Supported: Generate x64 boot image	Select to configure the Update Media Content Wizard to create WIM files and bootable media for 64-bit target computers. Selecting both the **x86** and **x64** check boxes generates a dual-boot ISO file. When started, a menu appears from which you can specify which boot image to use.
Generate a Lite Touch bootable ISO image	Select to configure the Update Media Content Wizard to create ISO files that you can use to boot VMs or create bootable DVDs. If this check box is selected, then you can specify the name of the ISO file to be generated in the **ISO file name**text box. The ISO file is created in the *media_folder* folder (where *media_folder* is the name of the folder that you specified for the media).

The media configuration settings are saved. The modifications are displayed in the details pane of the Deployment Workbench, and the content in the *media_folder*

folder is updated (where *media_folder* is the name of the folder that you specified for the media).

Configure the Media Properties Rules Tab

The media properties on the **Rules** tab are configured when you run the New Media Wizard. However, you can update the linked deployment share properties on the **Rules** tab of the ***media*** **Properties** dialog box (where *media* is the name of the media in the Deployment Workbench).

To modify existing media properties on the Rules tab

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/Advanced Configuration/Media (where *deployment_share* is the name of the deployment share in which you will configure the media).
3. In the details pane, select ***media*** (where *media* is the name of the media you want to configure).
4. In the Actions pane, select **Properties**.

   The ***media*** **Properties** dialog box opens (where *media* is the name of the media you want to configure).
5. On the **Rules** tab, configure the settings listed in Table 71 based on the requirements of your organization, and then select **OK**.

Table 71. Configuration Settings on the Rules Tab of Media Properties

Expand table

Setting	Description
CustomSettings.ini	Contains the current configuration of the CustomSetting.ini file for the deployment media
Edit Bootstrap.ini	Select to modify the contents of the Bootstrap.ini file that the Deployment Workbench generates for the deployment media

The media configuration settings are saved. The modifications are displayed in the details pane of the Deployment Workbench, and the content in the *media_folder* folder (where *media_folder* is the name of the folder that you specified for the media)is updated.

Configure the Media Properties Windows PE x86 Settings Tab

The media properties on the **Windows PE x86 Settings** tab are configured when you run the New Media Wizard. However, you can update the linked deployment share properties on the **Windows PE x86** tab of the ***media* Properties** dialog box (where *media* is the name of the media in the Deployment Workbench).

To configure the Windows PE x86 Settings tab

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/Advanced Configuration/Media (where *deployment_share* is the name of the deployment share in which you will configure the media).
3. In the details pane, select ***media*** (where *media* is the name of the media you want to configure).
4. In the Actions pane, select **Properties**.

   The ***media* Properties** dialog box opens (where *media* is the name of the media you want to configure).
5. On the **Windows PE x86 Settings** tab, configure the settings listed in Table 72 based on the requirements of your organization, and then select **OK**.

Table 72. Configuration Settings on the Windows PE x86 Settings Tab of Media Properties

Expand table

Setting	Description
Generate a Lite Touch Windows PE WIM file	Select to configure the Update Media Content Wizard to create a Windows PE WIM file that includes the LTI deployment scripts. If the check box is:

Setting	Description
	- Selected, the Update Media Content Wizard creates the LiteTouchPE_x86.wim file with the image description specified in the **Image description** text box - Cleared, the Update Media Content Wizard will not create the WIM file
Image description	Contains the image description for the Windows PE WIM file that the Update Media Content Wizard creates. The default value is **Lite Touch Windows PE (x86)**.
Generate a Lite Touch bootable ISO image	Select to configure the Update Media Content Wizard to create a bootable Windows PE ISO file that includes the LTI deployment scripts. If the check box is: - Selected, the Update Media Content Wizard creates the ISO file with the name specified in the **ISO file name** text box - Cleared, the Update Media Content Wizard will not create the ISO file
ISO file name	Contains the file name for the Windows PE ISO file that the Update Media Content Wizard creates. The default value for this text box is **LiteTouchPE_x86.iso**. This text box is enabled only if you select **Generate a Lite Touch bootable ISO image**.
Generate a generic Windows PE WIM file	Select to configure the Update Media Content Wizard to create a Windows PE WIM file that does not include the LTI deployment scripts. If the check box is: - Selected, the Update Media Content Wizard creates the GenericPE_x86.wim file with the image description specified in the **Image description** text box - Cleared, the Update Media Content Wizard will not create the WIM file
Image description	Contains the image description for the generic Windows PE WIM file that the Update Media Content Wizard creates. The default value for this text box is **Generic Windows PE (x86)**. This text box is enabled only if you select **Generate a generic Windows PE WIM file**.

Setting	Description
Generate a generic bootable ISO image	Select to configure the Update Media Content Wizard to create a bootable Windows PE ISO file that does not include the LTI deployment scripts. If the check box is: - Selected, the Update Media Content Wizard creates the Generic_x86.iso file with the image description specified in the **ISO file name** text box - Cleared, the Update Media Content Wizard will not create the ISO file This box is enabled only if you select **Generate a generic Windows PE WIM file**.
ISO file name	Contains the file name for the generic Windows PE ISO file that the Update Media Content Wizard creates. The default value for this text box is **Generic_x86.iso**. This text box is enabled only if you select Generate a **generic bootable ISO image**.
Custom background bitmap file	Contains the fully qualified path to the BMP file that is to be used as the custom background bitmap. The default value for this text box is **%INSTALLDIR%\Samples\Background.bmp**.
Extra directories to add	Contains the fully qualified path to a folder structure to be included in the Windows PE images. The default value for the text box is empty.
Scratch space size	Configures the size of the scratch space in megabytes; you can select a value of **32**, **64**, **128**, **256**, or **512** MB. The default value is **32**.

The media configuration settings are saved, and the media appears in the details pane of the Deployment Workbench.

Configure the Media Properties Windows PE x86 Components Tab

The media properties on the **Windows PE x86 Components** tab are configured when you run the New Media Wizard. However, you can update the linked deployment share properties on the **Windows PE x86 Components** tab of the ***media*** **Properties** dialog box (where *media* is the name of the media in the Deployment Workbench).

To configure the Windows PE x86 Components tab

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.

2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/Advanced Configuration/Media (where *deployment_share* is the name of the deployment share in which you will configure the media).

3. In the details pane, select ***media*** (where *media* is the name of the media you want to configure).

4. In the Actions pane, select **Properties**.

   The ***media*** **Properties** dialog box opens (where *media* is the name of the media you want to configure).

5. On the **Windows PE x86 Components** tab, configure the settings listed in Table 73 based on the requirements of your organization, and then select **OK**.

Table 73. Configuration Settings on the Windows PE x86 Components Tab of Media Properties

Expand table

Setting	Description
Selection profile	Use to select the device drivers and packages to be included in the Windows PE images based on the selection profile you choose. The default value is **All Drivers and Packages**. For more information about selection profiles, see Manage Selection Profiles.
Include all drivers from the selected driver group	Select to configure the Update Media Content Wizard to include all the device drivers in the selection profile specified in the **Selection profile** box.
Include only drivers of the following types	Select to configure the Update Media Content Wizard to include only the device drivers in the chosen selection profile that are specified in the following check boxes: - **Include all network drivers in the selected group** - **Include all video drivers in the selected group** - **Include all mass storage drivers in the selected group** - **Include all system-class drivers in the selected group**
Include all network drivers in the selected group	Select to configure the Update Media Content Wizard to include all network drivers in the chosen selection profile. If the check box is:

Setting	Description
	- Selected, the Update Media Content Wizard includes all network drivers in the selection profile specified in the **Selection profile** box in the Windows PE images - Cleared, the Update Media Content Wizard will not include all network drivers in the selection profile specified in the **Selection profile**box in the Windows PE images This check box is enabled only if you select **Include only drivers of the following types**.
Include all video drivers in the selected group	Select to configure the Update Media Content Wizard to include all video drivers in the chosen selection profile. If the check box is: - Selected, the Update Media Content Wizard includes all video drivers in the selection profile specified in the **Selection profile** box in the Windows PE images - Cleared, the Update Media Content Wizard will not include all video drivers in the selection profile specified in the **Selection profile** list box in the Windows PE images This check box is enabled only if you select **Include only drivers of the following types**.
Include all mass storage drivers in the selected group	Select to configure the Update Media Content Wizard to include all mass storage drivers in chosen the selection profile. If the check box is: - Selected, the Update Media Content Wizard includes all mass storage drivers in the selection profile specified in the **Selection profile** box in the Windows PE images - Cleared, the Update Media Content Wizard will not include all mass storage drivers in the selection profile specified in the **Selection profile** box in the Windows PE images This check box is enabled only if you select **Include only drivers of the following types**.
Include all system-class drivers in the selected group	Select to configure the Update Media Content Wizard to include all system-class drivers in the chosen selection profile. If the check box is: - Selected, the Update Media Content Wizard includes all system-class drivers in the selection profile specified in the **Selection profile** box in the Windows PE images

Setting	Description
	- Cleared, the Update Media Content Wizard will not include all system-class drivers in the selection profile specified in the **Selection profile** box in the Windows PE images This check box is enabled only if you select **Include only drivers of the following types**.
ADO	Select to add optional ADO components to the Windows PE images. These components are necessary for accessing SQL Server databases, such as the MDT DB. If the check box is: - Selected, the ADO components are added to the Windows PE images - Cleared, the ADO components are not added to the Windows PE images This check box is selected by default.
Optional Fonts	Select to configure the Update Media Content Wizard to include the following fonts: - Chinese (ZH-CN) - Chinese (ZH-HK) - Chinese (ZH-TW) - Japanese (JA-JP) - Korean (KO-KR) Add these fonts when performing an LTI deployment of Windows Server images and the Setup files are Japanese, Korean, or Chinese. If the check box for a corresponding font is: - Selected, the Update Media Content Wizard includes the font in the Windows PE images - Cleared, the Update Media Content Wizard will not include the font in the Windows PE images Adding fonts to Windows PE boot images increases the size of the images. Add fonts only if necessary.

The media configuration settings are saved, and the media appears in the details pane of the Deployment Workbench.

Configure the Media Properties Windows PE x64 Settings Tab

The media properties on the **Windows PE x64 Settings** tab are configured when you run the New Media Wizard. However, you can update the linked deployment share properties on the **Windows PE x64 Settings** tab of the ***media* Properties** dialog box (where *media* is the name of the media in the Deployment Workbench).

To configure the Windows PE x64 Settings tab

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/Advanced Configuration/Media (where *deployment_share* is the name of the deployment share in which you will configure the media).
3. In the details pane, select ***media*** (where *media* is the name of the media you want to configure).
4. In the Actions pane, select **Properties**.

   The ***media* Properties** dialog box opens (where *media* is the name of the media you want to configure).
5. On the **Windows PE x64 Settings** tab, configure the settings listed in Table 74 based on the requirements of your organization, and then select **OK**.

Table 74. Configuration Settings on the Windows PE x64 Settings Tab of Media Properties

Expand table

Setting	Description
Generate a Lite Touch Windows PE WIM file	Select to configure the Update Media Content Wizard to create a Windows PE WIM file that includes the LTI deployment scripts. If the check box is: - Selected, the Update Media Content Wizard creates the LiteTouchPE_x86.wim file with the image description specified in the **Image description** box - Cleared, the Update Media Content Wizard will not create the WIM file

Setting	Description
Image description	Contains the image description for the Windows PE WIM file that the Update Media Content Wizard creates. The default value is **Lite Touch Windows PE (x64)**.
Generate a Lite Touch bootable ISO image	Select to configure the Update Media Content Wizard to create a bootable Windows PE ISO file that includes the LTI deployment scripts. If the check box is: - Selected, the Update Media Content Wizard creates the ISO file with the name specified in the **ISO file name** box - Cleared, the Update Media Content Wizard will not create the ISO file
ISO file name	Contains the file name for the Windows PE ISO file that the Update Media Content Wizard creates. The default value for this text box is **LiteTouchPE_x64.iso**. This box is enabled only if you select **Generate a Lite Touch bootable ISO image**
Generate a generic Windows PE WIM file	Select to configure the Update Media Content Wizard to create a Windows PE WIM file that does not include the LTI deployment scripts. If the check box is: - Selected, the Update Media Content Wizard creates the GenericPE_x64.wim file with the image description specified in the **Image description** box - Cleared, the Update Media Content Wizard will not create the WIM file
Image description	Contains the image description for the generic Windows PE WIM file that the Update Media Content Wizard creates. The default value for this text box is **Generic Windows PE (x64).** This box is enabled only if you select **Generate a generic bootable ISO Windows PE WIM file**.
Generate a generic bootable ISO image	Select to configure the Update Media Content Wizard to create a bootable Windows PE ISO file that does not include the LTI deployments scripts. If the check box is: - Selected, the Update Media Content Wizard creates the Generic_x64.iso file with the image description specified in the **ISO file name** box - Cleared, the Update Media Content Wizard will not create the ISO file

Setting	Description
	This box is enabled only if you select **Generate a generic Windows PE WIM file**.
ISO file name	Contains the file name for the generic Windows PE ISO file that the Update Media Content Wizard creates. The default value for this text box is **Generic_x64.iso**. This text box is enabled only if the Generate a generic bootable ISO image check box is selected.
Custom background bitmap file	Contains the fully qualified path to the BMP file that is to be used as the custom background bitmap. The default value for this text box is **%INSTALLDIR%\Samples\Background.bmp**.
Extra directories to add	Contains the fully qualified path to a folder structure to be included in the Windows PE images. The default value for the text box is empty.
Scratch space size	Configures the size of the scratch space in megabytes; you can select a value of **32**, **64**, **128**, **256**, or **512** MB. The default value is **32**.

The media configuration settings are saved, and the media appears in the details pane of the Deployment Workbench.

Configure the Media Properties Windows PE x64 Components Tab

The media properties on the **Windows PE x64 Components** tab are configured when you run the New Media Wizard. However, you can update the linked deployment share properties on the **Windows PE x64 Components** tab of the ***media*** **Properties** dialog box (where *media* is the name of the media in the Deployment Workbench).

To configure the Windows PE x64 Components tab

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/Advanced Configuration/Media (where *deployment_share* is the name of the deployment share in which you will configure the media).
3. In the details pane, select ***media*** (where *media* is the name of the media you want to configure).
4. In the Actions pane, select **Properties**.

The ***media*** **Properties** dialog box opens (where *media* is the name of the media you want to configure).

5. On the **Windows PE x86 Components** tab, configure the settings listed in Table 75 based on the requirements of your organization, and then select **OK**.

Table 75. Configuration Settings on the Windows PE x64 Components Tab of Media Properties

Expand table

Setting	Description
Selection profile	Selects the device drivers and packages to be included in the Windows PE images based on the selection profile you choose. The default value is **All Drivers and Packages**. For more information about selection profiles, see Manage Selection Profiles.
Include all drivers from the selected driver group	Select to configure the Update Media Content Wizard to include all the device drivers in the selection profile specified in the **Selection profile** box.
Include only drivers of the following types	Select to configure the Update Media Content Wizard to include only the device drivers in the chosen selection profile that are specified in the following check boxes: - **Include all network drivers in the selected group** - **Include all video drivers in the selected group** - **Include all mass storage drivers in the selected group** - **Include all system-class drivers in the selected group**
Include all network drivers in the selected group	Select to configure the Update Media Content Wizard to include all network drivers in the chosen selection profile. If the check box is: - Selected, the Update Media Content Wizard includes all network drivers in the selection profile specified in the **Selection profile** box in the Windows PE images - Cleared, the Update Media Content Wizard will not include all network drivers in the selection profile specified in the **Selection profile** box in the Windows PE images This check box is enabled only if you select **Include only drivers of the following types**.

Setting	Description
Include all video drivers in the selected group	Select to configure the Update Media Content Wizard to include all video drivers in the chosen selection profile. If the check box is: - Selected, the Update Media Content Wizard includes all video drivers in the selection profile specified in the **Selection profile** box in the Windows PE images - Cleared, the Update Media Content Wizard will not include all video drivers in the selection profile specified in the **Selection profile** box in the Windows PE images This check box is enabled only if you select **include only drivers of the following types**.
Include all mass storage drivers in the selected group	Select to configure the Update Media Content Wizard to include all mass storage drivers in the chosen selection profile. If the check box is: - Selected, the Update Media Content Wizard includes all mass storage drivers in the selection profile specified in the **Selection profile**box in the Windows PE images - Cleared, the Update Media Content Wizard will not include all mass storage drivers in the selection profile specified in the **Selection profile** box in the Windows PE images This check box is enabled only if you select **include only drivers of the following types**.
Include all system-class drivers in the selected group	Select to configure the Update Media Content Wizard to include all system-class drivers in chosen the selection profile. If the check box is: - Selected, the Update Media Content Wizard includes all system-class drivers in the selection profile specified in the **Selection profile** box in the Windows PE images - Cleared, the Update Media Content Wizard will not include all system-class drivers in the selection profile specified in the **Selection profile** box in the Windows PE images This check box is enabled only if you select **Include only drivers of the following types**.
ADO	Select to add the optional ADO components to the Windows PE images. These components are necessary for accessing SQL Server databases, such as the MDT DB. If this check box is:

Setting	Description
	- Selected, the ADO components are added to the Windows PE images - Cleared, the ADO components are not added to the Windows PE images This check box is selected by default.
Optional Fonts	Use to configure the Update Media Content Wizard to include the following fonts: - Chinese (ZH-CN) - Chinese (ZH-HK) - Chinese (ZH-TW) - Japanese (JA-JP) - Korean (KO-KR) Add these fonts when performing an LTI deployment of Windows Server images when the Setup files are Japanese, Korean, or Chinese. If the check box for a corresponding font is: - Selected, the Update Media Content Wizard includes the font in the Windows PE images - Cleared, the Update Media Content Wizard will not include the font in the Windows PE images Adding fonts to Windows PE boot images increases the size of the images. Add fonts only if necessary.

The media configuration settings are saved, and the media appears in the details pane of the Deployment Workbench.

Copy Media in the Deployment Workbench

You can copy and paste media in the Deployment Workbench using the **Copy** and **Paste** actions as described in Copy Items in the Deployment Workbench.

Move Media in the Deployment Workbench

You can move media in the Deployment Workbench using the **Cut** and **Paste** actions as described in Move Items in the Deployment Workbench.

Delete Media from the Deployment Workbench

You can delete media in the Deployment Workbench using the Delete Selected Items Wizard as described in Delete Items from the Deployment Workbench. The Delete Selected Items Wizard allows you to delete individual media.

Generate Media Images in the Deployment Workbench

You can generate media images of the media content in the Deployment Workbench using the Update Media Content Wizard. The Update Media Content Wizard creates WIM file images of the media content that you can use to perform stand-alone LTI deployments from media. Ensure that sufficient storage exists for the folder containing the media content prior to running the Update Media Content Wizard, as the wizard does not verify that sufficient storage exists prior to generating the media content.

> **Note**
>
> The Update Media Content Wizard opens the media path in single-user mode, which assumes that no other users are simultaneously making updates to the files and folders in the media path. If other users make changes while the Update Media Content Wizard runs, those changes may be overwritten and lost during the generation process.

To generate media images of media content

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/Advanced Configuration/Media (where *deployment_share* is the name of the deployment share to which you will add the application).
3. In the details pane, select ***media*** (where *media* is the name of the media for which you want to generate the media).
4. In the Actions pane, select **Update Media Content**.

The Update Media Content Wizard starts. The replication process starts automatically and is displayed on the **Progress** wizard page.

5. Complete the Update Media Content Wizard using the information in Table 76.

Table 76. Information for Completing the Update Media Content Wizard

Expand table

On this wizard page	Do this
Progress	View the progress of the replication process.
Confirmation	You can select **Save Output** to save the output of the wizard to a file. You can also select **View Script** to view the Windows PowerShell scripts used to perform the wizard tasks. Select **Finish**.

> **Note**
>
> If you view the output of the wizard, the generation process appears to have occurred twice. However, the process is actually performed in two passes: The first pass copies new items into the media target folders, and the second pass deletes any items that are no longer needed in the media target folders.

The Update Media Content Wizard finishes, and the following files are created:

- An ISO file in the *media_folder* folder (where *media_folder* is the name of the folder that you specified for the media)

  Generating the ISO file is an option that you configure by selecting the **Generate a Lite Touch bootable ISO image** check box on the **General** tab of the ***media* Properties** dialog box. Clear this check box to reduce the time needed to generate the media unless you need to create bootable DVDs or start VMs from the ISO file.

- WIM files in the *media_folder*\Content\Deploy\Boot folder (where *media_folder* is the name of the folder that you specified for the media).

  The folders and the content you specified in the selection profile in the media are stored in the image files.

You can also create a bootable device that contains a copy of the *media_folder*\Content folder (where *media_folder* is the name of the folder you specified for the media) so that you can start a target computer from a UFD or USB hard disk. For more information, see Create Bootable Devices from Deployment Media.

Create Bootable Devices from Deployment Media

You may need to deploy images to target computers using a bootable device (such as a UFD or a USB hard disk) when the target computer does not have a high-speed, persistent connection to a deployment share.

> ⓘ **Note**
>
> The target computer must support starting from the device to use this method.

To create bootable devices from deployment media

1. On a computer running Windows 7 or later, insert the UFD or USB hard disk.
2. Run Diskpart.exe, and type **list disk** to determine the disk number associated with the device.
3. Type the following commands, where *N* is the disk number identified in step 2:
   - **select disk *N***
   - **clean**
   - **create partition primary**
   - **select partition 1**
   - **active**
   - **format fs=ntfs**
   - **assign**
   - **exit**
4. Copy the contents of the *media_folder*\Content folder (where *media_folder* is the name of the folder you specified in the media) to the device.

Manage the MDT DB

The MDT DB augments the configuration that CustomSettings.ini provides for both LTI and ZTI deployments. The MDT DB allows you to centrally manage configuration settings for the target computers. Although you can perform large-scale deployments using the CustomSettings.ini file, the MDT DB can help reduce the effort need to manage such deployments.

For more information on managing the MDT DB, see Performing Deployments Using the MDT DB.

In addtion to managing the MDT DB in the Deployment Workbench, you can manage the MDT DB using the MDT Windows PowerShell cmdlets. For more information on managing the MDT DB using the MDT Windows PowerShell cmdlets, see the following sections beneath the section, "MDT Windows PowerShell Cmdlets", in the MDT document *Toolkit Reference*:

- **New-MDTDatabase**
- **Update-MDTDatabaseSchema**

Configuring LTI Task Sequence Steps in the Deployment Workbench

Configure LTI task sequences steps in the Deployment Workbench by:

- Configuring LTI task sequence step conditions as described in Configure Task Sequence Step Conditions
- Configuring LTI task sequence steps that perform disk-related actions as described in Configure Disk Task Sequence Steps
- Configuring LTI task sequence steps that perform network-related actions as described in Configure Network Task Sequence Steps
- Configuring LTI task sequence steps that perform server role-related actions as described in Configure Server Role Task Sequence Steps for LTI
- Configuring the **Check Bios** task sequence step to include a list of incompatible basic input/output system (BIOS) versions as described in Configure the Check BIOS Task Sequence Step for the List of Incompatible BIOS Versions

Configure Task Sequence Step Conditions

In certain scenarios, consider conditionally running a task sequence step based on defined criteria. Configure task sequence step conditions on the **Options** tab of a task sequence step. Add any combinations of these conditions to determine whether the task sequence step should run. For example, you could use the values of a task sequence variable and of a registry setting to determine whether a task sequence step should run.

Configure conditional task sequence steps by performing any combination of the following actions:

- Add one or more IF statements to a task sequence step condition as described in Add IF Statements to Task Sequence Step Conditions.
- Add one or more task sequence variables to a task sequence step condition as described in Add Task Sequence Variables to Task Sequence Step Conditions.
- Add one or more target operating system versions to a task sequence step condition as described in Add Operating System Versions to Task Sequence Step Conditions.
- Add one or more WMI query results to a task sequence step condition as described in Add WMI Queries to Task Sequence Step Conditions.
- Add the value of one or more registry settings to a task sequence step condition as described in Add Registry Settings to Task Sequence Step Conditions.
- Add the test for software installed on the target computer to a task sequence step condition as described in Add a Test for Installed Software to Task Sequence Step Conditions.
- Add the test for various folder properties to a task sequence step condition as described in Add a Test for Folder Properties to Task Sequence Step Conditions.
- Add the test for various file properties to a task sequence step condition as described in Add a Test for File Properties to Task Sequence Step Conditions.

Add IF Statements to Task Sequence Step Conditions

All task sequence conditions include one or more `IF` statements, which are the foundation for creating conditional task sequence steps. A task sequence step condition can include only one `IF` statement, but you can nest multiple `IF` statements beneath the top-level `IF` statement to create more complex conditions.

You test an IF statement based on the conditions listed in Table 77, which you configure in the `IF` **Statement Properties** dialog box.

Table 77. Conditions Available in IF Statements

Expand table

Condition	Select this option to run the task sequence if
All conditions	All the conditions beneath this IF statement must be true.
Any conditions	Any the conditions beneath this IF statement are true.
None	None the conditions beneath this IF statement are true.

Complete the condition for running the task sequence step by adding other criteria to the conditions (for example, task sequence variables or values in a registry setting).

To add an IF statement condition to a task sequence step

1. On the ***step*** **Option** tab (where *step* is the name of the task sequence step to configure), select **Add**, and then select `If` **statement**.
2. In the `If` **Statement Properties** dialog box, select ***condition*** (where *condition* is one of the conditions listed in Table 77), and then select **OK**.

Add Task Sequence Variables to Task Sequence Step Conditions

Create conditions based on any task sequence variable (including those that MDT defines). These variables also include the environment variables available in the operating system.

To configure a condition based on a task sequence variable, provide the following information in the **Task Sequence Variable Condition** dialog box:

- **Variable**. The name of the task sequence variable to include as a condition. This name must match the exact spelling of the variable but is not case sensitive.
- **Condition**. This can be **exists** (which is true if the variable exists, regardless of its value) or a standard logical operator.
- **Value**. The value of the task sequence variable to use in the condition.

To add a Task Sequence Variable condition to a task sequence step

1. On the ***step*** **Options** tab (where *step* is the name of the task sequence step to configure), select **Add**, and then select **Task Sequence Variable**.
2. In the **Task Sequence Variable Condition** dialog box, in the **Variable box**, type ***variable*** (where *variable* is the name of the task sequence variable).
3. In the **Task Sequence Variable Condition** dialog box, in the **Condition box**, select ***condition*** (where *condition* is the logical operation to use in the condition as listed in Table 77).
4. In the **Task Sequence Variable Condition** dialog box, in the **Value box**, type ***value*** (where value is the *value* of the task sequence variable), and then select **OK**.

Add Operating System Versions to Task Sequence Step Conditions

Create conditions based on the operating system version by providing the following information in the **Task Sequence OS Condition** dialog box:

- **Architecture**. The name of the instruction set on which the operating system is designed, either x86 or x64
- **Operating System**. A version of Windows
- **Condition**. A logical operator

To add an Operating System Version condition to a task sequence step

1. On the ***step*** **Option** tab (where step is the name of the task sequence step to configure), select **Add**, and then select **Operating System Version**.
2. In the **Task Sequence OS Condition** dialog box, in the **Architecture box**, select ***architecture*** (where *architecture* is the name of the operating system architecture).
3. In the **Task Sequence OS Condition** dialog box, in the **Operating system box**, select the ***operating system to use***.
4. In the **Task Sequence OS Condition** dialog box, in the **Condition box**, select ***condition*** (where *condition* is the logical operation to use in the condition), and then select **OK**.

Add WMI Queries to Task Sequence Step Conditions

You can use WMI queries in a task sequence condition. WMI is the primary management technology for Windows operating systems and enables consistent and uniform

management, control, and monitoring of systems throughout the enterprise. Based on industry standards, WMI allows you to query, change, and monitor configuration settings on desktop and server systems, applications, networks, and other enterprise components. You can also write scripts that use the WMI scripting library to work with WMI and create a wide range of systems management and monitoring scripts. For more information about WMI, see the WMI Scripting Primer.

To configure a condition based on a WMI condition, provide the following information in the **Task Sequence WMI Condition** dialog box:

- **WMI namespace**. The default, \root\cimv2, refers to a specific WMI namespace. Namespaces are grouped hierarchically and are similar to the way folders are grouped in the operating system. Within each namespace is a collection of classes that correspond to a managed resource.
- **WQL query**. This dialog box contains the query that runs when the conditions are met. WMI is generally queried in two ways: by retrieving an entire WMI object or by using a Structured Query Language (SQL)-like query. In the query, system information or querying computers could be accessed across a network.

To add a WMI query result condition to a task sequence step

1. On the ***step*** **Option** tab (where *step* is the name of the task sequence step to configure), select **Add**, and then select **Query WMI**.
2. In the **Task Sequence WMI Condition** dialog box, in the **WMI namespace box**, type ***WMI namespace***.
3. In the **Task Sequence WMI Condition** dialog box, in the **WQL query box**, type the query script to be run, and then select **OK**.

Add Registry Settings to Task Sequence Step Conditions

Evaluate registry settings during the task sequence; based on defined criteria, choose whether to run additional processes. The registry contains two basic elements: keys and values.

- Registry keys are similar to folders. Each key can contain subkeys, which in turn can contain further subkeys, all of which might contain values. Keys are referenced with syntax similar to Windows path names, using backslashes (\) to indicate levels of hierarchy. For example, **HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows** refers to the subkey **Windows** of the subkey **Microsoft** of the key **Software** of the subtree **HKEY_LOCAL_MACHINE**.

- Registry values are name-data pairs stored within but referenced separately from these keys. Value names can contain backslashes, but including backslashes so makes the values difficult to distinguish from their key paths.

  To configure a condition based on a registry setting condition, provide the following information in the **Registry Setting** dialog box:

- **Root key**. The subtree of the registry key. Base the condition on one of the following:
  - **HKEY_CURRENT_USER**
  - **HKEY_LOCAL_MACHINE**
  - **HKEY_USERS**
  - **HKEY_CURRENT_CONFIG**
- **Key**. The value of the registry key to use in the condition
- **Condition**. A logical operator, such as `AND` or `OR`
- **Value name**. The name of the value to use in the condition
- **Value type**. The value type, which can be one of the following:
  - **REG_SZ**
  - **REG_EXPAND_SZ**
  - **REF_DWORD**
- **Value**. The value of the value type for the provided registry key to use in the condition

To add a Registry Setting condition to a task sequence step

1. On the ***step*** **Option** tab (where *step* is the name of the task sequence step to configure), select **Add**, and then select **Registry Setting**.
2. In the **Registry Setting** dialog box, in the **Root key box**, select ***Root key***.
3. In the **Registry Setting** dialog box, in the **Key box**, type ***key*** (where *key* is the remainder of the registry key minus the subtree).
4. In the **Registry Setting** dialog box, in the **Condition box**, select ***condition*** (where *condition* is the logical operation to use in the condition).

5. In the **Registry Setting** dialog box, in the **Value name box**, type the name of the ***Value name***.
6. In the **Registry Setting** dialog box, in the **Value type box**, select ***Value type***.
7. In the **Registry Setting** dialog box, in the **Value box**, type the value for which testing will occur, and then select **OK**.

Add a Test for Installed Software to Task Sequence Step Conditions

You can evaluate installed software based on the product information provided in the Microsoft Installer (MSI) file. You can use this information to match a specific product using both the product code and the upgrade code, or you can use it to match any version of this product using only the upgrade code.

To add an Installed Software condition to a task sequence step

1. On the ***step*** **Option** tab (where step is the name of the task sequence step to configure), select **Add**, and then select **Installed Software**.
2. In the **Installed Software** dialog box, in the **MSI file box**, browse to the specific MSI file associated with the installed software. The product information will be extracted from the MSI file and populate the respective boxes.
3. In the **Installed Software** dialog box, select one of the two following conditions:
    - Match this specific product (Product Code and Upgrade Code)
    - Match any version of this product (Upgrade Code only)
4. In the **Installed Software** dialog box, select **OK**.

Add a Test for Folder Properties to Task Sequence Step Conditions

You can evaluate folders based on folder properties. In addition to evaluating the path of the folder to be tested, test for its time stamp under a number of conditions.

To configure a condition based on a folder's property, provide the following information on the **Folder Properties** dialog box:

- **Path**. The path of the folder to test.
- **Condition**. (Optional) One of the logical operators

- **Date**. (Optional) The date of the file
- **Time**. (Optional) The time stamp of the file

To add a Folder Properties condition to a task sequence step

1. On the ***step*** **Option** tab (where step is the name of the task sequence step to configure), select **Add**, and then select **Folder Properties**.
2. In the **Folder Properties** dialog box, in the Path box, go to the folder to be tested.
3. In the **Folder Properties** dialog box, to check the time stamp of the file, select the **Check the timestamp** check box, set the condition of the value, set a date, and set the time. Otherwise, clear the **Check the timestamp** check box so that the additional condition will not be tested for.
4. In the **Folder Properties** dialog box, select **OK**.

Add a Test for File Properties to Task Sequence Step Conditions

You can evaluate files based on the file properties. In addition to evaluating the path of the file to be tested, test for its version and time stamp under a number of conditions.

To configure a condition based on a file property, provide the following information on the **File Properties**dialog box:

- **Path**. The path of the file being tested
- **Version**. (Optional) Version of the file being tested
- **Condition**. (Optional) A logical operator
- **Date**. (Optional) The date of the file
- **Time**. (Optional) The time stamp of the file

To add a File Properties condition to a task sequence step

1. On the ***step*** **Option** tab (where step is the name of the task sequence step to configure), select **Add**, and then select **File Properties**.
2. In the **File Properties** dialog box, in the **Path box**, browse to the file to be tested.
3. In the **File Properties** dialog box, if you want to check the version of the file, select the **Check the version** check box, set the condition of the value, and type the

version number to be tested for. Otherwise, clear the **Check the version** check box so that the additional condition will not be tested for.

4. In the **File Properties** dialog box, to check the time stamp of the file, select the **Check the timestamp** check box, set the condition of the value, set a date, and set the time. Otherwise, clear the **Check the timestamp** check box so that the additional condition will not be tested for.

5. In the **File Properties** dialog box, select **OK**.

Configure Disk Task Sequence Steps

You can customize task sequences to configure the disk settings on the target computer. Configure the disk settings in the Deployment Workbench or in the Configuration Manager console.

To configure task sequence steps that perform disk-related functions, perform the following steps:

- Configure the **Format and Partition Disk** task sequence step types as described in Configure Format and Partition Disk Task Sequence Steps
- Configure **Enable BitLocker** task sequence step types as described in Configure Enable BitLocker Task Sequence Steps

Configure Format and Partition Disk Task Sequence Steps

Task sequence steps based on the **Format and Partition Disk** task sequence step type allow the creation of multiple partitions and are typically used to create secondary partitions for storing data. Custom disk partitions are only supported in New Computer scenarios.

> ⓘ **Note**
>
> LTI does not support the deployment of the target operating system to logical drives or dynamic disks.

To configure task sequence steps based on the Format and Partition Disk task sequence step type

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.

2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/Task Sequences (where *deployment_share* is the name of the deployment share in which you will configure the task sequence).

3. In the details pane, select ***task_sequence_name*** (where *task_sequence_name* is the name of the task sequence you want to configure).

4. In the Actions pane, select **Properties**.

   The ***task_sequence_name*** **Properties** dialog box opens (where *task_sequence_name* is the name of the task sequence you want to configure).

5. On the **Task Sequence** tab, in the task sequence hierarchy, go to *task_sequence_step* (where *task_sequence_step* is the name of the task sequence step that is a **Format and Partition Disk Task Sequence** step type), and then select the **Properties** tab.

6. On the **Properties** tab, configure the settings listed in Table 78 based on the requirements of your organization, and then select **OK**.

Table 78. Configuration Settings on the Properties Tab of Format and Partition Disk Task Sequence Step Type

Expand table

Setting	Description
Type	Contains the task sequence type, which is always set to **Format and Partition Disk**.
Name	Contains the name of the task sequence step that appears in the task sequence hierarchy.
Comments	Provides descriptive information about the task sequence step.
Disk number	Includes the disk number to be partitioned and formatted; the disk number is zero-based, which means the first disk will be number zero (0).
Disk type	Contains the type of partition type, which can be **Standard (MBR)** or **GPT**.
Volume	Contains a list of the disk volumes to create on the partition.
Create new partition button	Select to create a new partition definition and access the **Partition Properties** dialog box. For information about completing the **Partition**

Setting	Description
	Properties dialog box, see Table 79.
Partition properties button	Select to display the **Partition Properties** dialog box for the partition selected in the **Volume** box For information about completing the **Partition Properties** dialog box, see Table 79.
Partition delete button	Select to delete the partition selected in the **Volume** box. The **Confirm Partition Delete** dialog box is displayed in which you can confirm (**Yes**) or cancel (**No**) the deletion.

Table 79 lists the configuration settings for the **Partition Properties** dialog box. Configure the settings in Table 79 based on the requirements of your organization, and then select **OK**.

Table 79. Configuration Settings on the Partition Properties Dialog Box

Expand table

Setting	Description
Partition name	Contains the name of the partition, which appears in the **Volume** box on the **Partition and Format Disk** task sequence step type's **Properties** tab.
Partition type	Contains the type of partition to be created, which can be **Primary** or **Extended**.
Use a percentage of remaining free space	Select to configure the task sequence step to create a partition based on a percentage of the remaining free disk space. If you select this option, specify the percentage in the **Size(%)** box.
Use specific size	Select to configure the task sequence step to create a partition of a specific size. If you select this option, specify the size of the partition in the **Size** box.
Make this a boot partition	Select to configure the task sequence step to configure the partition as the boot partition of the target computer. If the check box is: - Selected, the partition is configured as the boot partition - Cleared, the partition is not configured as the boot partition
File system	Contains the file system for the format process, which can be **NTFS** or **FAT32**.

Setting	Description
Quick format	Select to configure the task sequence step to format the partition by performing a quick format. If the check box is: - Selected, the partition format is performed using the quick format process - Cleared, the partition format is performed using the standard format process
Variable	Contains the name of a task sequence variable used to store the drive letter assigned to the partition. MDT automatically creates an additional partition for new computers when deploying Windows or when BitLocker has been requested.

Configure Enable BitLocker Task Sequence Steps

Use this task to enable the **BitLocker** task. BitLocker is a full-disk encryption feature included in Windows designed to protect data by providing encryption for entire volumes. By default, it uses the Advanced Encryption Standard (AES), also known as *Rijndael*, a block cipher adopted as an encryption standard by the U.S. government.

The AES algorithm in Cipher-block Chaining mode with a 128-bit key is often combined with the Elephant diffuser for additional security. BitLocker is available only in the Enterprise and Ultimate editions of Windows.

Select one of the following methods of enabling BitLocker:

- In a task sequence, enable the **BitLocker** task.

  In addition, configure the partition in the **Format and Partition** task, which is necessary for New Computer scenarios but not in Refresh Computer scenario. The most common configurations are:

  - One partition: 100%
  - One partition and some unallocated space

- In the Deployment Wizard, configure the **BitLocker** page. This requires that the **Enable BitLocker** task be enabled in the task sequence used for deployment.
- In the CustomSettings.ini file, set the following properties:

  - BDEInstall=TPM

- BdeInstallSuppress=NO
- BDeWaitForEncryption=False
- BDEDriveSize=2000
- BDEDriveLetter=S:
- BDEKeyLocation=C:
- SkipBitLocker=YES

For more information about enabling BitLocker, see BitLocker Frequently Asked Questions.

Configure Network Task Sequence Steps

You can customize task sequences in the Deployment Workbench to configure the network settings on the target computer. To configure task sequence steps that perform network-related functions, perform the following steps:

- Configure **Capture Network Settings** task sequence steps as described in Configure Capture Network Settings Task Sequence Steps
- Configure **Apply Network Settings** task sequence steps as described in Configure Apply Network Settings Task Sequence Steps

Configure Capture Network Settings Task Sequence Steps

Task sequence steps based on the **Capture Network Settings** task sequence step allow you to capture network configuration settings for all network adapters on the target computer that have statically configured IP addresses for Refresh Computer deployment scenarios.

The LTI task sequence templates provided with MDT do not include a task sequence step based on the **Capture Network Settings** task sequence step type. For Refresh Computer deployment scenarios, add a task sequence step based on the **Capture Network Settings** task sequence step type in the State Capture phase.

> **Note**

The settings captured by sequence steps based on the **Capture Network Settings** task sequence step in the Refresh Computer deployment scenario override any IP configuration settings specified in the CustomSettings.ini file or in the MDT DB.

To configure task sequence steps based on the Capture Network Settings task sequence step type

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/Task Sequences (where *deployment_share* is the name of the deployment share in which you will configure the task sequence).
3. In the details pane, select *task_sequence_name* (where *task_sequence_name* is the name of the task sequence you want to configure).
4. In the Actions pane, select **Properties**.

   The ***task_sequence_name*** **Properties** dialog box opens (where *task_sequence_name* is the name of the task sequence you want to configure).
5. On the **Task Sequence** tab, in the task sequence hierarchy, go to *task_sequence_step* (where *task_sequence_step* is the name of a task sequence that is a **Capture Network Settings** task sequence step type), and then select the **Properties** tab.
6. On the **Properties** tab, configure the settings listed in Table 80 based on the requirements of your organization, and then select **OK**.

Table 80. Configuration Settings on the Properties Tab of the Capture Network Settings Task Sequence Step Type

Expand table

Setting	Description
Type	Contains the task sequence type, which is always set to **Capture Network Settings**

Setting	Description
Name	Contains the name of the task sequence step that appears in the task sequence hierarchy
Description	Provides descriptive information about the task sequence step

Configure Apply Network Settings Task Sequence Steps

Task sequence steps based on the **Apply Network Settings** task sequence step type allow the configuration of network settings for each network adapter in the target computer. The network settings that this task sequence step type configures are the same settings configured in the properties of a network adapter.

For each network adapter in the target computer, configure the corresponding network settings. If no configuration settings are specified, the **Apply Network Settings** task sequence step type configures the task sequence step that in turn configures the network adapters on the target computer to use Dynamic Host Configuration Protocol (DHCP) for configuration.

> ⓘ **Note**
>
> Network configuration settings the **Capture Network Settings** task sequence step captures in the State Capture phase override any configuration settings you specify in this task sequence type.

Some of the LTI task sequence templates in MDT include a task sequence step in the State Restore phase named Apply Network Settings. In most instances, configure the existing task sequence step instead of creating a new task sequence step.

To configure task sequence steps based on the Apply Network Settings task sequence step type

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/Task Sequences (where *deployment_share* is the name of the deployment share where you will configure the task sequence).

3. In the details pane, select ***task_sequence_name*** (where *task_sequence_name* is the name of the task sequence you want to configure).

4. In the Actions pane, select **Properties**.

   The ***task_sequence_name*** **Properties** dialog box opens (where *task_sequence_name* is the name of the task sequence you want to configure).

5. On the **Task Sequence** tab, in the task sequence hierarchy, go to *task_sequence_step* (where *task_sequence_step* is the name of a task sequence that is an **Apply Network Setting** task sequence step type), and then select the **Properties** tab.

6. On the **Properties** tab, configure the settings listed in Table 81 based on the requirements of your organization, and then select **OK**.

Table 81. Configuration Settings on the Properties Tab of the Apply Network Settings Task Sequence Step Type

Expand table

Setting	Description
Type	Contains the task sequence type, which is always set to **Capture Network Settings**.
Name	Contains the name of the task sequence step that appears in the task sequence hierarchy.
Description	Provides descriptive information about the task sequence step.
Create new network adapter setting button	Select to create a new network adapter setting definition and access the **Network Settings** dialog box. For information about completing the tabs of the **Network Settings** dialog box, see Table 82, Table 83, and Table 84.
Network adapter setting properties button	Select to display the **Network Settings** dialog box for the network adapter you selected. For information about completing the tabs of the **Network Settings** dialog box, see Table 82, Table 83, and Table 84.
Network adapter settings delete button	Select to delete the network adapter setting you selected. In the **Confirm Adapter Delete** dialog, confirm (**Yes**) or cancel (**No**) the deletion.

Table 82 lists the configuration settings for the **General** tab of the **Network Settings** dialog box. Configure the settings in Table 82 based on the requirements of your organization, and then select **OK**.

Table 82. Configuration Settings on the General tab of the Network Settings Dialog Box

Expand table

Setting	Description
Name	Contains the name of the network adapter, which appears in the box on the **Apply Network Settings** task sequence step type's **Properties** tab.
Obtain an IP address automatically	Select to configure the task sequence step to configure the IP address of the network adapter on the target computer using DHCP.
Use the following IP address	Select to configure the task sequence step to configure the IP address of the network adapter on the target computer using the values you specify in the **Network Settings** and **Gateway Settings** boxes.
Network Settings	Contains a list of IP addresses and subnet masks to be configured for the network adapter. Select one of the following to modify the IP addresses and subnet masks in this list: - **Add**. Add an IP address and subnet mask combination to the list box. - **Properties**. Modify the IP address and subnet mask for an existing entry. - **Delete**. Delete an existing entry. This box is enabled only if you select **Use the following IP address**.
Gateway Settings	Contains a list of gateway IP addresses and routing cost metrics to be configured for the network adapter. Select one of the following to modify the gateway IP address and routing cost metric in this list: - **Add**. Add the IP address and cost metric for a gateway to the list box. - **Properties**. Modify the IP address and cost metric for a gateway an existing entry.

Setting	Description
	- **Delete**. Delete an existing entry. This box is enabled only if you select **Use the following IP address**.

Table 83 lists the configuration settings for the **DNS** tab of the **Network Settings** dialog box. Configure the settings in Table 83 based on the requirements of your organization, and then select **OK**.

Table 83. Configuration Settings on the DNS tab of the Network Settings Dialog Box

Expand table

Setting	Description
Obtain DNS server automatically	Select to configure the task sequence step to configure the DNS settings of the network adapter on the target computer using DHCP.
Use the following DNS Servers	Select to configure the task sequence step to configure the DNS settings of the network adapter on the target computer using the values specified in the **DNS Server addresses in order of use** box.
DNS Server addresses in order of use	Contains a list of DNS server IP addresses to be configured for the network adapter. Select one of the following to modify the list of DNS servers: - **Add**. Add the DNS Server IP address to the list box. - **Properties**. Modify the DNS server IP address for an existing entry. - **Delete**. Delete an existing entry.
DNS Suffix	Contains the DNS domain name suffix to be added to any DNS queries.
Register this connection's address in DNS	Select to configure the network adapter to register the network adapters IP address of the primary DNS server. If the check box is: - Selected, the IP address of the network adapter is registered in DNS - Cleared, the IP address of the network adapter is not registered in DNS

Setting	Description
	By default, this check box is selected.
Use this connection's suffix in DNS registration	Select to configure the network adapter to register the network adapter's IP address using the suffix listed in **DNS Suffix**. If the check box is: - Selected, the IP address of the network adapter is registered using the suffix listed in **DNS Suffix** - Cleared, the IP address of the network adapter is not registered using the suffix listed in **DNS Suffix** By default, this check box is cleared.

Table 84 lists the configuration settings for the **WINS** tab of the **Network Settings** dialog box. Configure the settings in Table 84 based on the requirements of your organization, and then select **OK**.

Table 84. Configuration Settings on the WINS tab of the Network Settings Dialog Box

Expand table

Setting	Description
WINS server addresses in order of use	Contains a list of WINS server IP addresses to be configured for the network adapter. Select one of the following to modify the list of WINS servers: - **Add**. Add the WINS server IP address to the box. - **Properties**. Modify the WINS server IP address for an existing entry. - **Delete**. Delete an existing entry.
Enable LMHOSTS lookup	Select to configure the network adapter to enable lookups in the LMHOSTS file on the target computer. If the check box is: - Selected, the LMHOSTS file is used for Network BIOS (NetBIOS) name resolution on the network adapter - Cleared, the LMHOSTS file is not used for NetBIOS name resolution on the network adapter

Setting	Description
	By default, this check box is cleared.
Default	Select to configure the network adapter to use the default NetBIOS over TCP/IP settings of the target operating system. The other options are **Enable NetBIOS over TCP/IP** and **Disable NetBIOS over TCP/IP**.
Enable NetBIOS over TCP/IP	Select to enable NetBIOS over TCP/IP for the network adapter. The other options are **Default** and **Disable NetBIOS over TCP/IP**.
Disable NetBIOS over TCP/IP	Select to disable NetBIOS over TCP/IP for the network adapter. The other options are **Default** and **Enable NetBIOS over TCP/IP**.

Configure Server Role Task Sequence Steps for LTI

LTI can help automate the deployment of server roles in Windows Server. Configure LTI task sequence steps to deploy the supported server roles, which include:

- AD DS
- DNS Server
- DHCP Server

  The process for configuring the server role task sequence steps is similar for LTI and ZTI. For more information about configuring server role task sequence steps for LTI, see Configuring Server Role Task Sequence Steps.

Configure the Check BIOS Task Sequence Step for the List of Incompatible BIOS Versions

The default task order for a task sequence includes the **Check BIOS** task in the **Non-Replace** group in the **Validation** group. The **Check BIOS** task runs the ZTIBIOSCheck.wsf script, which checks the BIOS version of the target computers against the list of incompatible BIOS versions in the ZTIBIOSCheck.xml file.

Modify the ZTIBIOSCheck.xml file to contain the list of BIOS versions incompatible with the target operating system in the operating system build. Modify ZTIBIOSCheck.xml for each task sequence created in the Deployment Workbench. The ZTIBIOSCheck.xml file resides in the *deployment_share*\Scripts folder (where *deployment_share* is the name of the folder that is the root of the deployment share).

Update the attributes in the ZTIBIOSCheck.xml file listed in Table 85 for each incompatible BIOS version.

Table 85. Attributes to Update in the ZTIBIOSCheck.xml File

Expand table

Attribute	Description
Description	Text description of the target computer that has an incompatible BIOS version, including any references to possible updates to the BIOS version
Computer Manufacturer	Manufacturer of the target computer with an incompatible BIOS version (This value must match the value stored in the BIOS.)
Model	Model of the target computer with an incompatible BIOS version (This value must match the value stored in the BIOS.)
Date	Date of the BIOS of the target computer with an incompatible BIOS version (This value must match the value stored in the BIOS.)

The ZTIBIOSCheck.xml file contains the source for a script (ZTIBIOS_Extract_Utility.vbs) that can help extract the attributes listed in Table 85.

To use the ZTIBIOS_Extract_Utility.vbs utility to extract attributes on a target computer

1. Start Microsoft Notepad.
2. Copy the script source from ZTIBIOSCheck.xml file into Notepad.
3. Save the script source in Notepad as ZTIBIOS_Extract_Utility.vbs.
4. Run ZTIBIOS_Extract_Utility.vbs on a target computer that has an incompatible BIOS.
5. Update ZTIBIOSCheck.xml to include the BIOS based on the attributes retrieved in the previous steps.

Running the Deployment Wizard

To initiate the deployment of Windows to target computers, run the Deployment Wizard. Initiate the Deployment Wizard manually or by using Windows Deployment

Services. Each deployment scenario (Replace Computer, New Computer, or Refresh Computer) uses a different process. Initiate the deployment from Windows Deployment Services, a network share, from local drives, or using a DVD. The deployment process prompts for any configuration settings not already specified.

The Deployment Wizard is run just prior to the Deployment Wizard and is responsible for initializing the environment. The Deployment Wizard displays wizard pages based on the deployment scenario you selected and the configuration options you specified in CustomSettings.ini. The logic for displaying (or not displaying) a wizard page is noted for each wizard page in the following steps.

To deploy the new operating system to the target computer using the Deployment Wizard

1. Verify that the folders in Table 86 no longer exist on the target computer.

   The Deployment Wizard creates and uses the folders listed in Table 86 during the deployment process. If a previous deployment was terminated because of an error, these folders might still exist on the target computer. If these folders are not removed, the process will continue from the point where the previous process terminated instead of starting from the beginning. Remove these folders before initiating a new deployment. These folders exist on *target_drive* (where *target_drive* is the drive on the target computer that has the most available disk space).

Table 86. Folders to Remove Before Initiating the Deployment Wizard

Expand table

Folder	Description
target_drive:\MININT	This folder is preserved through the deployment process and contains deployment state information (such as user state migration information and log files).
target_drive:_*SMSTaskSequence*	This folder contains state information specific to the Task Sequencer.

2. Initiate the Deployment Wizard.

   a. Start the target computer with LTI bootable media from a CD, DVD, removable device, or Windows Deployment Services.

b. Connect to the appropriate deployment share (for example, \\server_name\Distribution$\Scripts), and then type **cscript litetouch.vbs**.

3. The Deployment Wizard starts. This wizard is responsible for initialing the Windows PE environment prior to running the Deployment Wizard. The Initializing wizard page appears, which displays the initialization status. The Windows Deployment wizard connects to the deployment share and displays the **Welcome** wizard page when the conditions in Table 87 are met.

Table 87. Welcome Wizard page Conditions

Expand table

Property	Condition
DeployRoot	Begins when the **DeployRoot** property equals " " (blank) or **X:**. Typically, the Windows Deployment console automatically sets this value.
SkipBDDWelcome	Not equal to "YES"

The **Welcome** wizard page in the Deployment Wizard is displayed.

4. On the **Welcome** page, select one of the options listed in Table 88 based on the environment's requirements. To proceed with the Deployment Wizard, select **Run the Deployment Wizard to install a new Operating System**.

Table 88. Welcome Wizard Page Options

Expand table

Option	Select this option to
Run the Deployment Wizard to install a new Operating System	Start the Deployment Wizard and complete the LTI deployment process.
Run the Windows Recovery Wizard	Open the **System Recovery Options** dialog box in Windows Recovery Environment. From here, run the Recovery Wizard to help repair an existing Windows installation.
Exit to Command Prompt	Open a Command Prompt window in Windows PE for troubleshooting prior to completing the LTI deployment process.

Option	Select this option to
Keyboard Layout	Select the keyboard layout for Windows PE prior to completing the LTI deployment process.
Configure with Static IP Address	Configure the IP configuration settings for Windows PE when DHCP configuration is not available. Select this button to open the **Configure Static IP Network Settings**dialog box. The IP configuration settings configured in the **Configure Static IP Network Settings** dialog box override any IP configuration settings specified in the CustomSettings.ini file or in the MDT DB.

The **Specify credentials for connecting to network shares** wizard page in the Deployment Wizard is displayed.

5. Complete the **Specify credentials for connecting to network shares** page, and then select **OK**.

   Complete the **Specify credentials for connecting to network shares** page, and then select **OK**.

   This wizard appears when the conditions in Table 89 are met.

Table 89. Specify Credentials for Connecting to Network Shares Page Conditions

Expand table

Property	Condition
UserID_isDirty	Equal to TRUE
UserID	Equal to ""
DeploymentType	Not equal to REPLACE
DeploymentMethod	Not equal to MEDIA

You use these credentials to access network shared folders used during the deployment process. These shared folders include folders used to store user state migration data or image backups of the target computer.

Table 90. Credentials Options

Expand table

Option	Type
User Name	***user_name*** (where *user_name* is the user name of the account that has the appropriate permissions for the network shared folders that the deployment scripts use).
Password	***password*** (where *password* is the password for the user account specified in the **User Name** box).
Domain	domain (where domain is the name of the domain in which the user account, specified in the User Name box, is located).

The **Task Sequence** page is displayed.

6. On the **Task Sequence** page, beneath **Select a task sequence to execute on this computer**, select ***task_sequence*** (where *task_sequence* is the name of the task sequence to run).

> **Note**
>
> If a task sequence based on the Litetouch OEM task sequence template appears in the list, ensure that the deployment is being performed from a removable media (MEDIA) deployment share. Although task sequences based on the Litetouch OEM task sequence template can be selected from other deployment shares, the task sequence will not finish successfully.

7. On the **Computer Details** page, in the **Computer name** box, type ***computer_name*** (where *computer_name* is the computer name to assign to the target computer), select one of the options listed in Table 92 based on your environment's requirements and then select **Next**.

   This wizard appears when the conditions in Table 91 are met.

Table 91. Configure the Computer Details Page Conditions

Expand table

Property	Condition
SkipComputerName	Not equal to YES

Property	Condition
SkipDomainMembership	Not equal to YES
DeploymentType	Not equal to StateRestore or REPLACE or CUSTOM
DeploymentType	Not equal to REPLACE

Table 92. Specify Computer Membership Configuration

Expand table

For this option	Perform this task to
Join a domain	Join an existing AD DS domain: - In the **Domain to join** box, type ***domain*** (where *domain* is the name of the domain to be joined). - In the **Organizational Unit** box, type ***organizational_unit*** (where *organizational_unit* is the name of the organizational unit [OU] in the AD DS domain in which the computer account will be created). - In the **User Name** box, type ***user_name*** (where *user_name* is the name of a user account that has sufficient permissions to create the computer account in the domain). - In the **Password** box, type ***password*** (where *password* is the password for the user account specified in the **User Name** box). - In the **Domain** box, type ***domain*** (where *domain* is the name of the domain in the user account specified in the **User Name** box is located).
Join a workgroup	Join a Windows workgroup: - In the **Workgroup** box, type ***workgroup*** (where *workgroup* is the name of the workgroup to join).

The **User Data** page is displayed.

8. On the **User Data** page, select one of the options listed in Table 94 based on your environment's requirements, and then select **Next**.

   This wizard appears when the conditions in Table 93 are met.

Table 93. User Data Page Conditions

Expand table

Property	Condition
SkipUserData	**Not equal to YES**
DeploymentType	Not equal to REFRESH or REPLACE or StateRestore
ImageFlags	Does not contain SERVER
IsServerOS	Not equal to TRUE

Table 94. User Data Page Options

Expand table

Option	Select this option to
Automatically determine the location	- Allow the MDT scripts and process rules to automatically determine the best location based on local available disk space on the target computer. - Optionally, select the **Allow data and settings to be stored locally when possible** check box to give preference to storing the data locally.
Specify a location	Save the user state migration data to a specific location. In the Location box, type location (where location is the fully qualified path to the locations for storing the user state migration data). Alternatively, select **Browse** to go to the location.
Do not save data and settings	Discard any existing user state migration data or deploy a new computer with no existing data.

The **Move Data and Settings** page is displayed.

9. On the **Move Data and Settings** page, select one of the options listed in Table 96 based on your environment's requirements, and then select **Next**.

This wizard appears when the conditions in Table 95 are met.

Table 95. Move Data and Settings Page Conditions

Expand table

Property	Condition
SkipUserData	Not equal to YES
DeploymentType	Equal to NEWCOMPUTER
ImageFlags	Does not contain SERVER
IsServerOS	Not equal to TRUE
FindTaskSequenceStep	Task sequence includes **Offline User State Capture** and **ZTIUserState.wsf**

Table 96. Move Data and Settings Page Options

Expand table

Option	Select this option to
Do not move user data and settings	Install the new operating system without capturing any user data or settings from any existing operating systems. This option sets the **USMTOfflineMigration** property to "FALSE". Optionally, select the **Keep existing partitions**check box to give preference to storing the data locally. If the check box is: - **Selected**, then the new operating system is installed on the existing partitions. This option sets the **DoNotFormatAndPartition** property to "YES". - **Cleared**, then the target computer is partitioned and formatted and the target operating system is installed on the new partition structure. This option does not set the **DoNotFormatAndPartition** property. The **Keep existing partitions**check box is disabled for the MDT New Computer deployment scenario because no existing partitions exist.
Move user data and settings	Capture the user data and settings from an existing operating system and restore them into the new operating system. This option sets the: - **DoNotFormatAndPartition** property to "YES". - **USMTOfflineMigration** property to "TRUE". Optionally, if the target computer has multiple operating systems installed, the **Select the partition that contains the operating system and data** group of options is displayed. There is an option for each partition that contains an

Option	Select this option to
	operating system. Select the partition that contains the user state migration data that you want to capture.

The **User Data (Restore)** page is displayed.

10. On the **User Data (Restore)** page, select one of the options listed in Table 98 based on your environment's requirements, and then select **Next**.

    This wizard appears when the conditions in Table 97 are met.

Table 97. User Data (Restore) Page Conditions

Expand table

Property	Condition
SkipUserData	Not equal to YES
DeploymentType	Equal to NEWCOMPUTER or StateRestore and not equal to CUSTOM
ImageFlags	Does not contain SERVER
IsServerOS	Not equal to TRUE

Table 98. User Data (Restore) Options

Expand table

Option	Select this option if
Do not restore user data and settings	The migration type is New Computer and there is no user state migration data to restore.
Specify a location	The migration type is Replace Computer.In the **Location** box, type ***location*** (where *location* is the fully qualified path to the location in which the user state migration back files are stored).

11. On the **Computer Backup** page, select one of the options listed in Table 100 based on requirements, and then select **Next**.

    This wizard appears when the conditions in Table 99 are met.

Table 99. Computer Backup Page Conditions

Expand table

Property	Condition
SkipComputerBackup	Not equal to YES
DeploymentType	Not equal to NEWCOMPUTER and not equal to CUSTOM and not equal to STATERESTORE and equal to REFRESH or equal to REPLACE

Table 100. Computer Backup Options

Expand table

Option	Select this option to
Automatically determine the location	- Allow the MDT scripts and process rules to automatically determine the best location based on local available disk space on the target computer. - Optionally, select the **Allow data and settings to be stored locally when possible** check box to give preference to storing the data locally.
Specify a location	Save the computer image backup to a specific location. In the **Location** box, type ***location*** (where *location* is the fully qualified path to the locations for storing the computer backup).
Do not back up the existing computer	Discard any existing data on the target computer or deploy a new computer with no existing data.

The Deployment Wizard uses the ImageX utility to perform the backup. ImageX is not intended to be used as a part of the overall backup and disaster recovery process. Instead, it is designed to create a backup of the target computer to assist in recovering user state migration information that might not have been captured correctly.

> **Note**
>
> MDT uses the ImageX utility during migration because it works on all platforms that MDT supports. Use tools such as Windows Backup for enhanced disaster recovery protection after migration is complete.

12. On the **Product Key** page, in the **Product key** box, type ***product_key*** (where *product_key* is the product key to be assigned to the target computer), and then select **Next** (see Table 102).

    This wizard appears when the conditions in Table 101 are met.

Table 101. Product Key Page Conditions

Expand table

Property	Condition
SkipProductKey	Not equal to YES.
DeploymentType	Not equal to REPLACE and not equal to CUSTOM and not equal to StateRestore.

Table 102. Product Key Options

Expand table

Option	Select this option to
No product key is required	Assign product keys to target computers using a KMS key.
Activate the machine with a Multiple Activation Key (MAK)	Assign a MAK to the target computer and activate the computer over the Internet. In the **Multiple activation key** box, type ***mak*** (where *mak* is the MAK to be assigned to the target computer).
Use a specific product key	Assign a specific license key for installation or retail activation. In the **Product_key** box, type ***product_key*** (where *product_key* is the product key to be assigned to the target computer).

13. On the **Language Packs** page, in the **Select the language packs to install** box, select the check box next to ***language_pack*** (where *language_pack* is the language pack to be installed), and then select **Next**.

> **Tip**

You can select multiple language packs by selecting multiple check boxes that correspond to the language packs.

This wizard appears when the conditions in Table 103 are met.

Table 103. Language Packs Page Conditions

Expand table

Property	Condition
SkipPackageDisplay	Not equal to YES
DeploymentType	Not equal to REPLACE or CUSTOM
ImageBuild	The first character in the property is equal to 6 (which indicates the Windows 7, Windows 8, or Windows 8.1 build), and there are active packages to display to the user.

For Ultimate, Enterprise, and all editions of Windows Server you can select more than one language. For all other Windows editions, select only one language.

14. On the **Locale and Time** page, select the appropriate values for each option listed in Table 105 based on your requirements, and then select **Next**.

    This wizard appears when the conditions in Table 104 are met.

Table 104. Locale and Time Page Conditions

Expand table

Property	Condition
SkipLocaleSelection	Not equal to YES.
DeploymentType	Not equal to REPLACE or CUSTOM.
ImageBuild	The first character in the property is equal to 6 (which indicates the Windows 7, Windows 8, or Windows 8.1 build).

Table 105. Locale and Time Options

Expand table

In this box	Select
Language to install	The default language for the target operating system.
Time and currency format (Locale)	The default locale for the target operating system.
Keyboard layout	Keyboard layout to be used with the target operating system.
Time zone	The time zone in which the target computer is located.

15. On the **Roles and Features** page, select the appropriate values for each option listed in Table 107 based on your requirements, and then select **Next**.

    This wizard appears when the conditions in Table 106 are met.

Table 106. Roles and Features Page Conditions

Expand table

Property	Condition
SkipRoles	Not equal to YES.
FindTaskSequenceStep	Task sequence includes **BDD_InstallRoles** and ZTIOSRole.wsf
OSVersion	Not equal to WinPE.
ImageBuild	Not Null.

Table 107. Roles and Features Options

Expand table

In this box	Select
The following roles and role services are available	The check boxes that correspond to the desired Windows roles, role services, or features.
Select All	Select this button to select all the check boxes associated with the Windows roles, role services, or features.
Select None	Select this button to deselect all the check boxes associated with the Windows roles, role services, or features.

16. On the **Applications** page, select the check box next to ***application_name*** (where *application_name* is the name of the application you want to deploy, and then select **Next**.

    > **Tip**
    >
    > You can select multiple applications by selecting multiple check boxes that correspond to the applications.

    This wizard appears when the conditions in Table 108 are met.

Table 108. Applications Page Conditions

Expand table

Property	Condition
SkipApplications	Not equal to YES
DeploymentType	Not equal to REPLACE
IsThereAtLeastOneApplicationPresent	Greater than one

17. On the **Administrator Password** page, in the **Administrator Password** and **Confirm Administrator Password** boxes, type ***password*** (where *password* is the password for the local built-in Administrator account on the target computer), and then select **Next**.

    This wizard page appears when the conditions in Table 109 are met.

Table 109. Administrator Password Page Conditions

Expand table

Property	Condition
SkipAdminPassword	Not equal to YES
DeploymentType	Not equal to REPLACE or CUSTOM
TaskSequenceTemplate	Not equal LTIOEM.XML

18. On the **Local Administrators** page, in the **Administrator Accounts**box, type ***admin_accounts*** (where *admin_accounts* are the accounts that you want to add to

the local built-in Administrator account on the target computer), and then select **Next**.

This wizard page appears when the conditions in Table 109 and Table 110 are met.

Table 110. Local Administrators Page Conditions

Expand table

Property	Condition
SkipAdminAccounts	Not equal to YES
DeploymentType	Not equal to REPLACE and not equal to CUSTOM
JoinDomain	Not equal to ""

> **Note**
>
> Unlike other Deployment Wizard pages, the **Administrator Accounts** page is skipped by default, because the default value for the **SkipAdminAccount** property is **YES**. For more information, see the **SkipAdminAccounts** property in the MDT document *Toolkit Reference*.

19. On the **Capture Image** page, select one of the options listed in Table 112 based on requirements, and then select **Next**.

    This wizard appears when the conditions in Table 111 are met.

Table 111. Capture Image Page Conditions

Expand table

Property	Condition
SkipCapture	Not equal to YES
DeploymentType	Not equal to REFRESH or not equal to REPLACE
JoinDomain	Equal to ""

Table 112. Capture Image Options

Expand table

Option	Select this option to
Capture an image of this reference computer	Run Sysprep, and then capture an image of the target computer. Then, store the image in the location specified. In the **Location** box, type ***location*** (where *location* is the fully qualified path to the location for storing the image of the target computer).In the **File name** box, type ***file_name*** (where *file_name* is the name of the image file).
Sysprep this computer	Copy the required Sysprep files to the target computer, initiate Sysprep, but do not capture an image of the target computer.
Prepare to capture the machine	Copy the required Sysprep files to the target computer, but do not initiate Sysprep.
Do not capture an image of this computer	Deploy the target operating system to the target computer without capturing a Sysprep image of the computer.

20. On the **BitLocker** page, select one of the options listed in Table 114 based on your environment's requirements, and then select **Next**.

    This wizard appears when the conditions in Table 113 are met.

Table 113. BitLocker Page Conditions

Expand table

Property	Condition
SkipBitLocker	Not equal to YES.
DeploymentType	Equal to REPLACE or CUSTOM.
DoCapture	Not equal to YES or not equal to PREPARE.
ImageBuild	The first character in the property is equal to 6 (which indicates an operating system that is Windows Vista or later).
ImageFlags	Equal to ENTERPRISE or ULTIMATE.

Table 114. BitLocker Configuration Options

Expand table

Option	Select this option to
Do not enable BitLocker for this computer	Deploy the new operating system without activating BitLocker.
Enable BitLocker	Activate BitLocker and use TPM version 1.2 or later. Then, select one of the following options for using TPM: - To use TPM only, select **Enable BitLocker using TPM only**. - To use TPM with a PIN, select **Enable BitLocker using TPM and a PIN**; in the **Pin** box, type ***pin*** (where *pin* is the BitLocker PIN for the target computer). The value provided can be numeric only or alphanumeric depending on the value of the **BDEAllowAlphaNumericPin** property. - To use TPM with a startup key, select **Enable BitLocker using TPM and a startup key**; in the box, select the drive on which the startup key resides. - To use only an External Startup Key, select **Enable BitLocker using only an External Startup Key**; in the box, select the drive on which the external startup key resides. - To store the recovery key in AD DS, under **Choose where to store the Recovery Key**, select **In Active Directory**. - To not create a recovery key, under **Choose where to store the Recovery Key**, select **Do not create a recovery key**. - To configure the deployment process to wait until encryption is complete on all drives before continuing, select the **Wait for BitLocker Encryption to complete on all drives before continuing** check box.

> ⓘ **Note**
>
> The default setting for BitLocker is disabled.

21. Review the information on the **Ready to begin** page, and then select **Begin**.

> ⓘ **Note**
>
> To expand the details of this deployment, select **Details**.

The Deployment Wizard closes, and deployment of the new operating system begins.

Performing ZTI Deployments Using Configuration Manager

You perform ZTI deployments using Configuration Manager and MDT within an AD DS domain, within a Windows workgroup, or from removable media. Perform ZTI deployments by:

- Preparing the ZTI deployment environment as described in Preparing the ZTI Deployment Environment for Configuration Manager
- Preparing for ZTI deployment to the reference computer as described in Preparing for ZTI Deployment to the Reference Computer Using Configuration Manager
- Deploying to and capturing an image of the reference computer in ZTI as described in Deploying To and Capturing an Image of the Reference Computer Using Configuration Manager
- Preparing for ZTI deployment to the target computers as described in Preparing for ZTI Deployment to Target Computers Using Configuration Manager
- Deploying captured images to the target computer in ZTI as described in Deploying Captured Images to Target Computers Using Configuration Manager

Preparing the ZTI Deployment Environment for Configuration Manager

After you have prepared the prerequisite infrastructure for MDT, you are ready to prepare the MDT deployment environment for ZTI.

To prepare the MDT deployment environment for ZTI deployments

1. Preparing the prerequisite infrastructure as described in Prepare the Prerequisite ZTI Infrastructure for Use with Configuration Manager.
2. Install a new instance of MDT on the deployment server, or upgrade an existing instance of MDT to MDT as described in Install or Upgrade to MDT for the ZTI Deployment Process Using Configuration Manager.

3. Obtain the software that ZTI requires as described in Obtain the Software That the ZTI Deployment Process Using Configuration Manager Requires.

4. Enable Configuration Manager console integration with MDT as described in Enable Configuration Manager Console Integration for Configuration Manager.

Prepare the Prerequisite ZTI Infrastructure for Use with Configuration Manager

ZTI deployments using Configuration Manager require that a properly configured Configuration Manager infrastructure exist prior to installing MDT and performing deployments. Ensure that your new or existing Configuration Manager infrastructure is specifically optimized for the Operating System Deployment feature.

> **Note**
>
> Windows PowerShell version 2.0 or later must be installed on any computer on which MDT is installed for management of ZTI deployments.

For more information about:

- Hardware and software requirements for Configuration Manager, see Supported Configurations for Configuration Manager
- Configuring a Configuration Manager infrastructure to support ZTI deployments, see the section, "Step 1: Prepare the Prerequisite Infrastructure", in the MDT document *Quick Start Guide for Microsoft System Center 2012 R2 Configuration Manager*.

Install or Upgrade to MDT for the ZTI Deployment Process Using Configuration Manager

The first step in performing ZTI deployments is to have at least one instance of MDT running in your environment. Install MDT on each computer that has the Configuration Manager console installed and that you will use to create or edit task sequences that MDT generates. If your existing environment has:

- No computers currently running MDT or a previous version of MDT, install one or more new instances of MDT as described in Installing a New Instance of MDT.
- One or more computers running a previous version of MDT, upgrade those instances to MDT as described in Upgrading to MDT. After the upgrade process is

complete:

- **Run the Configure ConfigMgr Integration Wizard**. This wizard must be run after the upgrade to register the new components and install the ZTI new task sequence templates.
- **Run the Remove PXE Filter Wizard**. If you had previously installed and configured the PXE filter to support the unknown computer capability in previous versions of MDT. This support is now provided in Configuration Manager and has been removed in MDT.
- **Ensure you create a new Microsoft Deployment Toolkit Files package for any new ZTI task sequences you create**. The existing Microsoft Deployment Toolkit Files package can be used for any ZTI task sequences created prior to the upgrade, but a new Microsoft Deployment Toolkit Files package must be created for new ZTI task sequences.
- **Ensure any ZTI task sequences created prior to the upgrade use the Microsoft Deployment Toolkit Files package that existed prior to the upgrade**. You can modify these ZTI task sequences, but you cannot use any of the new MDT task sequence actions or steps. To use the new MDT task sequence actions or steps, create a new ZTI task sequence.

  > ⓘ **Note**
  >
  > If you upgraded from a previous version of Configuration Manager, you can use ZTI task sequences for MDT that were created in the previous version of Configuration Manager as long as they were created using the same version of MDT.

Obtain the Software That the ZTI Deployment Process Using Configuration Manager Requires

Collect the software needed during the ZTI deployment process for Configuration Manager. This software will be imported or added to deployment shares unless it already exists in the deployment share.

> ⓘ **Note**
>
> MDT supports the Windows ADK for Windows 8.1, Windows PE 5.0, and System Center 2012 R2 Configuration Manager.

Required software includes:

- Operating system source files for each operating system to be deployed to the reference and target computers
- Operating system packages for the operating systems, such as security updates, feature packs, and language packs
- Device drivers for the reference and target computers that are not included as part of the operating system
- Applications that are to be installed as a part of the operating system image or during the deployment of the reference image
- USMT source files used to create a software package that is deployed to the target computers to capture user state migration data

Enable Configuration Manager Console Integration for Configuration Manager

Before you can use the Configuration Manager integration features of MDT, run the Configure ConfigMgr Integration Wizard. This wizard copies the appropriate Configuration Manager integration files to the *Configuration Manager_root* (where *Configuration Manager_root* is the folder in which the Configuration Manager console is installed).

The wizard also adds WMI classes for the new MDT custom actions. You add these classes by compiling a Managed Object Format (.mof) file that contains the new class definitions.

To run the Configure ConfigMgr Integration Wizard

1. > ⓘ **Note**
   >
   > The Configuration Manager console should be closed when performing this procedure.

   Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Configure ConfigMgr Integration**.

   The Configure ConfigMgr Integration Wizard starts.

2. Complete the Configure ConfigMgr Integration Wizard using the information in Table 115. Accept the default values unless otherwise specified.

Table 115. Information for Completing the Configure ConfigMgr Integration Wizard

Expand table

On this wizard page	Do this
Options	- Verify that the **Install the MDT extensions for Configuration Manager** option is selected. - Verify that the **Install the MDT console extensions for System Center 2012 R2 Configuration Manager** check box is selected. - Verify that the **Add the MDT task sequence actions to a System Center 2012 R2 Configuration Manager server** check box is selected. - In **Site Server Name**, type ***ConfigMgr_server_name*** (where *ConfigMgr_server_name* is the name of the Configuration Manager server on which to install MDT integration). - In **Site Code**, type ***ConfigMgr_site_code*** (where *ConfigMgr_site_code* is the Configuration Manager site code that installs MDT integration), and then select **Next**. The Site Server Name and Site Code fields will be automatically populated with the most recent connection details if the Configuration Manager console has been opened once.
Confirmation	Review the completion status of the wizard, and then select **Finish**.

When the wizard finishes, the Configuration Manager console is configured for MDT integration.

Preparing for ZTI Deployment to the Reference Computer Using Configuration Manager

Regardless of the ZTI deployment scenario you are performing, always start by creating a reference computer, and then capturing an image of that computer. Later in the ZTI deployment process, you will deploy the captured image of your reference computer to the appropriate target computers.

Create a reference computer for each image that you want to create for deployment to the target computers. For more information about determining the number of images required in your organization (and subsequently the number of reference computers required), see Estimate Storage Requirements for Configuration Manager Distribution Points. For more information about the use of reference computers in MDT deployments, see Using Reference Computers in MDT Deployments.

To prepare for deployment to the reference computer using Configuration Manager

1. Create a new task sequence that will deploy the target operating system to the reference computer using the Create MDT Task Sequence Wizard in the Configuration Manager console as described in Creating a ZTI Task Sequence Using MDT Task Sequence Templates in Configuration Manager.

   > **Tip**
   >
   > Create the task sequence for deploying to the reference computer based on the Client Task Sequence or Server Task Sequence template included in MDT.

2. Configure Configuration Manager to contain the appropriate software for deployment to the reference computer, including the following:

   - Configuring applications and operating system packages as described in Managing Software Packages in Configuration Manager
   - Configuring device drivers as described in Managing Device Drivers in Configuration Manager

3. Ensure that the distribution points for the packages and operating system images that the new ZTI task sequence uses are distributed properly as described in Managing Distribution Points in Configuration Manager.

   > **Note**
   >
   > Most production networks have multiple distribution points. When performing this step in a production environment, select the appropriate distribution points for the network.

4. Customize the MDT configuration files to the needs of your organization as described in Configuring MDT Deployments.

> **Important**
>
> If you are capturing an image of the reference computer, you must at least add the **DoCapture** property to the Customsettings.ini file for the task sequence by specifying `DoCapture=YES or DoCapture=SYSPREP`.

5. Optionally, enable monitoring of the MDT deployment process as described in Monitoring MDT Deployments.

6. Customize the task sequence to the needs of your organization as described in Configuring ZTI Task Sequence Steps in Configuration Manager.

   > **Note**
   >
   > The ZTI deployment process is unable to perform Sysprep operations on a target computer that is encrypted by using BitLocker Drive Encryption. Do not enable BitLocker on the reference computer, and enable BitLocker on the target computers only after the target operating system is completely deployed.

7. Update the distribution points so that any changes to the packages are distributed properly as described in Managing Distribution Points in Configuration Manager.

   > **Note**
   >
   > Most production networks have multiple distribution points. When performing this step in a production environment, select the appropriate distribution points for the network.

Deploying To and Capturing an Image of the Reference Computer Using Configuration Manager

After the distribution points are updated, advertise the task sequence to the reference computer and start the reference computer with the bootable Windows PE image created earlier in the process. The task sequence created earlier will deploy the target operating system, device drivers, operating system packages, and applications to the reference computer, and then capture an image of the reference computer.

To deploy to and capture an image of the reference computer

1. Add the reference computer to the Configuration Manager site database as described in Manually Adding Computers to the Site Database in Configuration Manager.

2. Create a collection that contains the reference computer as described in Managing Computer Collections in Configuration Manager.

3. Deploy the task sequence to the reference computer as described in Managing Task Sequence Deployment in Configuration Manager.

4. Create a task sequence bootable media disk by using the Task Sequence Media Wizard as described in Creating Task Sequence Bootable Media in Configuration Manager.

5. Start the reference computer with the task sequence bootable media disk as described in Deploying an Operating System Using Task Sequence Bootable Media in Configuration Manager.

6. Optionally, monitor the deployment process using the Monitoring node in the Deployment Workbench or using the **Get-MDTMonitorData** cmdlet.

Preparing for ZTI Deployment to Target Computers Using Configuration Manager

After the images of the reference computers are captured, deploy them to the target computers. In preparation for deploying the captured images to the target computers, create one or more task sequences for deploying the captured images, ensure that the necessary deployment resources exist, and customize the MDT deployment process.

To prepare for ZTI deployment to target computers

1. Prepare network shares for storing migration data and MDT deployment logs as described in Preparing the MDT Migration Resources.

2. Optionally, prepare Windows Deployment Services to start the appropriate Windows PE images that will in turn start the ZTI deployment process to the target computers as described in Preparing Windows Deployment Services for ZTI Deployments Using Configuration Manager.

3. Create additional distribution points to help in larger deployments as described in Managing Distribution Points in Configuration Manager.

4. Prepare the ZTI task sequences, the MDT configuration files, and the MDT DB for each deployment scenario as described in the following:

   - Prepare for the ZTI New Computer Deployment Scenario to Target Computers Using Configuration Manager
   - Prepare for the ZTI Refresh Computer Deployment Scenario to Target Computers Using Configuration Manager
   - Prepare for the ZTI Replace Computer Deployment Scenario to Target Computers Using Configuration Manager

   Depending on the target computers in your organization, any combination of these deployments scenarios might be necessary. For more information about MDT deployment scenarios, see Identifying Deployment Scenarios.

Prepare for the ZTI New Computer Deployment Scenario to Target Computers Using Configuration Manager

In the New Computer deployment scenario, you deploy a new installation of a Windows operating system to a new computer. There is no user migration information to save and restore and no existing file systems to preserve. Use the Client Task Sequence template to deploy the captured image of the reference computer to the target computer.

To prepare for the New Computer deployment scenario to target computers

1. Create a new task sequence that will deploy the target operating system to the reference computer using the Create MDT Task Sequence Wizard in the Configuration Manager console as described in Creating a ZTI Task Sequence Using MDT Task Sequence Templates in Configuration Manager, but ensure that you specifically follow the configuration settings on the wizard pages listed in Table 116 and select the appropriate values on the other wizard pages based on your organization's requirements.

   > **Tip**
   >
   > Create the task sequence for deploying to the reference computer based on the Client Task Sequence or Server Task Sequence template included in MDT.

Table 116. Information for Completing the Create MDT Task Sequence Wizard for Performing New Computer Deployment Scenario Using ZTI

Expand table

On this wizard page	Do this
OS Image	Select **Create a new OS image**, and specify the fully qualified UNC path to the WIM image captured from the reference computer.
Deployment Method	Select **Perform a "Zero Touch Installation" OS deployment, with no user interaction**.

2. Configure Configuration Manager to contain the appropriate software for deployment to the target computer, including:
    - Configuring applications and operating system packages as described in Managing Software Packages in Configuration Manager
    - Configuring device drivers as described in Managing Device Drivers in Configuration Manager
3. Customize the MDT configuration files to the needs of your organization as described in Configuring MDT Deployments.
4. Optionally, customize the MDT DB to the needs of your organization as described in Performing Deployments Using the MDT DB (if you are using the MDT DB to provide MDT configuration information).
5. Optionally, enable monitoring of the MDT deployment process as described in Monitoring MDT Deployments.
6. Customize the task sequence to the needs of your organization as described in Configuring ZTI Task Sequence Steps in Configuration Manager.
7. Ensure that the distribution points for the packages and operating system images that the new ZTI task sequence uses are distributed properly as described in Managing Distribution Points in Configuration Manager.

> **Note**
>
> Most production networks have multiple distribution points. When performing this step in a production environment, select the appropriate

distribution points for the network.

8. Update the distribution points so that any changes to the packages are distributed properly as described in Managing Distribution Points in Configuration Manager.

> **Note**
>
> Most production networks have multiple distribution points. When performing this step in a production environment, select the appropriate distribution points for the network.

Prepare for the ZTI Refresh Computer Deployment Scenario to Target Computers Using Configuration Manager

In the Refresh Computer deployment scenario, a computer is refreshed, including computers that must be re-imaged for image standardization or to address a problem. There is user migration information to save and restore but no existing file systems to preserve. Use the Client Task Sequence template to deploy the captured image of the reference computer to the target computer.

To prepare for the Refresh Computer deployment scenario to target computers

1. Create a new task sequence that will deploy the target operating system to the reference computer using the Create MDT Task Sequence Wizard in the Configuration Manager console as described in Creating a ZTI Task Sequence Using MDT Task Sequence Templates in Configuration Manager, but ensure that you follow the configuration settings on the wizard pages listed in Table 117 and select the appropriate values on the other wizard pages for your organization's requirements.

> **Tip**
>
> Create the task sequence for deploying to the reference computer based on the Client Task Sequence or Server Task Sequence template included in MDT.

Table 117. Information for Completing the Create MDT Task Sequence Wizard for Performing New Computer

Deployment Scenario Using ZTI

Expand table

On this wizard page	Do this
OS Image	Select **Create a new OS image**, and specify the fully qualified UNC path to the WIM image captured from the reference computer.
Deployment Method	Select **Perform a "Zero Touch Installation" OS deployment, with no user interaction**.

2. Configure the appropriate software for deployment to the target computer in the Configuration Manager Console, including:
    - Configuring applications and operating system packages as described in Managing Software Packages in Configuration Manager
    - Configuring device drivers as described in Managing Device Drivers in Configuration Manager
3. Optionally, customize the MDT configuration files or the MDT DB to the needs of your organization as described in:
    - Configuring MDT Deployments
    - Performing Deployments Using the MDT DB
4. Optionally, enable monitoring of the MDT deployment process as described in Monitoring MDT Deployments.
5. Customize the task sequence to the needs of your organization as described in Configuring ZTI Task Sequence Steps in Configuration Manager.
6. Ensure that the distribution points for the packages and operating system images that the new ZTI task sequence uses are distributed properly as described in Managing Distribution Points in Configuration Manager.

> **Note**
>
> Most production networks have multiple distribution points. When performing this step in a production environment, select the appropriate distribution points for the network.

7. Update the distribution points so that any changes to the packages are distributed properly as described in Managing Distribution Points in Configuration Manager.

   > **Note**
   >
   > Most production networks have multiple distribution points. When performing this step in a production environment, select the appropriate distribution points for the network.

Prepare for the ZTI Replace Computer Deployment Scenario to Target Computers Using Configuration Manager

In the Replace Computer deployment scenario, one computer replaces another computer. Create a computer association record that associates the existing target computer and the new target computer. The existing user state migration data is saved from the existing target computer. Then, a new installation of Windows is deployed to a new computer. Finally, the user state data is restored to the new computer. There are no existing file systems to preserve.

> **Important**
>
> You must establish a computer association record for each existing target computer and each new target computer prior to performing the deployment to the target computer.

Use the:

- **Client Replace Task Sequence** template to save the user state migration of the existing target computer

  > **Important**
  >
  > Run this task sequence before running the task sequence based on the Client Task Sequence template on the new target computer.

- **Client Task Sequence** template to deploy the captured image of the reference computer to the new target computer and restore the user state migration data

  > **Important**

Run this task sequence after running the task sequence based on the Client Replace Task Sequence template on the existing target computer.

To prepare for the Replace Computer deployment scenario to target computers

1. Create a computer association between the existing target computer and the new target computer as described in the subsection, "How to Perform a Side-by-Side Operating System Deployment," in the section, "How to Deploy Operating Systems in Configuration Manager," in the Configuration Manager Documentation Library, which is installed with Configuration Manager.

2. Create a new task sequence that will save the user state migration data of the existing target computer as described in Creating a ZTI Task Sequence Using MDT Task Sequence Templates in Configuration Manager.

   > **Tip**
   >
   > Create the task sequence for capturing the user state migration data from the target computer based on the Client Task Replace Sequence template included in MDT.

3. Create a new task sequence that will deploy the captured image of the reference computer to the target computer, and restore the user state migration data saved by the **Client Replace Task Sequence** as described in Creating a ZTI Task Sequence Using MDT Task Sequence Templates in Configuration Manager, but ensure that you specifically follow the configuration settings on the wizard pages listed in Table 118 and select the appropriate values on the other wizard pages for your organization's requirements.

   > **Tip**
   >
   > Create the task sequence for deploying to the target computer based on the Client Task Sequence template included in MDT.

Table 118. Information for Completing the Create MDT Task Sequence Wizard for Performing the ZTI Replace Computer Deployment Scenario

Expand table

On this wizard page	Do this
OS Image	Select **Create a new OS image**, and specify the fully qualified UNC path to the WIM image captured from the reference computer.
Deployment Method	Select **Perform a "Zero Touch Installation" OS deployment, with no user interaction**.

4. Configure the appropriate software for deployment to the target computer in the Configuration Manager Console, including:
    - Configuring applications and operating system packages as described in Managing Software Packages in Configuration Manager
    - Configuring device drivers as described in Managing Device Drivers in Configuration Manager
5. Customize the MDT configuration files or the MDT DB to the needs of your organization as described in:
    - Configuring MDT Deployments
    - Performing Deployments Using the MDT DB
6. Optionally, enable monitoring of the MDT deployment process as described in Monitoring MDT Deployments.
7. Customize the task sequences to the needs of your organization as described in Configuring ZTI Task Sequence Steps in Configuration Manager.
8. Ensure that the distribution points for the packages and operating system images that the new ZTI task sequence uses are distributed properly as described in Managing Distribution Points in Configuration Manager.

    > **Note**
    >
    > Most production networks have multiple distribution points. When performing this step in a production environment, select the appropriate distribution points for the network.

9. Update the distribution points so that any changes to the packages are distributed properly as described in Managing Distribution Points in Configuration Manager.

> **Note**
>
> Most production networks have multiple distribution points. When performing this step in a production environment, select the appropriate distribution points for the network.

Deploying Captured Images to Target Computers Using Configuration Manager

The deployment of the captured images to the target computers is slightly different for each MDT deployment scenario using ZTI. Deploy the captured image of the reference computer to target computers for each respective deployment scenario in your organization.

To deploy the capture image of the reference computer to the target computers

1. Add the target computer to the Configuration Manager site database:
    - Manually, as described in Manually Adding Computers to the Site Database in Configuration Manager
    - Automatically as described in the section, "How to Manage Unknown Computer Deployments in Configuration Manager," in the Configuration Manager Documentation Library, which is installed with Configuration Manager
2. Create a collection that contains the target computers as described in Managing Computer Collections in Configuration Manager.

    > **Note**
    >
    > Create a collection for each MDT deployment scenario to be performed, and ensure that the collection includes the target computers requiring the corresponding deployment scenario.

3. Deploy the task sequence to the target computers as described in Managing Task Sequence Deployment in Configuration Manager.
4. Provide a method for starting the target computers by doing any combination of the following:

- Create a task sequence bootable media disk using the Task Sequence Media Wizard as described in Creating Task Sequence Bootable Media in Configuration Manager.

- Prepare Windows Deployment Services to start the appropriate Windows PE images that will in turn start the ZTI deployment process to the target computers as described in Preparing Windows Deployment Services for ZTI Deployments Using Configuration Manager.

5. Deploy the captured reference computer image to the target computers for each deployment scenario as described in:

   - Deploy Captured Images to Target Computers in the ZTI New Computer Deployment Scenario Using Configuration Manager

   - Deploy Captured Images to Target Computers in the ZTI Refresh Computer Deployment Scenario Using Configuration Manager

   - Deploy Captured Images to Target Computers in the Replace Computer Deployment Scenario Using Configuration Manager

     Depending on the target computers in your organization, any combination of deployments scenarios might be necessary. For more information about the MDT deployment scenarios, see Identifying Deployment Scenarios.

Deploy Captured Images to Target Computers in the ZTI New Computer Deployment Scenario Using Configuration Manager

Start the target computer with the task sequence bootable media created earlier in the process or from Windows Deployment Services. Either method starts Windows PE on the target computer and initiates the ZTI deployment process. At the end of the process, the captured image of the reference computer is deployed on the target computer.

To deploy the capture images to the target computers in the ZTI New Computer Deployment Scenario using Configuration Manager

1. Start the target computer with the task sequence bootable media created earlier in the process or from Windows Deployment Services.

   The Task Sequence Wizard starts.

2. Complete the Task Sequence Wizard, ensuring that you specifically follow the configuration settings on the wizard pages listed in Table 119 and select the appropriate values on the other wizard pages for your organization's requirements.

> **Note**
>
> This wizard will not appear if you configure ZTI to perform a PXE boot and have configured a mandatory advertisement or if only one task sequence is advertised to the target computer.

Table 119. Information for Completing the Task Sequence Wizard in the ZTI New Computer Deployment Scenario Using Configuration Manager

Expand table

On this wizard page	Do this
Select a Task Sequence	Select the task sequence you created for the target computer deployment for the New Computer deployment scenario.

The wizard starts, and the operating system deployment starts.

3. Optionally, view the MDT deployment process using the Monitoring node in the Deployment Workbench or using the **Get-MDTMonitorData** cmdlet.

   For more information about monitoring MDT deployments, see View MDT Deployment Progress.

Deploy Captured Images to Target Computers in the ZTI Refresh Computer Deployment Scenario Using Configuration Manager

Start ZTI by running the Configuration Manager task sequence deployment for capturing the user state migration data that you created earlier in the process. This task sequence runs in the current operating system on the existing target computer.

To deploy the capture images to the target computers in the Refresh Computer Deployment Scenario Using ZTI

1. Run the Configuration Manager advertisement for capturing the Refresh Computer deployment scenario that you created earlier in the deployment process.
2. Optionally, view the MDT deployment process using the Monitoring node in the Deployment Workbench or using the **Get-MDTMonitorData** cmdlet.

For more information about monitoring MDT deployments, see View MDT Deployment Progress.

The task sequence runs in the current operating system to capture user state migration data. The task sequence restarts the computer, starts Windows PE, and then initiates installation of the new operating system. Finally, the task sequence restarts the computer, starts the new operating system, restores the user state migration data, installs any packages, installs any applications, and performs any other actions configured in the task sequence. The target computer is now deployed.

Deploy Captured Images to Target Computers in the Replace Computer Deployment Scenario Using Configuration Manager

The Replace Computer deployment scenario requires two separate steps to complete the migration. First, run the advertisement for the task sequence you created to capture the user state migration data from the existing target computer (old computer). Second, run the Task Sequence Wizard to deploy the captured image of the reference computer to the new target computer (new computer) and restore the user state saved earlier in the process.

To deploy captured images of the reference computer to target computers

1. Save the user state migration data from the existing target computer as described in Save the User State Migration Data from the Existing Target Computer Using Configuration Manager.

2. Deploy the captured image of the reference computer to the new target computer as described in Deploy the Captured Image to the New Target Computer with the User State Migration Data from the Existing Computer Using Configuration Manager.

Save the User State Migration Data from the Existing Target Computer Using Configuration Manager

Start the ZTI deployment process by running the Configuration Manager advertisement for capturing the user state migration data that you created earlier in the process. This task sequence runs in the current operating system on the existing target computer.

To deploy the capture images to the target computers in the Replace Computer Deployment Scenario Using Configuration Manager

1. Run the Configuration Manager advertisement for capturing the Refresh Computer deployment scenario that you created earlier in the process.

2. Optionally, view the MDT deployment process using the Monitoring node in the Deployment Workbench or using the **Get-MDTMonitorData** cmdlet.

   For more information about monitoring MDT deployments, see View MDT Deployment Progress.

   The task sequence runs in the current operating system to capture user state migration data. At the end of the task sequence, the user state migration data of the existing target computer is saved to the Configuration Manager state migration point.

Deploy the Captured Image to the New Target Computer with the User State Migration Data from the Existing Computer Using Configuration Manager

Start the target computer with the ZTI bootable media created earlier in the process or from Windows Deployment Services. The ZTI bootable media starts Windows PE on the target computer and initiates the ZTI. At the end of the deployment process, the captured image of the reference computer is deployed on the target computer, and the user state migration data is restored from the Configuration Manager state migration point.

deployment scenario for deploying the captured image

1. Start the reference computer with the ZTI bootable media created earlier in the process or from Windows Deployment Services.

   Windows PE starts, and then the Windows Deployment Wizard starts.

2. Complete the Task Sequence Wizard, ensuring that you follow the configuration settings for the wizard pages listed in Table 120 and select values on the other wizard pages for your organization's requirements.

   > **Note**
   >
   > This wizard will not appear if you configure ZTI to perform a PXE boot and have configured a mandatory advertisement or if only one task sequence is

advertised to the target computer.

Table 120. Information for Completing the Task Sequence Wizard for the Replace Computer Deployment Scenario for Deploying the Captured Image Using Configuration Manager

Expand table

On this wizard page	Do this
Select a Task Sequence	Select the task sequence you created for the target computer deployment in the Replace Computer deployment scenario to deploy the captured image of the reference computer to the new target computer.

The wizard starts, and the operating system deployment starts.

3. Optionally, view the MDT deployment process using the Monitoring node in the Deployment Workbench or using the **Get-MDTMonitorData** cmdlet.

   For more information about monitoring MDT deployments, see View MDT Deployment Progress.

   The new target computer is deployed with the user state from the existing target computer automatically restored to the new target computer.

Managing ZTI Deployments in the Configuration Manager Console

You manage ZTI deployments using Configuration Manager through the Configuration Manager console. You use the Deployment Workbench in ZTI deployments only to configure the MDT DB. The wizards used to configure ZTI are integrated into the Configuration Manager console.

Manage ZTI deployments in the Configuration Manager console by:

- Creating a new task sequence for ZTI deployments using the Create MDT Task Sequence Wizard as described in Creating a ZTI Task Sequence Using MDT Task Sequence Templates in Configuration Manager

- Managing operating systems for ZTI deployments in the Configuration Manager console as described in Managing Operating Systems in Configuration Manager
- Managing device drivers for ZTI deployments in the Configuration Manager console as describe in Managing Device Drivers in Configuration Manager
- Deploying an operating system using task sequence bootable media as described in Deploying an Operating System Using Task Sequence Bootable Media in Configuration Manager
- Creating task sequence bootable media for ZTI as described in Creating Task Sequence Bootable Media in Configuration Manager
- Creating boot images for use with ZTI using the Create Image Using Microsoft Deployment Wizard as described in Creating ZTI Boot Images in Configuration Manager
- Managing software packages for ZTI in the Configuration Manager console as described in Managing Software Packages in Configuration Manager
- Deploying task sequences to reference or target computers for ZTI as described in Managing Task Sequence Deployment in Configuration Manager
- Manually adding computers to the site database for ZTI as described in Manually Adding Computers to the Site Database in Configuration Manager
- Managing computer collections for ZTI as described in Managing Computer Collections in Configuration Manager
- Managing distribution points for ZTI as described in Managing Distribution Points in Configuration Manager
- Configuring individual ZTI task sequence steps as described in Configuring ZTI Task Sequence Steps in Configuration Manager
- Configuring ZTI task sequence steps that perform server role-related actions as described in Configuring ZTI Server Role Task Sequence Steps in Configuration Manager

Creating a ZTI Task Sequence Using MDT Task Sequence Templates in Configuration Manager

Use the Create MDT Task Sequence Wizard in the Configuration Manager console to create task sequences in Configuration Manager that are integrated with MDT. MDT

includes task sequence templates that you can use to deploy the reference and target computers.

Create ZTI task sequences using the MDT task sequence templates by:

- Identifying the ZTI task sequence templates that are a part of MDT as described in Identify the Task Sequence Templates in MDT in Configuration Manager
- Identifying the packages and images that the MDT task sequence templates require as described in Identify the Packages and Images That the MDT Task Sequence Templates in Configuration Manager Require
- Creating ZTI task sequences as described in Create ZTI Task Sequences Using the Create MDT Task Sequence Wizard in Configuration Manager

Identify the Task Sequence Templates in MDT in Configuration Manager

Table 121 lists the task sequences templates included in MDT for Configuration Manager, the file name for each template, and a description of the template. The template files are located in the *install_folder*\SCCM folder (where *install_folder* is the folder in which MDT was installed).

Table 121. Task Sequence Templates Included in MDT for Configuration Manager

Expand table

Template	File name	Select this template to
Client Task Sequence	SCCM_Client.xml	Deploy client operating systems to target computers for all scenarios except the MDT Replace Computer deployment scenario.
Client Replace Task Sequence	SCCM_ClientReplace.xml	Captures user state migration data from target computers for the MDT Replace Computer deployment scenario.
Microsoft Deployment Custom Task Sequence	SCCM_Custom.xml	Create a task sequence that can be customizable to meet the needs of your organization.
Standard Server Task Sequence	SCCM_Server.xml	Deploy server operating systems to target computers for all scenarios.

Template	File name	Select this template to
User Driven Installation Task Sequence	SCCM_UDI.xml	Deploy operating systems to target computers using UDI.
User Driven Installation Replace Task Sequence	SCCM_UDIReplace.xml	Captures user state migration data from target computers for the MDT Replace Computer deployment scenario using UDI.

> **Note**
>
> Always use the Create MDT Task Sequence Wizard to import the task sequence templates. Although you can manually import the task sequence templates, doing so is not recommended.

Identify the Packages and Images That the MDT Task Sequence Templates in Configuration Manager Require

Table 122 lists the packages and images that the task sequence templates in MDT require. These packages and images must exist (or be created) for the task sequences to run correctly in Configuration Manager.

Table 122. Packages and Images Required by the Task Sequence Templates Included in MDT for Configuration Manager

Expand table

This package or image	Contains the
Boot image package	Boot image used to initiate the ZTI deployment process and in the middle of the process when performing the Refresh Computer deployment scenario.
Microsoft Deployment Files package	Script and tools necessary for the MDT task sequence templates for Configuration Manager.
OS image package	Image of the target operating system to be deployed to the target computer.

This package or image	Contains the
OS install package	All the files required to install the operating system (using Windows Setup.exe).
Client package	Configuration Manager client installation files.
USMT package	USMT files used to capture and restore user state.
Custom Settings package	Contains unattended files and customsettings.ini.

> **Note**
>
> You can use the generic boot images (WIM files) that the Deployment Workbench generates in ZTI deployments. However, you cannot use the LTI LiteTouch boot images (WIM files) that the Deployment Workbench generates in ZTI deployments using Configuration Manager.

The Create MDT Task Sequence Wizard can automatically create these packages and images or can use existing packages and images. The task sequence templates contain placeholders for each package and image listed in Table 122. The Create MDT Task Sequence Wizard substitutes the packages and images selected for the placeholders in the task sequence templates. After completing the wizard, the new created task sequence references the appropriate packages and images.

In addition to the packages and images that the task sequence templates require, consider creating and including the following elements in the task sequences to provide similar functionality in the Deployment Workbench:

- **Software distribution packages**. This package includes any software that will be installed as part of the operating system deployment (similar to the Applications node in the Deployment Workbench). These packages are created as packages and programs in Configuration Manager. For more information on how to create these packages, see the following sections in the Configuration Manager Documentation Library, which is included with Configuration Manager:
  - "Content Management in Configuration Manager"
  - "Application Management in Configuration Manager"
- **Windows package file (software update) packages**. These packages include any Windows package files that contain software updates (such as language packs,

security updates, and service packs) that will be installed as part of the operating system deployment (similar to the OS Packages node in the Deployment Workbench). You can use these software update packages:

  - Without modification by using the Software Updates feature in Configuration Manager. For more information on using these packages in the Software Updates feature, see the section, "Software Updates in Configuration Manager," in the Configuration Manager Documentation Library, which is included with Configuration Manager.

  - As installed directly by ZTI using the **Install Updates Offline** task sequence step type. For more information about configuring a task sequence step based on this type, see Configuring ZTI Task Sequence Steps in Configuration Manager.

  - Custom software distribution packages in Configuration Manager. For more information on how to create these packages, see the section, "Technical Reference for Content Management in Configuration Manager," in the Configuration Manager Documentation Library, which is included with Configuration Manager.

- **Device driver package**. Configuration Manager uses driver packages to control the distribution of drivers to distribution points. You can specify device driver categories in an **Auto Apply Drivers** task sequence step type to limit which drivers are installed, or you can install all device drivers using an **Apply driver package** task sequence step type. For more information about how to include device drivers in the operating system image, see the section, "How to Install Device Drivers to Computers by Using Task Sequences," in the Configuration Manager Documentation Library, which is included with Configuration Manager.

Create ZTI Task Sequences Using the Create MDT Task Sequence Wizard in Configuration Manager

The Create MDT Task Sequence Wizard in Configuration Manager substitutes the packages and images selected for the placeholders in the task sequence templates. After completing the wizard, the new task sequence references the appropriate packages and images.

> **Note**

Always use the Create MDT Task Sequence Wizard to create task sequences based on the MDT task sequence templates. Although you can manually import the task sequence templates, doing so is not recommend.

To create a ZTI task sequence using the Create MDT Task Sequence Wizard in Configuration Manager

1. Select **Start**, point to **All Programs**, and then point to **Microsoft System Center 2012**. Point to **Configuration Manager**, and then select **Configuration Manager Console**.

2. In the Configuration Manager console, in the navigation pane, select **Software Library**.

3. In the Software Library workspace, go to Overview/Operating Systems/Task Sequences.

4. On the Ribbon, on the **Home** tab, in the **Task Sequences** group, select **Create MDT Task Sequence**.

   The Create MDT Task Sequence Wizard starts.

5. Complete the Create MDT Task Sequence Wizard using the information in Table 123. Accept the default values unless otherwise specified.

Table 123. Information for Completing the Create MDT Task Sequence Wizard

Expand table

On this wizard page	Do this
Choose Template	- In **The following SCCM task sequence templates are available**, select ***task_sequence_template*** (where *task_sequence_template* is the task sequence template to be selected from Table 121). - Select **Next**.
Choose Template: General	- In **Task sequence name**, type ***task_sequence_name*** (where *task_sequence_name* is the name of the task sequence displayed in the Configuration Manager console). - In **Task sequence comments**, type ***comment*** (where *comment* is descriptive text that describes the purpose of the task sequence).

On this wizard page	Do this
	- Select **Next**.
Choose Template: Details	a. Select **Join a workgroup**. b. The other option is **Join a domain**. If you select this option, in **Workgroup**, type ***workgroup_name*** (where *workgroup_name* is the name of the workgroup to join). c. Select **Join a domain**. The other option is **Join a workgroup**. If you select this option, perform the following steps: - In **Domain**, type ***domain_name*** (where *domain_name* is the name of the workgroup to join). - Select **Set**. The **Windows User Account** dialog box appears. - In the **Windows User Account** dialog box, in **User name**, type ***domain_name\user_name*** (where *user_name* is the name of the user account used to join the computer to the domain and *domain_name* is the name of the domain in which the user account resides. This account must be a member of the Domain Admins group in the domain or have sufficient delegated permissions). - In **Password** and **Confirm password**, type ***password*** (where *password* is the password for the user account), and then select **OK**. d. In **User name**, type ***user_name*** (where *user_name* is the name of the registered user that appears in the Windows operating system properties). e. In **Organization name**, type ***organization_name*** (where *organization_name* is the name of the registered organization that appears in the Windows operating system properties). f. In **Product key**, type ***product_key*** (where *product_key* is the product key for operating system). g. Select **Next**.
Choose Template:	a. Select **This task sequence will never be used to capture and image**. The other option is **This task sequence may be used to capture and**

On this wizard page	Do this
Capture Settings	**image**. b. Select **This task sequence may be used to capture and image**. The other option is **This task sequence will never be used to capture and image**. If you select this option, perform the following steps: - In **Capture destination**, type ***unc_path*** (where *unc_path* is the fully qualified UNC path for the location of the WIM file, including the name of the WIM file). - Select **Set**. The **Windows User Account** dialog box appears. - In the **Windows User Account** dialog box, in **User name**, type ***domain_name\user_name*** (where *user_name* is the name of the user account that has Write permission to the folder specified in **Capture destination** and *domain_name* is the name of the domain in which the user account resides). - In **Password** and **Confirm password**, type ***password*** (where *password* is the password for the user account), and then select **OK**. c. Select **Next**.
Boot Image	a. Select **Specify an existing boot image**. The other option is **Create a new boot image package**. If you select this option, perform the following steps: - Select **Browse**. The **Select a Package** dialog box appears. - In **Select a Package** dialog box, select ***package_name*** (where *package_name* is the name of the existing boot image), and then select **OK**. b. Select **Create a new boot image package**.The other option is **Create a new boot image package**. If you select this option, in **Package source to be created**, type ***unc_path*** (where *unc_path* is the fully qualified UNC path for the location of the folder in which the package source will be stored). The Configuration Manager service account must have permission to modify the contents of this UNC path to update the boot images. Ensure

On this wizard page	Do this
	that the Configuration Manager service account has the appropriate permission on this network shared folder. You can also select **Browse** to locate the UNC path. c. Select **Next**.
Boot Image: General Settings	This wizard page appears only if you select **Create a new boot image package** on the **Boot Image** wizard page. a. In **Name**, type ***package_name*** (where *package_name* is the name to be given to the boot image package). b. In **Version**, type ***version_number*** (where *version_number* is the version number to be assigned to the boot image package). c. In **Comments**, type ***comment_text*** (where *comment_text* is descriptive information about the boot image package). d. Select **Next**.
Boot Image: Components	This wizard page appears only if you select **Create a new boot image package** on the **Boot Image** wizard page. a. In **Platform**, select ***platform*** (where *platform* is the platform architecture for the boot image—x86 or X64). b. Select or clear the ADO check box if you want to add ADO components to the boot image, which is needed to access SQL Server databases such as the MDT DB. By default, this check box is selected. If the check box is: - Selected, the ADO components are added to the boot image - Cleared, the ADO components are not added to the boot image Scripting, Hypertext Markup Language Application (HTA), XML, and WMI support are always added to the boot image. c. Select or clear the ***font*** check box (where font is the name of the font to be added, which can be Chinese [ZH-CN], Chinese [ZH-HK], Chinese [ZH-TW], Japanese [JA-JP], or Korean [KO-KR]) to add support for the optional fonts. d. Select **Next**.

On this wizard page	Do this
Boot Image: Customization	a. Select or clear the **Add prestart command files to enable the Deployment Wizard for this boot media** check box. If this check box is: - Selected, the prestart command files are added to the boot image. In **Command line**, type the prestart command script to run, which defaults to ZTIMediaHook.wsf. In **Folder for prestart command files**, type ***unc_path*** (where *unc_path* is the fully qualified UNC path to a writable folder). Alternatively, select **Browse** to find the folder in which the prestart command files reside. - Cleared, the prestart command files are not added to the boot image. b. Select or clear the **Add extra files to the new boot image** check box. If this check box is: - Selected, the extra files are added to the boot image. In **Path**, type ***path*** (where *path* is the fully qualified or relative local or UNC path to a writable folder). Alternatively, select **Browse** to find the folder in which the extra files reside. - Cleared, the extra files are not added to the boot image. c. In **Use a custom background bitmap file (UNC Path)**, type ***unc_path*** (where *unc_path* is the fully qualified UNC path to the bitmap file that you want to use as the background). Alternatively, select **Browse** to find the bitmap file. d. Select or clear the **Enable command support (F8)** check box. If this check box is: e. Select **Next**.
Boot Image: Customization	a. Select **Specify an existing Microsoft Deployment Toolkit files package**. The other option is **Create a new Microsoft Deployment Toolkit files package**. If you select this option, perform the following steps: - Select **Browse**. The **Select a Package** dialog box appears.

On this wizard page	Do this
	- In the **Select a Package** dialog box, select **package_name** (where *package_name* is the name of the existing package), and then select **OK**. b. Select **Create a new Microsoft Deployment Toolkit files package**. The other option is **Specify an existing Microsoft Deployment Toolkit files package**. If you select this option, in **Package source to be created**, type ***unc_path*** (where *unc_path* is the fully qualified UNC path for the location of the folder in which the package source will be stored). You can also select **Browse** to locate the UNC path. c. Select **Next**.
MDT Package: MDT Details	This wizard page appears only if you select **Create a new Microsoft Deployment Toolkit files package** on the **MDT Package** wizard page. a. In **Name**, type ***package_name*** (where *package_name* is the name to be given to the Microsoft Deployment Toolkit files package). b. In **Version**, type ***version_number*** (where *version_number* is the version number to be assigned to the Microsoft Deployment Toolkit files package). c. In **Language**, type ***language*** (where *language* is the language of the Microsoft Deployment Toolkit files package). d. In **Manufacturer**, type ***manufacturer*** (where *manufacturer* is the manufacturer of the Microsoft Deployment Toolkit files package). e. In **Comments**, type ***comment_text*** (where *comment_text* is descriptive information about the Microsoft Deployment Toolkit files package). f. Select **Next**.
OS Image	On this wizard page, you can select (or create) OS image packages or OS install packages. OS image packages are created from WIM files either from distribution media or from custom WIM files you have created. OS install packages contain all the necessary files to install the operating system, such as the contents of a Windows DVD. The ZTI task sequence templates always configure the **Apply Operating System Image** task sequence step to deploy the image index equal to **1**. If you want to deploy an image with a different index, change the configuration of the **Apply Operating System Image** task sequence step.

On this wizard page	Do this
	Also, if you select a SKU that is different from the default SKU, remove the following section from Unattend.xml in the Settings package: `<MetaData> <Key>/image/index</Key> <Value>1</Value> </MetaTag>` a. Select **Specify an existing OS image**. The other options are **Create a new OS image, Specify an existing OS installation package**, or **Create a new OS install package**. If you select this option, perform the following steps: - Select **Browse**. The **Select a Package** dialog box appears. - In **Select a Package** dialog box, select ***package_name*** (where *package_name* is the name of the existing package), and then select **OK**. b. Select **Create a new OS image**. Select **Create a new OS image**. The other options are **Specify an existing OS image**,**Specify an existing OS installation package**, or **Create a new OS install package**. If you select this option, perform the following steps: - In **OS image file (WIM) location**, type ***unc_path*** (where *unc_path* is the fully qualified UNC path for the location of the folder in which the WIM file is located, including the name of the WIM file). You can also select **Browse** to locate the WIM file. - In **Package source folder to be created**, type ***unc_path*** (where *unc_path* is the fully qualified UNC path for the location of the folder in which the package source will be stored). In **Package source folder to be created**, type ***unc_path*** (where *unc_path* is the fully qualified UNC path for the location of the folder in which the package source will be stored). You can also select **Browse** to locate the UNC path. c. Select **Specify an existing OS installation package**. The other options are **Specify an existing OS image**, **Create a new OS image**, or **Create a new OS install package**. If you select this option, perform the following steps: - Select **Browse**.

On this wizard page	Do this
	The **Select a Package** dialog box appears. - In the **Select a Package** dialog box, select ***package_name*** (where *package_name* is the name of the existing package), and then select **OK**. d. Select **Create a new OS install package**. e. The other options are **Specify an existing OS image**, **Create a new OS image**, or **Specify an existing OS installation package**. The other options are **Specify an existing OS image**, **Create a new OS image**, or **Specify an existing OS installation package**. The OS installation folder contents are copied from the specified UNC location to the package source folder, which are then used to create the package. If you select this option, perform the following steps: - In **OS installation folder location**, type ***unc_path*** (where *unc_path* is the fully qualified UNC path for the location of the folder in which the package source will be stored). You can also select **Browse** to locate the UNC path. - In **Package source folder to be created**, type ***unc_path*** (where *unc_path* is the fully qualified UNC path for the location of the folder in which the package source will be stored). You can also select **Browse** to locate the UNC path. f. Select **Next**.
OS Image: Image Details	This wizard page appears only if you select **Create a new OS image** on the **OS Image** wizard page. a. In **Name**, type ***package_name*** (where *package_name* is the name to be given to the package). b. In **Version**, type ***version_number*** (where *version_number* is the version number to be assigned to the package). c. In **Comments**, type ***comment_text*** (where *comment_text* is descriptive information about the package). d. Select **Next**.

On this wizard page	Do this
OS Image: Install Source	This wizard page appears only if you select **Create a new OS install package** on the **OS Image** wizard page. a. In **Name**, type ***package_name*** (where *package_name* is the name to be given to the package). b. In **Version**, type ***version_number*** (where *version_number* is the version number to be assigned to the package). c. In **Comments**, type ***comment_text*** (where *comment_text* is descriptive information about the package). d. Select **Next**.
OS Image: OS Image Index	a. In **The selected operating system image (WIM) file contains multiple images. Specify which image you would like to deploy**, select ***image_index*** (where *image_index* is the image index of the image that contains the operating system you want to deploy). b. Select **Next**.
Client Package	a. Select **Specify an existing ConfigMgr client package**. The other option is **Create a new ConfigMgr client package**. If you select this option, perform the following steps: - Select **Browse**. The **Select a Package** dialog box appears. - In the **Select a Package** dialog box, select ***package_name*** (where *package_name* is the name of the existing package), and then select **OK**. b. Select **Create a new ConfigMgr client package**. Select **Create a new ConfigMgr client package**. The other option is **Specify an existing ConfigMgr client package**. c. Select **Next**.
USMT Package	You cannot specify the location for storing the user state migration data in the CustomSettings.ini file for ZTI using Configuration Manager. The **Request state store** task automatically determines the state migration point to use for storing user state migration data. For more information, see the section, "How to Capture and Restore User State," in the section, "How to Manage the User State in Configuration Manager," in the Configuration Manager Documentation Library, which is installed with

On this wizard page	Do this
	Configuration Manager. a. Select **Specify an existing USMT package**. The other option is **Create a new USMT package**. If you select this option, perform the following steps: The other option is **Create a new USMT package**. If you select this option, perform the following steps: - Select **Browse**. The **Select a Package** dialog box appears. - In the **Select a Package** dialog box, select ***package_name*** (where *package_name* is the name of the existing package), and then select **OK**. b. Select **Create a new USMT package**. The other option is **Specify an existing USMT package**. If you select this option, perform the following steps: - In **Path to USMT executables and related**, type ***path*** (where *path* is the fully qualified path for the location of the folder in which the USMT source files are located). You can also select **Browse** to locate the path. - In **Package source to be created**, type ***unc_path*** (where *unc_path* is the fully qualified UNC path for the location of the folder in which the package source will be stored). You can also select **Browse** to locate the UNC path. c. Select **Next**. USMT can only back up and restore locally cached user profiles, not network copies. For more information on using roaming profiles, see the Folder Redirection, Offline Files, and Roaming User Profiles overview.
USMT Package: USMT Details	This wizard page appears only if you select **Create a new USMT package** on the **USMT Package** wizard page. a. In **Name**, type ***package_name*** (where *package_name* is the name to be given to the package). b. In **Version**, type ***version_number*** (where *version_number* is the version number to be assigned to the package).

On this wizard page	Do this
	c. In **Language**, type ***language*** (where *language* is the language of the package). d. In **Manufacturer**, type ***manufacturer_name*** (where *manufacturer_name* is the name of the software manufacturer in the package). e. In **Comments**, type ***comment_text*** (where *comment_text* is descriptive information about the package). f. Select **Next**.
Settings Package	a. Select **Specify an existing settings package**. The other option is **Create a new settings package**. If you select this option, perform the following steps: - Select **Browse**. The **Select a Package** dialog box appears. - In the **Select a Package** dialog box, select ***package_name*** (where *package_name* is the name of the existing package), and then select **OK**. b. Select **Create a new settings package**. The other option is **Specify an existing settings package**. If you select this option, in **Package source to be created**, type ***unc_path*** (where *unc_path* is the fully qualified UNC path for the location of the folder in which the package source will be stored). You can also select **Browse** to locate the UNC path. c. Select **Next**.
Settings Package: Settings Details	This wizard page appears only if you select **Create a new settings package** on the **Settings Package** wizard page. This wizard page appears only if you select **Create a new settings package** on the **Settings Package** wizard page. a. In **Name**, type ***package_name*** (where *package_name* is the name to be given to the package). b. In **Version**, type ***version_number*** (where *version_number* is the version number to be assigned to the package). c. In **Language**, type ***language*** (where *language* is the language of the

On this wizard page	Do this
	package). d. In **Manufacturer**, type ***manufacturer_name*** (where *manufacturer_name* is the name of the manufacturer for the software in the package). e. In **Comments**, type ***comment_text*** (where *comment_text* is descriptive information about the package). f. Select **Next**.
Sysprep Package	a. Select **No Sysprep package is required**.
Summary	Select Next.

The **Summary** wizard page displays a status bar that shows the progress of the tasks defined in the wizard. The Create MDT Task Sequence Wizard closes when the task sequence is created.

Managing Operating Systems in Configuration Manager

Manage operating systems in the Operating Systems node in the Software Library workspace. The operating systems are contained and managed in the following nodes beneath the Operating Systems node:

- **Operating System Installers**. This node contains operating systems that are used to deploy reference computers and are based on the install.wim file from the original operating system media.
- **Operating System Images**. This node contains captured operating system images from reference computers and that are deployed to your target computers.

  For more information about managing operating systems in the Configuration Manager console, see the section, "Configuring Configuration Manager for Operating System Deployments," in the Configuration Manager Documentation Library, which is installed with Configuration Manager.

Managing Device Drivers in Configuration Manager

Manage device drivers in the Configuration Manager console in Configuration Manager by:

- Importing the device drivers into Configuration Manager as described in Import Drivers into Configuration Manager
- Creating a new driver package that contains the device drivers as described in Create a New Configuration Manager Driver Package
- Adding device drivers and device driver packages to operating systems and boot images as described in Add Device Drivers to Operating System and Boot Images in Configuration Manager
- Deploying specific device drivers to target computers for ZTI deployments as described in Deploy Specific Device Drivers to Target Computers in Configuration Manager

Import Drivers into Configuration Manager

To import drivers into Configuration Manager, use the Import New Driver Wizard. For information about this wizard, see the section, "How to Import Windows Device Drivers into the Driver Catalog," in the Configuration Manager Documentation Library, which is installed with Configuration Manager.

Create a New Configuration Manager Driver Package

A driver package contains the content associated with one or more device drivers. You must add device drivers to a driver package and copy them to a distribution point before Configuration Manager clients can install them. For information about creating a new driver package, see the section, "How to Create a New Driver Package," in the Configuration Manger Documentation Library, which is installed with Configuration Manager.

Add Device Drivers to Operating System and Boot Images in Configuration Manager

When you have added device drivers to the driver catalog, you can add them to existing operating systems and boot images. The driver catalog helps manage the cost and complexity of deploying an operating system in an environment that contains different types of computers and devices. Storing device drivers in the driver catalog and not with each individual operating system image greatly reduces the number of operating system images you need.

For information about managing the driver catalog, see the section, "How to Manage the Driver Catalog in Configuration Manager," in the Configuration Manager

Documentation Library, which is installed with Configuration Manager.

To add device drivers to operating systems and boot images in Configuration Manager

- Add device drivers from the driver catalog to existing operating systems as described in Add Device Drivers to an Operating System in Configuration Manager.
- Add device drivers from the driver catalog to existing boot images as described in Add Device Drivers to a Boot Image in Configuration Manager.

Add Device Drivers to an Operating System in Configuration Manager

Add new device drivers to an existing operating system image using the Task Sequence Editor. To allow Configuration Manager to search in the driver catalog for the new device drivers, add an **Auto Apply Drivers** task sequence step to an existing task sequence.

For information about adding device drivers to an operating system, see the section, "How to Install Device Drivers to Computers by Using Task Sequences," in the Configuration Manager Documentation Library, which is installed with Configuration Manager.

Add Device Drivers to a Boot Image in Configuration Manager

You can add Windows device drivers that you have imported into the driver catalog to one or more boot images. Only mass storage device drivers and network adapter device drivers should be added to boot images, because other types of drivers are not needed and will increase the size of the boot image. Only add valid device drivers that are intended for use with Windows 8.1, because the version of Windows PE is based on Windows 8.1.

For information about adding device drivers to boot images, see the section, "How to Add and Remove Device Drivers That Are Associated with Driver Packages and Boot Images," in the Configuration Manager Documentation Library, which is installed with Configuration Manager.

Deploy Specific Device Drivers to Target Computers in Configuration Manager

By default, ZTI using Configuration Manager deploys all device drivers to the target computers. Then, the target operating system uses Plug-and-Play IDs to identify the device drivers needed for the devices on the target computers.

To change this default behavior, configure the ZTI deployment process to install specific drivers to target computers as described in Control Device Driver Deployments Using Configuration Manager for ZTI. For more information about strategies for device driver management, see Select the Device Driver Management Strategy.

Deploying an Operating System Using Task Sequence Bootable Media in Configuration Manager

To initiate ZTI deployment using Configuration Manager from bootable media, start the target computer with the bootable media. The boot process starts Windows PE, and then starts ZTI. You can start the target computer from a UFD, CD, or DVD.

> **Note**
>
> The ZTI deployment process using Configuration Manager can also be initiated by starting the target computer from Windows Deployment Services. However, for reference computers it may be easier to start the ZTI deployment process from bootable media.

For more information about how to deploy an operating system using task sequence bootable media, see the section "How to Deploy Operating Systems by Using Media in Configuration Manager" in the Configuration Manager Documentation Library, which is installed with Configuration Manager.

Creating Task Sequence Bootable Media in Configuration Manager

To initiate the ZTI deployment process using Configuration Manager from bootable media, provide a method for starting the computer with Windows PE and the necessary software by creating the task sequence bootable media disk. Use the Task Sequence Media Wizard in Configuration Manager console to create bootable media for storage on a UFD, CD, or DVD.

> **Note**

The ZTI deployment process using Configuration Manager can also be initiated by starting the target computer from Windows Deployment Services. However, for reference computers it may be easier to start the ZTI deployment process from bootable media.

For more information about how to create task sequence bootable media, see the section, "How to Create Bootable Media ," in the Configuration Manager Documentation Library, which is installed with Configuration Manager.

Creating ZTI Boot Images in Configuration Manager

Some situations call for you to create a new boot image for the ZTI process without running the Create MDT Task Sequence Wizard. You can create new boot images for ZTI using the Create Boot Image using MDT Wizard in the Boot Images node in the Configuration Manager console.

To create a ZTI boot image in Configuration Manager

1. Select **Start**, point to **All Programs**, and then point to **Microsoft System Center 2012**. Point to **Configuration Manager**, and then select **Configuration Manager Console**.
2. In the Configuration Manager console, in the navigation pane, select **Software Library**.
3. In the Software Library workspace, go to Overview/Operating Systems/Boot Images.
4. On the Ribbon, on the **Home** tab, in the **Task Sequences** group, select **Create Boot Image using MDT**.

   The Create Boot Image Using MDT Wizard starts.
5. Complete the Create Boot Image Using MDT Wizard using the information in Table 124. Accept the default values unless otherwise specified.

Table 124. Information for Completing the Create Boot Image using MDT Wizard

Expand table

On this wizard page	Do this
Package Source	a. In **Package source folder to be created (UNC) path**, type ***unc_path*** (where *unc_path* is the fully qualified UNC path to the folder you want to use as the source for the boot image package). The Configuration Manager service account must have permission to modify the contents of this UNC path to update the boot images. Ensure that the Configuration Manager service account has the appropriate permission on this network shared folder. Alternatively, select **Browse** to find the network shared folder. b. Select **Next**.
General Settings	a. In **Name**, type ***package_name*** (where *package_name* is descriptive name displayed in the Configuration Manager Console). b. In **Version**, type ***package_version*** (where *package_version* is version number that you want to assign to the package). c. In **Comments**, type ***package_comments*** (where *package_comments* is text that describes the purpose of the boot image). d. Select **Next**.
General Settings: Options	a. In **Platform**, select ***platform*** (where *platform* is the platform architecture for the boot image—x86 or x64). b. In **Scratch Space**, select ***scratch_space*** (where *scratch_space* is the amount of writeable space available on the Windows PE system volume when booted in ramdisk mode and is specified in MB). c. Select **Next**.
General Settings: Components	a. In **Platform**, select ***platform*** (where *platform* is the platform architecture for the boot image—x86 or x64). b. Select or clear the ***component*** check box (where component is the name of the component to be selected). If the check box is: - Selected, the component is added to the boot image - Cleared, the component is not added to the boot image Scripting, HTA, XML, and WMI support are always added to the boot image.

On this wizard page	Do this
	c. Select **Next**.
Customization	a. Select or clear the **Add prestart command files to enable the Deployment Wizard for this boot media** check box. If this check box is: - Selected, the prestart command files are added to the boot image. In **Command line**, type the prestart command script to run, which defaults to ZTIMediaHook.wsf. In **Folder for prestart command files**, type ***unc_path*** (where *unc_path* is the fully qualified UNC path to a writable folder). Alternatively, select **Browse** to find the folder in which the prestart command files reside. - Cleared, the prestart command files are not added to the boot image. b. Select or clear the **Add extra files to the new boot image** check box. If this check box is: - Selected, the extra files are added to the boot image. In **Path**, type ***path*** (where *path* is the fully qualified or relative local or UNC path to a writable folder). Alternatively, select **Browse** to find the folder in which the extra files reside. - Cleared, the extra files are not added to the boot image. c. In **Use a custom background bitmap file (UNC Path)**, type ***unc_path*** (where *unc_path* is the fully qualified UNC path to the bitmap file that you want to use as the background). Alternatively, select **Browse** to find the bitmap file. d. Select or clear the **Enable command support (F8)** check box. e. Select **Next**.
Summary	Review the information in **Details**, and then select **Next**.
Confirmation	You can select **Save Output** to save the output of the wizard to a file. You can also select **View Script** to view the Windows PowerShell scripts used to perform the wizard tasks. Select **Close**.

After the Create Boot Image using MDT Wizard finishes, the new boot image appears in the preview pane in the Configuration Manager console.

Managing Software Packages in Configuration Manager

Manage software packages in the Configuration Manager console in Configuration Manager by:

- Adding language packs as described in Add Language Packs in Configuration Manager
- Adding software updates as described in Add Software Updates in Configuration Manager

Add Language Packs in Configuration Manager

Language packs are .cab files that you can add to Configuration Manager packages either offline or online. Before adding language packs, however, create a Configuration Manager package that contains one or more language packs.

The number of language packs that you add to a Configuration Manager package is based on the type of deployment being performed. When deploying language packs using:

- ZTI, put one or more language packs in each Configuration Manager package. This allows you to bundle the necessary language packs for your organization and include them in one Configuration Manager package.

  If you bundle two or more language packs in a Configuration Manager package and deploy that package, all language packs will be deployed. If you want to deploy different combination of language packs, then consider bundling one language pack in a Configuration Manager package and create a separate, conditional task sequence step to deploy each different language pack.

- UDI, put only one language pack in each Configuration Manager package. This one-to-one relationship allows the user to select individual language packs as desired in the UDI Wizard.

To create a Configuration Manager package that contains one or more language packs

1. Create a folder that will contain the package source for the language pack (.cab file).

> ⓘ **Note**
>
> Create this folder where previously created packages are stored or where space is available.

2. Create a subfolder beneath the folder you created in step 1 for the language pack (.cab file).

3. Add the language pack (.cab file) downloaded from Microsoft into the subfolder created in step 2.

   > ⓘ **Note**
   >
   > Language packs already in .cab format are available for download from the Microsoft Volume Licensing website.

4. Repeat steps 2 and 3 for each language pack that you want to be a part of the package.

5. Run the **New Package Wizard** in the Configuration Manager console, ensuring the following options are selected during the wizard:

   - On the **Data Source** wizard page, select **This package contains source files**, and then select **Set** to set the source directory to the folder you created in step 1.

   - On the **Data Source** wizard page, select the **Enable binary differential replication** check box.

     For more information about how to create a Configuration Manager package, see the section, "How to Create a Package and Program by using the Create Package and Program Wizard ," in the Configuration Manager Documentation Library, which is installed with Configuration Manager.

To add language packs offline to Windows

> ⓘ **Note**
>
> When used with MDT, the term *offline* means that the computer is booted into Windows PE, and thus the image can be modified offline—not in the currently booted operating system.

1. Select **Start**, point to **All Programs**, and then point to **Microsoft System Center 2012**. Point to **Configuration Manager**, and then select **Configuration Manager Console**.

2. In the Configuration Manager console, in the navigation pane, select **Software Library**.

3. In the Software Library workspace, go to Overview/Operating Systems/Task Sequences.

4. In the preview pane, select ***task_sequence*** (where *task_sequence* is the name of the task sequence for which you want to add the language pack).

5. On the Ribbon, on the **Home** tab, in the **Task Sequences** group, select **Edit**.

   The ***task_sequence_name*** **Task Sequence Editor** dialog box opens (where *task_sequence_name* is the name of the task sequence to which you want to add language packs offline).

6. In the ***task_sequence_name*** **ask Sequence Editor** dialog box, under the **Postinstall** phase, go to the **Setup Windows and ConfigMgr** task.

7. Select the task immediately above this task so that the new added task will be positioned immediately before the **Setup Windows and ConfigMgr** task.

8. On the menu bar, select **Add**, select **MDT**, and then select **Install Language Packs Offline**.

   The **Install Language Packs Offline** task sequence step is added to the task sequence.

9. On the **Properties** tab of the newly added task sequence step, type the relevant information in the **Name** and **Description** boxes.

10. On the **Properties** tab of the newly added task sequence step, select **Browse**.

    The **Select a Package** dialog box appears.

11. In the **Select a Package** dialog box, select ***language_package*** (where *language_package* is the name of the package that contains the language pack you want to install), and then select **OK**.

12. In the ***task_sequence_name*** **Task Sequence Editor** dialog box, select **OK**.

    To add language packs online to Windows

> **Note**
>
> When used in MDT, the term *online* means that the computer is booted into an operating system but run as an Administrator user so that final configurations can be made to the running operating system.

1. Select **Start**, point to **All Programs**, and then point to **Microsoft System Center 2012**. Point to **Configuration Manager**, and then select **Configuration Manager Console**.
2. In the Configuration Manager console, in the navigation pane, select **Software Library**.
3. In the Software Library workspace, go to Overview/Operating Systems/Task Sequences.
4. In the preview pane, select ***task_sequence*** (where *task_sequence* is the name of the task sequence for which you want to add the language pack).
5. On the Ribbon, on the **Home** tab, in the **Task Sequences** group, select **Edit**.

   The ***task_sequence_name*** **Task Sequence Editor** dialog box opens (where *task_sequence_name* is the name of the task sequence to which you want to add language packs offline).
6. In the ***task_sequence_name*** **Task Sequence Editor** dialog box, under the **State Restore** group, select the **Gather**step (so that the newly added task will be positioned immediately after the **Gather**task sequence step in the **State Restore** group).
7. On the menu bar, select **Add**, select **MDT**, and then select **Install Language Packs Online**.

   The **Install Language Packs Online** task sequence step is added to the task sequence.
8. On the **Properties** tab of the newly added task sequence step, type the relevant information in the **Name** and**Description** boxes.
9. On the **Properties** tab of the newly added task sequence step, select **Browse**.

   The **Select a Package** dialog box appears.

10. In the **Select a Package** dialog box, select ***language_package*** (where *language_package* is the name of the package that contains the language pack you want to install), and then select **OK**.

11. In the ***task_sequence_name*** **Task Sequence Editor** dialog box, select **OK**.

Add Software Updates in Configuration Manager

Use Configuration Manager to add updates—online or offline—during the task sequence. Manage software updates in Configuration Manager using a server configured as a software update point. For detailed information on software updates using Configuration Manager, see the section "Configuring Software Updates in Configuration Manager," in the Configuration Manager Documentation Library, which is installed with Configuration Manager.

Use deployment packages to deploy software updates. For more information about configuration and deployment of software update packages, see the section "Download Software Updates," in the Configuration Manager Documentation Library, which is installed with Configuration Manager.

To install operating system updates online, add the updates to a **Deployment Management** item. Create a **Deployment Management** item using the Deploy Package Wizard. For more information on deploying packages and deployment management, see the following sections in the Configuration Manager Documentation Library, which is installed with Configuration Manager:

- "Manage Software Update Settings".
- "Deploy Software Updates".

To add offline updates to Windows

1. Select **Start**, point to **All Programs**, and then point to **Microsoft System Center 2012**. Point to **Configuration Manager**, and then select **Configuration Manager Console**.

2. In the Configuration Manager console, in the navigation pane, select **Software Library**.

3. In the Software Library workspace, go to Overview/Operating Systems/Task Sequences

4. In the preview pane, select ***task_sequence*** (where *task_sequence* is the name of the task sequence for which you want to add the language pack).

5. On the Ribbon, on the **Home** tab, in the **Task Sequences** group, select **Edit**.

   The ***task_sequence_name*** **Task Sequence Editor** dialog box opens (where *task_sequence_name* is the name of the task sequence to which you want to add language packs offline).

6. In the ***task_sequence_name*** **Task Sequence Editor** dialog box, under the **Postinstall** phase, go to the **Setup Windows and ConfigMgr** task.

7. Select the task immediately above this task so that the new added task will be positioned just above the **Setup Windows and ConfigMgr** task.

   > ⓘ **Note**
   >
   > Only updates in CAB files are supported using the **Install Updates Offline** task sequence type. Other formats of update files are ignore, such as MSI, MSP, or executable (.exe) files.

8. On the menu bar, select **Add**, select **MDT**, and then select **Install Updates Offline**.

   The **Install Updates Offline** task sequence step is added to the task sequence.

9. On the **Properties** tab of the newly added task sequence step, type the relevant information in the **Name** and **Description** boxes.

10. On the **Properties** tab of the newly added task sequence step, select **Browse**. The **Select a Package** dialog box appears.

11. In the **Select a Package** dialog box, select ***update_package*** (where *update_package* is the name of the package that contains the updates you want to install), and then select **OK**.

12. In the ***task_sequence_name*** **Task Sequence Editor** dialog box, select **OK**.

    To add online updates to Windows

> ⓘ **Note**
>
> The task sequence templates in MDT include the **Install Software Updates** task sequence step to perform online updates. This step is only necessary when creating custom task sequences.

1. Select **Start**, point to **All Programs**, and then point to **Microsoft System Center 2012**. Point to **Configuration Manager**, and then select **Configuration Manager Console**.

2. In the Configuration Manager console, in the navigation pane, select **Software Library**.

3. In the Software Library workspace, go to Overview/Operating Systems/Task Sequences.

4. In the preview pane, select ***task_sequence*** (where *task_sequence* is the name of the task sequence for which you want to add the language pack).

5. On the Ribbon, on the **Home** tab, in the **Task Sequences** group, select **Edit**.

   The ***task_sequence_name*** **Task Sequence Editor** dialog box opens (where *task_sequence_name* is the name of the task sequence to which you want to add language packs offline).

6. In the ***task_sequence_name*** **Task Sequence Editor** dialog box, under the **State Restore** phase, go to the **Restart Computer** task.

7. Select the task immediately above this task so that the new added task will be positioned just above the **Restart Computer** task.

8. In the **Task Sequence Editor** dialog box, select **Add**, select **General**, and then select **Install Software Updates**.

9. On the menu bar, select **Add**, select **General**, and then select **Install Software Updates**.

   The **Install Software Updates** task sequence step is added to the task sequence.

10. On the **Properties** tab of the newly added task sequence step, type the relevant information in the **Name** and **Description** boxes.

11. On the **Properties** tab of the newly added task sequence step, select one of the following options:

    - **Mandatory Software Updates**. This option installs only the software updates that are categorized as being mandatory. This option is selected by default.

    - **All Software Updates**. This option installs all software updates, including mandatory and optional software updates.

12. In the ***task_sequence_name*** **Task Sequence Editor** dialog box, select **OK**.

> **Note**
>
> The Software Update Point role and Windows Server Update Services (WSUS) must be properly configured to work with this task sequence step type.

Managing Task Sequence Deployment in Configuration Manager

In ZTI deployments using Configuration Manager, you must deploy the task sequences to the target computers using the Deploy Software Wizard. The task sequence is deployed to a collection that includes the reference computer or target computers. For more information about deploying task sequences, see the section "How to Deploy a Task Sequence," in the section, "How to Manage Task Sequences in Configuration Manager," in the Configuration Manager Documentation Library, which is installed with Configuration Manager.

Manually Adding Computers to the Site Database in Configuration Manager

In ZTI deployments using Configuration Manager, computers must exist in the Configuration Manager site database before you can advertise a task sequence to the computer. Configuration Manager includes a feature for automatically adding target computers to the site database. However, for reference computers, it is easier to manually add the reference computer to the site database.

For more information about manually adding computers to the site database, see the section, "How to Add a Computer to the Configuration Manager Database," in the section, "How to Deploy Operating Systems in Configuration Manager," in the Configuration Manager Documentation Library, which is installed with Configuration Manager.

Managing Computer Collections in Configuration Manager

In ZTI deployments using Configuration Manager, the task sequences must be advertised to a collection of target computers. In Configuration Manager, *collections* are a grouping of one or more computers. For more information about managing computer collections, see the following sections in the Configuration Manager Documentation Library, which is installed with Configuration Manager:

- "Introduction to Collections in Configuration Manager"
- "Planning for Collections in Configuration Manager"
- "Operations and Maintenance for Collections in Configuration Manager"
- "Security and Privacy for Collections in Configuration Manager"

Managing Distribution Points in Configuration Manager

In ZTI deployments using Configuration Manager, *distribution points* are the repository for the files being deployed to the reference and target computers. Your organization may have more than one distribution point. Configure distribution points for the operating system images and software packages that MDT uses, ensuring that each reference and target computer has a persistent, high-speed connection to a distribution point.

If you make any changes to the operating system images and software packages that MDT uses, update the distribution points where these images and packages are stored.

For more information about managing distribution points, see the section, "Operations and Maintenance for Content Management in Configuration Manager," in the Configuration Manager Documentation Library, which is installed with Configuration Manager.

Configuring ZTI Task Sequence Steps in Configuration Manager

After you create a ZTI task sequence using the Create MDT Task Sequence Wizard in Configuration Manager, you can customize it using the Configuration Manager console. The Configuration Manager console allows you to:

- Add new task sequence steps
- Modify existing task sequence steps
- Delete existing task sequence steps

  You perform these tasks using the standard methods available in the Configuration Manager console. For more information about:

- Configuring ZTI task sequence steps using the Configuration Manager console in Configuration Manager, see the section, "How to Edit an Existing Task Sequence," in the section, "How to Manage Task Sequences in Configuration Manager," in the

Configuration Manager Documentation Library, which is installed with Configuration Manager.

- Task sequences, see the section, "Planning a Task Sequences Strategy in Configuration Manager," in the Configuration Manager Documentation Library, which is installed with Configuration Manager.

Configuring ZTI Server Role Task Sequence Steps in Configuration Manager

ZTI can help automate the deployment of server roles in Windows Server. Configure ZTI task sequence steps in Configuration Manager to deploy the supported server roles, which include:

- AD DS
- DNS Server
- DHCP Server

  The process for configuring the server role task sequence steps is similar for ZTI and LTI. For more information about configuring server role task sequence steps for ZTI in Configuration Manager, see Configuring Server Role Task Sequence Steps.

Performing UDI Deployments

You perform UDI deployments in MDT using Configuration Manager within an AD DS domain, within a Windows workgroup, or from removable media.

> ⓘ **Note**
>
> If you are unfamiliar with UDI, review the UDI terms and terminology in the section, "UDI Concepts", in the MDT document *Microsoft Deployment Toolkit Reference*. Familiarizing yourself with these terms and terminology will help you be more successful in applying the remainder of this guide to your organization.

Perform UDI deployments by:

- Reviewing the overview information for UDI deployments as described in Overview of UDI Deployments

- Preparing the UDI deployment environment as described in Preparing the UDI Deployment Environment
- Preparing for UDI deployment to the reference computer as described in Preparing for UDI Deployment to the Reference Computer
- Deploying to and capturing an image of the reference computer in UDI as described in Deploying To and Capturing an Image of the Reference Computer Using UDI
- Preparing for UDI deployment to the target computers as described in Preparing for UDI Deployment to Target Computers
- Deploying captured images to the target computer using UDI as described in Deploying Captured Images to Target Computers Using UDI

Overview of UDI Deployments

UDI allows the interactive deployment of Windows operating systems and applications using Configuration Manager. Typically, when deploying operating systems using the OSD feature in Configuration Manager and ZTI in MDT, you must provide all the information necessary to deploy the operating system. Prior to performing the deployment, the information is configured in configuration files or in databases (such as the CustomSettings.ini file or the MDT DB). During the ZTI deployment process, ZTI converts the appropriate configuration settings to task sequence variables, which the MDT task sequences consume for UDI. All of the configuration settings must be provided before you can initiate the deployment.

UDI provides a wizard-driven interface that runs on the target computer, which allows you to provide configuration information immediately prior to operating system and application deployment. This allows you to create generic OSD task sequences, and then have other users provide computer specific information at the time of deployment, which provides greater flexibility in the deployment process.

> **Note**
>
> If you are unfamiliar with UDI, review the UDI terms and terminology in the section, "UDI Concepts", in the MDT document *Microsoft Deployment Toolkit Reference*. Familiarizing yourself with these terms and terminology will help you be more successful in applying the remainder of this guide to your organization.

Review the overview information about UDI deployments in the subsequent sections:

- Overview of UDI in MDT Deployment Scenarios
- Overview of Built-in UDI Components

Table 135 lists additional content resources for UDI administration and development.

Table 135. Additional Content Resources for UDI Administration and Development

Expand table

Resource	Description
Overview of UDI Administration	This content provides an overview of UDI administration.
Configuring UDI Wizard Behavior	This content describes how to configure the behavior of the UDI Wizard using the UDI Wizard Designer.
Review the UDI Wizard Designer User Interface	This content provides an overview of the UI for the UDI Wizard designer.
"UDI Reference" section in *Microsoft Deployment Toolkit Reference*	This content provides reference material for UDI, including information on: - OSDResults - AppInstaller - Built-in UDI stage groups and stages - UDI tasks - UDI validators - UDI wizard pages
User-Driven Installation Developers Guide	This content provides guidance on how to customized and extend UDI to meet the needs of your organization.
"UDI Concepts" section in *Microsoft Deployment Toolkit Reference*	This content provides definitions of UDI terms and terminology and conceptual information about UDI.

Overview of UDI in MDT Deployment Scenarios

UDI supports the New Computer, Refresh Computer, and Replace Computer MDT deployment scenarios, which were described in Identifying Deployment Scenarios. UDI

supports these deployments scenarios using the Configuration Manager task sequence templates provided with MDT. Table 136 lists the MDT deployment scenarios and the corresponding UDI task sequence templates used to perform the deployment scenario.

Table 136. MDT Deployment Scenarios and UDI Task Sequence Templates Used to Perform the Scenarios

Expand table

Deployment scenario	UDI tasks sequences used
New Computer	Run task sequence created using **User Driven Installation Task Sequence** task sequence template.
Refresh Computer	Run task sequence created using **User Driven Installation Task Sequence** task sequence template.
Replace Computer	- Run task sequence created using **User Driven Installation Replace Task Sequence** task sequence template on existing computer. - Run the task sequence created using the **User Driven Installation Task Sequence** task sequence template on the new computer.

The UDI Wizard is run by the **UDI Wizard** task sequence step in these task sequences at the appropriate place in the task sequence. To identify how UDI performs each of the MDT deployment scenarios, see the subsequent sections:

- UDI in the New Computer Deployment Scenario
- UDI in the Refresh Computer Deployment Scenario
- UDI in the Replace Computer Deployment Scenario

  For more details about how UDI operates in each of these MDT deployment scenarios, see the corresponding sections in "UDI Stage Reference" in *Microsoft Deployment Toolkit Reference*.

UDI in the New Computer Deployment Scenario

For the New Computer Deployment Wizard, the operating system images can be in the following locations:

- **On a distribution point**. This method uses the traditional OSD deployment methodology in Configuration Manager.

- **On a local disk on the target computer**. This method leverages the prestaged media feature in Configuration Manager.

  Task sequences created using the **User Driven Installation Task Sequence** task sequence template automatically detect which method to use and perform the deployment accordingly. The New Computer deployment scenario always begins with the target computer running Windows PE, which is how the tasks sequence knows the difference between the New Computer deployment scenario and the Refresh Computer deployments scenario. Figure 2 illustrates how UDI is used in the New Computer deployment scenario using the traditional OSD deployment methodology in Configuration Manager with the operating system image located on a distribution point.

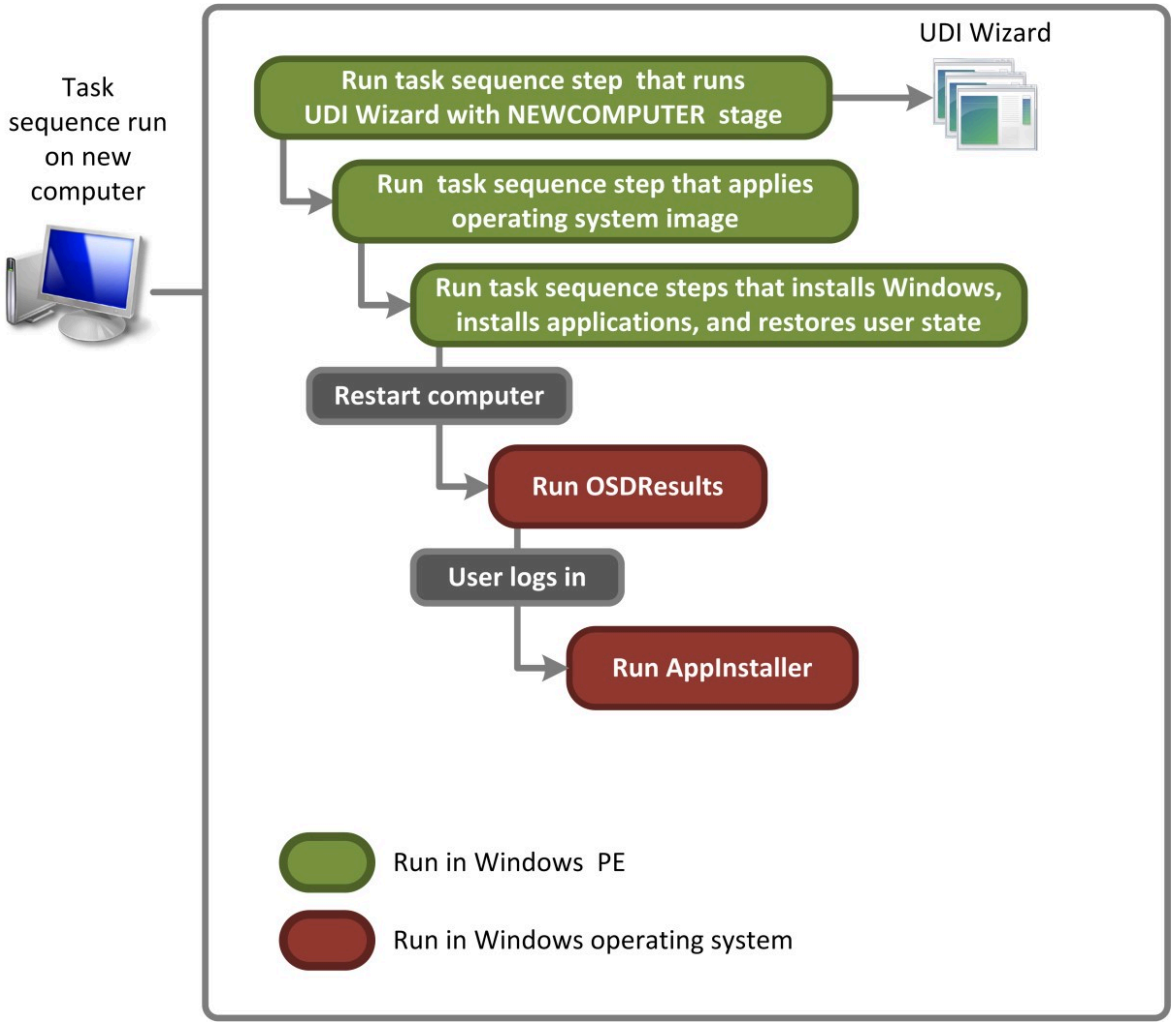


Figure 2. Process flow for UDI performing the New Computer deployment scenario for images stored on distribution points

Figure 3 illustrates how UDI is used in the New Computer deployment scenario using the prestaged media feature in Configuration Manager with the operating system image located on a local disk on the target computer.

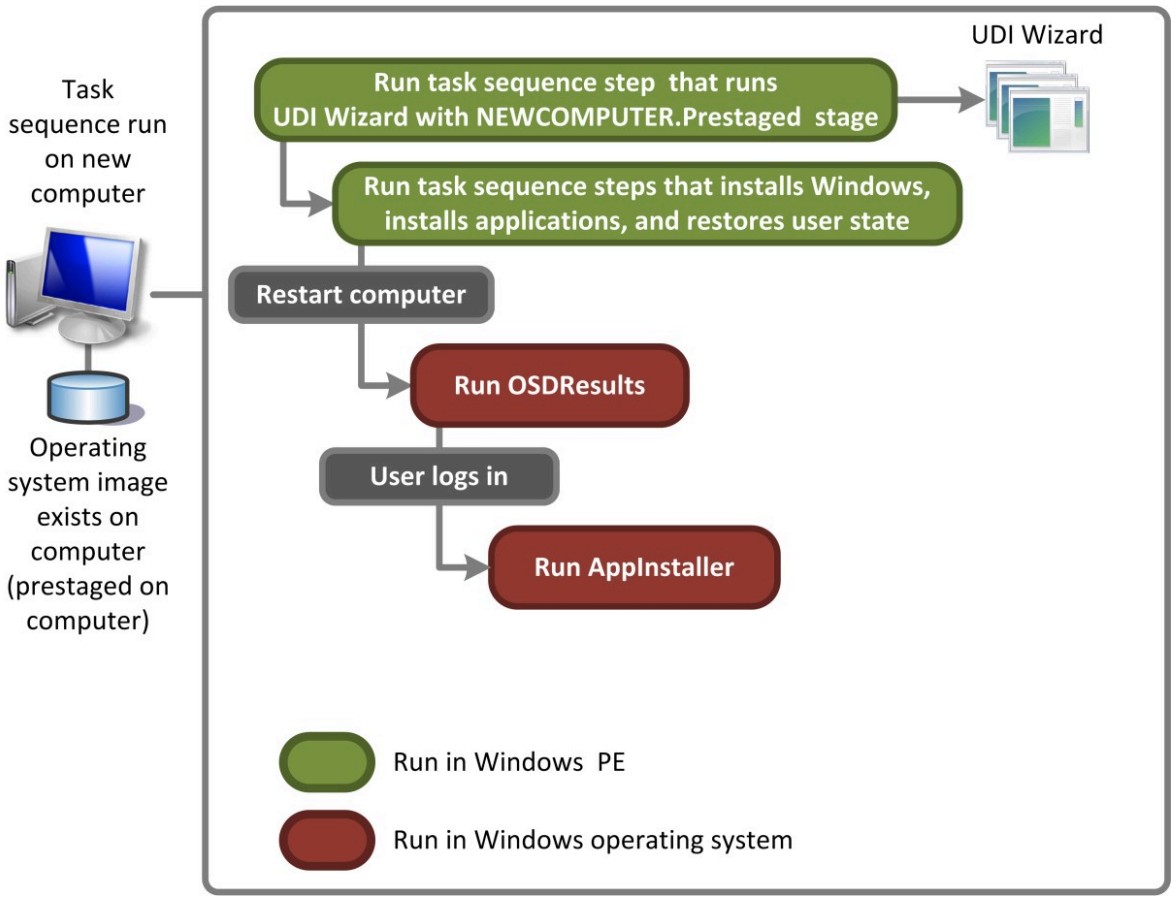


Figure 3. Process flow for UDI performing the New Computer deployment scenario for prestaged media

UDI in the Refresh Computer Deployment Scenario

Task sequences used to perform the Refresh Computer scenario use the same task sequence template as the New Computer scenario, the **User Driven Installation Task Sequence** task sequence template. The Refresh Computer deployment scenario always begins with the target computer running the existing Windows operating system, which is how the tasks sequence knows the difference between the Refresh Computer deployment scenario and the New Computer deployments scenario. Figure 4 illustrates how UDI is used in the Refresh Computer deployment scenario.

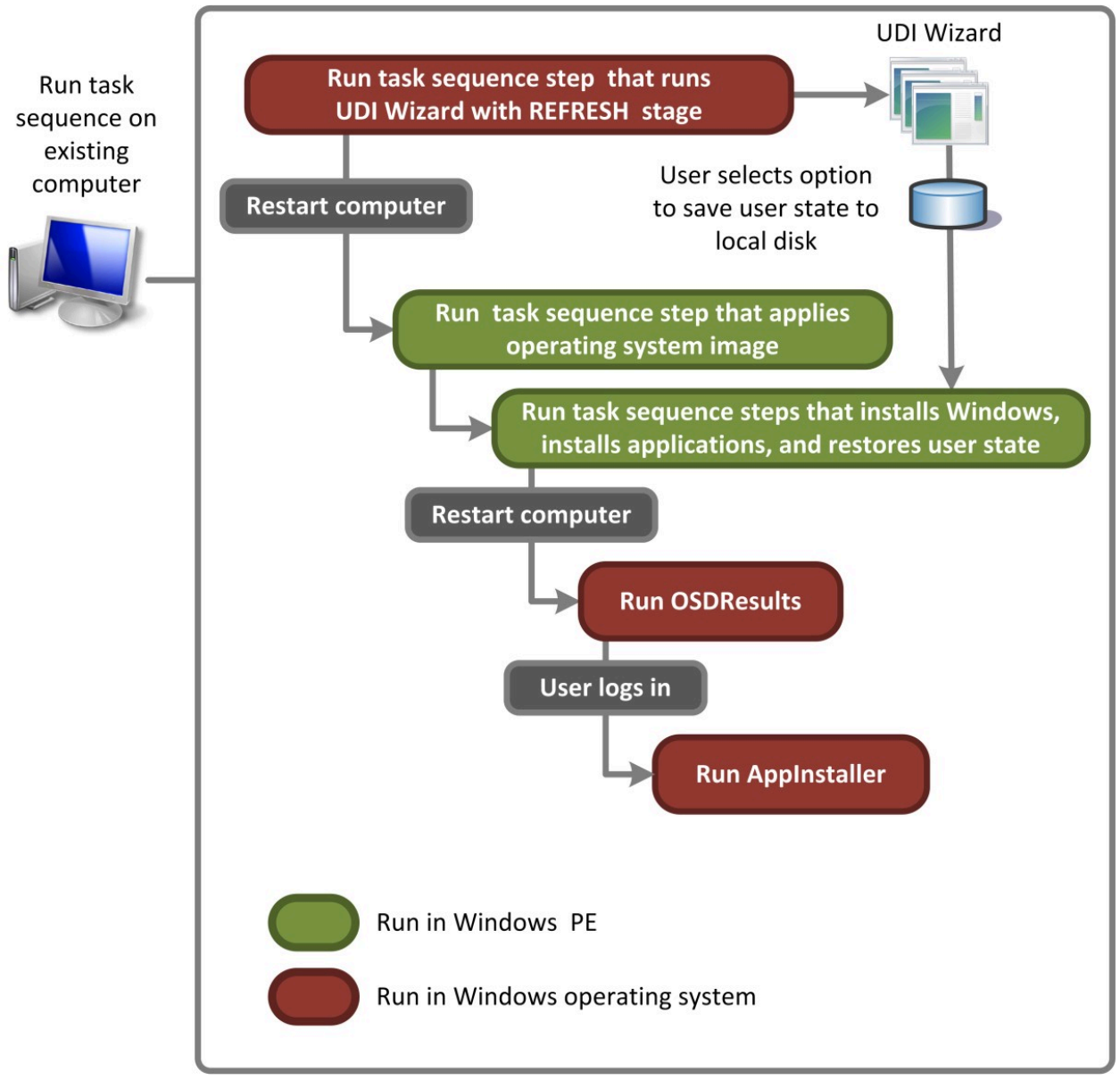


Figure 4. Process flow for UDI performing the Refresh Computer deployment scenario

UDI in the Replace Computer Deployment Scenario

The Replace Computer scenario requires the following task sequences.

- A task sequence created using the **User Driven Installation Replace Task Sequence** task sequence template. This task sequence is run first on the existing computer and is used to capture user state migration data to a network shared folder or to a USB disk that is attached to the existing computer.
- A task sequence created using the **User Driven Installation Task Sequence** task sequence template. This task sequence is run second on the replacement computer and is used to install the operating system, install the applications, and to restore user state migration data saved by the task sequence run on the existing computer.

Figure 5 illustrates how UDI is used in the Replace Computer deployment scenario.

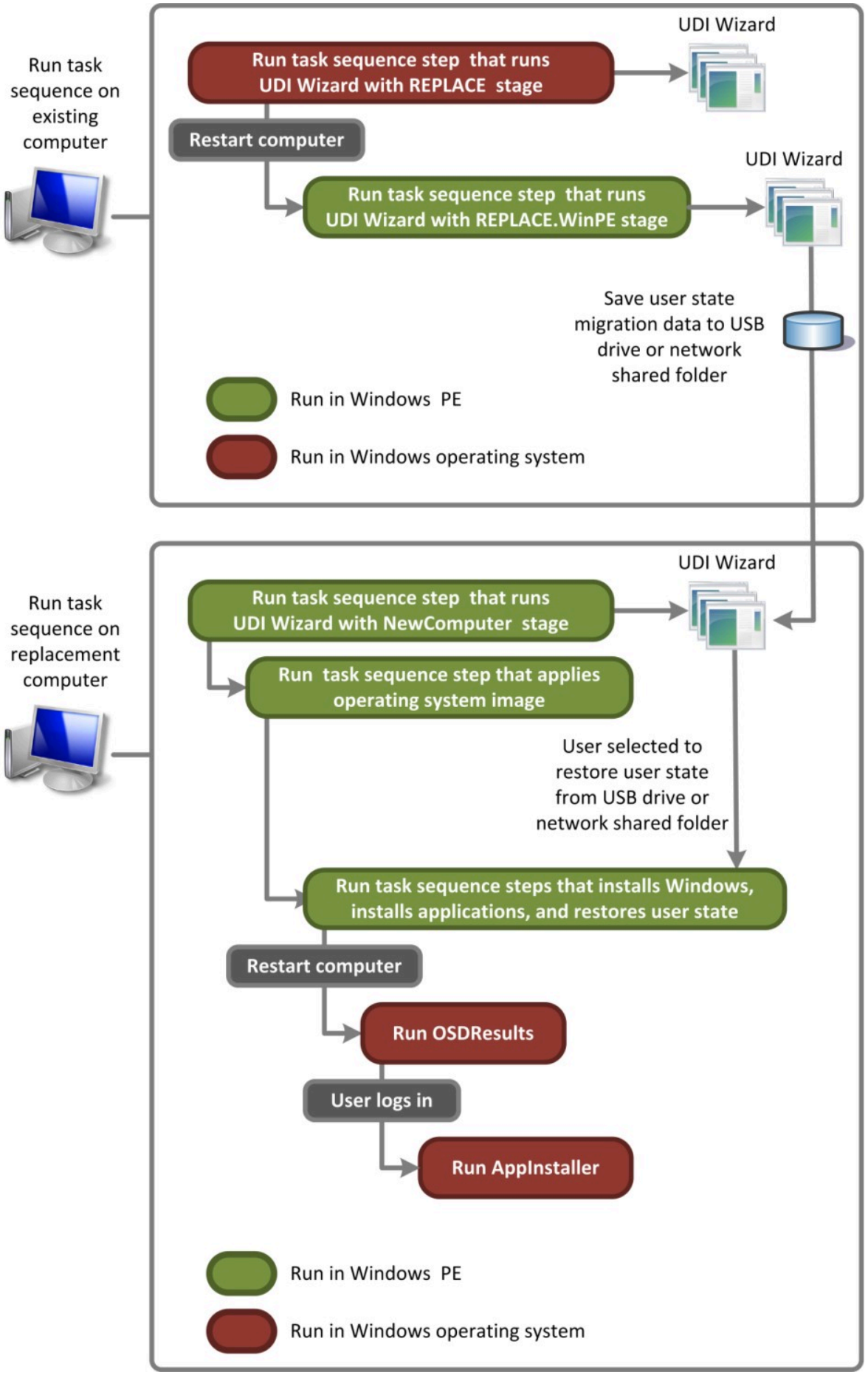


Figure 5. Process flow for UDI performing the Replace Computer deployment scenario

Overview of Built-in UDI Components

UDI comes with built-in Configuration Manager task sequences, stage groups, stages, tasks, validators, and wizard pages that can perform most common deployment scenarios without the assistance of a developer. These built-in components can be configured using the UDI Wizard Designer:

For more information about the built-in UDI components, see the following sections in the *Microsoft Deployment Toolkit Reference*:

- "UDI Stage Reference"
- "UDI Wizard Page Reference"
- "UDI Task Reference"
- "UDI Validator Reference"

  In addition to these built-in components, you can create custom wizard pages, wizard page editors, tasks, and validators using the UDI software development kit (SDK). The UDI SDK is installed with MDT and contains example solutions for Microsoft Visual Studio 2010. For more information about extending UDI using the UDI SDK, see the MDT document *User-Driven Installation Developers Guide*.

Preparing the UDI Deployment Environment

After you have prepared the prerequisite infrastructure for MDT, you are ready to prepare the MDT deployment environment for UDI.

To prepare the MDT deployment environment for UDI deployments

1. Preparing the prerequisite infrastructure as described in Prepare the Prerequisite Infrastructure for UDI Deployments.
2. Install a new instance of MDT on the deployment server, or upgrade an existing instance of MDT to MDT as described in Install or Upgrade to MDT for UDI Deployments.
3. Obtain the software that UDI requires as described in Obtain the Software That the UDI Deployment Process Requires.
4. Enable Configuration Manager console integration with MDT as described in Enable Configuration Manager Console Integration for UDI.

Prepare the Prerequisite Infrastructure for UDI Deployments

UDI deployments require that a properly configured Configuration Manager infrastructure exist prior to installing MDT and performing deployments. Ensure that your new or existing Configuration Manager infrastructure is specifically optimized for the Operating System Deployment feature.

> ⓘ **Note**
>
> Windows PowerShell version 2.0 or later must be installed on any computer on which MDT is installed for management of UDI deployments.

For more information about:

- Hardware and software requirements for Configuration Manager, see Supported Configurations for Configuration Manager.
- Configuring a Configuration Manager infrastructure to support UDI deployments, see the section, "Step 1: Prepare the Prerequisite Infrastructure", in the MDT document Quick Start Guide for User-Driven Installation.

Install or Upgrade to MDT for UDI Deployments

The first step in performing UDI deployments is to have at least one instance of MDT running in your environment. Install MDT on each computer that has the Configuration Manager console installed and that you will use to create or edit task sequences that MDT generates. If your existing environment has:

- No computers currently running MDT or a previous version of MDT, install one or more new instances of MDT as described in Installing a New Instance of MDT.
- One or more computers running a previous version of MDT, upgrade those instances to MDT as described in Upgrading to MDT. After the upgrade process is complete:
    - **Run the Configure ConfigMgr Integration Wizard**. This wizard must be run after the upgrade to register the new components and install the UDI new task sequence templates.
    - **Ensure you create a new Microsoft Deployment Toolkit Files package for any new UDI task sequences you create**. The existing Microsoft Deployment Toolkit Files package can be used for any UDI task sequences created prior to the

upgrade, but a new Microsoft Deployment Toolkit Files package must be created for new UDI task sequences.

- **Ensure any UDI task sequences created prior to the upgrade use the Microsoft Deployment Toolkit Files package that existed prior to the upgrade**. You can modify these UDI task sequences, but you cannot use any of the new MDT task sequence actions or steps. To use the new MDT task sequence actions or steps, create a new UDI task sequence.

Obtain the Software That the UDI Deployment Process Requires

Collect the software needed during the UDI deployment process. This software will be imported or added to deployment shares unless it already exists in the deployment share.

> ⓘ **Note**
>
> UDI requires Configuration Manager.

Required software includes:

- Operating system source files for each operating system to be deployed to the reference and target computers
- Operating system packages for the operating systems, such as security updates, feature packs, and language packs
- Device drivers for the reference and target computers that are not included as part of the operating system
- Applications that are to be installed as a part of the operating system image or during the deployment of the reference image
- USMT source files used to create a software package that is deployed to the target computers to capture user state migration data

Enable Configuration Manager Console Integration for UDI

Before you can use the Configuration Manager integration features of MDT, run the Configure ConfigMgr Integration Wizard. This wizard copies the appropriate Configuration Manager integration files to the *Configuration Manager _root* (where

Configuration Manager _root is the folder in which the Configuration Manager console is installed).

The wizard also adds WMI classes for the new MDT custom actions. You add these classes by compiling a .mof file that contains the new class definitions.

To run the Configure ConfigMgr Integration Wizard

> **Note**
>
> The Configuration Manager console should be closed when performing this procedure.

- Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Configure ConfigMgr Integration**.

  The Configure ConfigMgr Integration Wizard starts.

- Complete the Configure ConfigMgr Integration Wizard using the information in Table 137. Accept the default values unless otherwise specified.

Table 137. Information for Completing the Configure ConfigMgr Integration Wizard

Expand table

On this wizard page	Do this
Options	1. Verify that the **Install the MDT console extensions for System Center 2012 R2 Configuration Manager** check box is selected. 2. Verify that the **Add the MDT task sequence actions to a System Center 2012 R2 Configuration Manager server** check box is selected. 3. In **Site server name**, type ***ConfigMgr_server_name*** (where *ConfigMgr_server_name* is the name of the Configuration Manager server on which to install MDT integration). 4. In **Site code**, type ***ConfigMgr_site_code*** (where *ConfigMgr_site_code* is the Configuration Manager site code that installs MDT integration). 5. Select **Next**. The Site Server Name and Site Code fields will be automatically

On this wizard page	Do this
	populated with the most recent connection details if the Configuration Manager console has been opened once.
Confirmation	Select **Finish**.

The Configure ConfigMgr Integration Wizard finishes, and MDT is integrated with Configuration Manager.

Preparing for UDI Deployment to the Reference Computer

Regardless of the MDT deployment scenario you are performing using UDI, always start by creating a reference computer, and then capturing an image of that computer. Later in the MDT deployment process, you will deploy the captured image of your reference computer to the appropriate target computers. In addition, you can use existing operating system images in WIM format.

Create a reference computer for each image that you want to create for deployment to the target computers. For more information about determining the number of images required in your organization (and subsequently the number of reference computers required), see Estimate Storage Requirements for Configuration Manager Distribution Points. For more information about the use of reference computers in MDT deployments, see Using Reference Computers in MDT Deployments.

To prepare for deployment to the reference computer

1. Create a new task sequence that will deploy the target operating system to the reference computer using the Create MDT Task Sequence Wizard in the Configuration Manager console as described in Creating a UDI Task Sequence Using MDT Task Sequence Templates.

   > **Tip**
   >
   > Create the task sequence for deploying to the reference computer based on the User-Driven Installation task sequence template included in MDT.

2. Configure Configuration Manager to contain the appropriate software for deployment to the reference computer, including the following:

- Configuring applications and operating system packages as described in Managing Software Packages in Configuration Manager which is the same process for UDI and ZTI deployments.
- Configuring device drivers as described in Managing Device Drivers in Configuration Manager, which is the same process for UDI and ZTI deployments.

3. Ensure that all packages, including operating system deployment packages, that the new UDI task sequence uses are properly distributed to the assigned distribution points as described in Managing Distribution Points in Configuration Manager, which is the same process for UDI and ZTI deployments.

   > ⓘ **Note**
   >
   > Most production networks have multiple distribution points. When performing this step in a production environment, select the appropriate distribution points for the network.

4. Customize the MDT configuration files to the needs of your organization as described in Configuring MDT Deployments.

   > ⓘ **Important**
   >
   > If you are capturing an image of the reference computer, you must at least add the **DoCapture** property to the CustomSettings.ini file for the task sequence by specifying `DoCapture=YES` or `DoCapture=SYSPREP`.

5. Optionally, enable monitoring of the MDT deployment process as described in Monitoring MDT Deployments.

6. Customize the task sequence to the needs of your organization as described in Configuring ZTI Task Sequence Steps in Configuration Manager, which is the same process for UDI and ZTI deployments.

   > ⓘ **Note**
   >
   > The UDI deployment process is unable to perform Sysprep operations on a target computer that is encrypted by using BitLocker Drive Encryption. Do not enable BitLocker on the reference computer, and enable BitLocker on the

target computers only after the target operating system is completely deployed.

Deploying To and Capturing an Image of the Reference Computer Using UDI

After the distribution points are updated, advertise the task sequence to the reference computer and start the reference computer with the bootable Windows PE image created earlier in the process. The task sequence created earlier will deploy the target operating system, device drivers, operating system packages, and applications to the reference computer, and then capture an image of the reference computer.

To deploy to and capture an image of the reference computer

1. Add the reference computer to the Configuration Manager site database as described in Manually Adding Computers to the Site Database in Configuration Manager , which is the same process for UDI and ZTI deployments.
2. Create a collection that contains the reference computer as described in Managing Computer Collections in Configuration Manager, which is the same process for UDI and ZTI deployments.
3. Deploy the task sequence to the reference computer as described in Managing Task Sequence Deployment in Configuration Manager, which is the same process for UDI and ZTI deployments.
4. Optionally, enable monitoring of the MDT deployment process as described in Monitoring MDT Deployments.
5. Create a task sequence bootable media disk by using the Task Sequence Media Wizard as described in Creating Task Sequence Bootable Media in Configuration Manager, which is the same process for UDI and ZTI deployments.
6. Start the reference computer with the task sequence bootable media disk as described in Deploying an Operating System Using Task Sequence Bootable Media in Configuration Manager which is the same process for UDI and ZTI deployments.

Preparing for UDI Deployment to Target Computers

After the images of the reference computers are captured, deploy them to the target computers. In preparation for deploying the captured images to the target computers,

create one or more task sequences for deploying the captured images, ensure that the necessary deployment resources exist, and customize the MDT deployment process.

To prepare for UDI deployment to target computers

1. Prepare network shares for storing migration data and MDT deployment logs as described in Preparing the MDT Migration Resources.

2. Optionally, prepare Windows Deployment Services to start the appropriate Windows PE images that will in turn start the UDI deployment process to the target computers as described in Preparing Windows Deployment Services for UDI Deployments.

3. Create additional distribution points to help in larger deployments as described in Managing Distribution Points in Configuration Manager, which is the same process for UDI and ZTI deployments.

4. Prepare the UDI task sequences, the MDT configuration files, the UDI Wizard configuration files, and the MDT DB for each deployment scenario as described in the following sections:

    - Prepare for the New Computer Deployment Scenario to Target Computers Using UDI
    - Prepare for the Refresh Computer Deployment Scenario to Target Computers Using UDI
    - Prepare for the Replace Computer Deployment Scenario to Target Computers Using UDI

      Depending on the target computers in your organization, any combination of these deployments scenarios might be necessary. For more information about MDT deployment scenarios, see Identifying Deployment Scenarios.

Prepare for the New Computer Deployment Scenario to Target Computers Using UDI

In the New Computer deployment scenario, you deploy a new installation of a Windows operating system to a new computer. There is no user migration information to save and restore and no existing file systems to preserve. Use the User-Driven Installation task sequence template to deploy the captured image of the reference computer to the target computer.

To prepare for the New Computer deployment scenario to target computers using UDI

1. Create a new task sequence that will deploy the target operating system to the reference computer using Create MDT Task Sequence Wizard in the Configuration Manager Console as described in Creating a ZTI Task Sequence Using MDT Task Sequence Templates in Configuration Manager, but ensure that you specifically follow the configuration settings on the wizard pages listed in Table 138 and select the appropriate values on the other wizard pages based on your organization's requirements.

   > **Tip**
   >
   > Create the task sequence for deploying to the reference computer based on the Client Task Sequence task sequence template included in MDT.

Table 138. Information for Completing the Create MDT Task Sequence Wizard for Performing New Computer Deployment Scenario Using UDI

Expand table

On this wizard page	Do this
OS Image	Select **Create a new OS image**, and specify the fully qualified UNC path to the WIM image captured from the reference computer.
Deployment Method	Select **Perform a "User-Drive Installation"**.

2. Configure the application and operating system packages for deployment to the reference computer, including:

   - Configuring applications and operating system packages as described in Managing Software Packages in Configuration Manager, which is the same process for UDI and ZTI deployments.
   - Configuring device drivers as described in Managing Device Drivers in Configuration Manager, which is the same process for UDI and ZTI deployments.

3. Ensure that all packages, including operating system deployment packages, that the new UDI task sequence uses are properly distributed to the assigned distribution points as described in Managing Distribution Points in Configuration Manager, which is the same process for UDI and ZTI deployments.

   > **Note**
   >
   > Most production networks have multiple distribution points. When performing this step in a production environment, select the appropriate distribution points for the network.

4. Customize the MDT configuration files to the needs of your organization as described in Configuring MDT Deployments.

5. Optionally, customize the MDT DB to the needs of your organization as described in Performing Deployments Using the MDT DB (if you are using the MDT DB to provide MDT configuration information).

6. Customize the task sequence to the needs of your organization as described in Configuring ZTI Task Sequence Steps in Configuration Manager, which is the same process for UDI and ZTI deployments.

7. Customize the UDI Wizard configuration files to the needs of your organization as described in Configuring UDI Wizard Behavior.

8. Optionally, enable monitoring of the MDT deployment process as described in Monitoring MDT Deployments.

9. Ensure that all packages, including operating system deployment packages, that the new UDI task sequence uses are properly distributed to the assigned distribution points as described in Managing Distribution Points in Configuration Manager, which is the same process for UDI and ZTI deployments.

   > **Note**
   >
   > Most production networks have multiple distribution points. When performing this step in a production environment, select the appropriate distribution points for the network.

10. Update the distribution points so that any changes to the packages are distributed properly as described in Managing Distribution Points in Configuration Manager, which is the same process for UDI and ZTI deployments.

> **Note**
>
> Most production networks have multiple distribution points. When performing this step in a production environment, select the appropriate distribution points for the network.

Prepare for the Refresh Computer Deployment Scenario to Target Computers Using UDI

In the Refresh Computer deployment scenario, a computer is refreshed, including computers that must be re-imaged for image standardization or to address a problem. There is user migration information to save and restore but no existing file systems to preserve. Use the **User Driven Installation Task Sequence** template to deploy the captured image of the reference computer to the target computer.

To prepare for the Refresh Computer deployment scenario to target computers using UDI

1. Create a new task sequence that will deploy the target operating system to the reference computer using the Create MDT Task Sequence Wizard in the Configuration Manager console as described in Creating a UDI Task Sequence Using MDT Task Sequence Templates, but ensure that you follow the configuration settings on the wizard pages listed in Table 139 and select the appropriate values on the other wizard pages for your organization's requirements.

   > **Tip**
   >
   > Create the task sequence for deploying to the reference computer based on the Client Task Sequence task sequence template included in MDT.

Table 139. Information for Completing the Create MDT Task Sequence Wizard for Performing Refresh Computer Deployment Scenario Using UDI

Expand table

On this wizard page	Do this
OS Image	Select **Create a new OS image**, and specify the fully qualified path to the WIM image captured from the reference computer.
Deployment Method	Select **Perform a "User-Drive Installation"**.

2. Configure the appropriate software for deployment to the target computer in the Configuration Manager console, including:

    - Configuring applications and operating system packages as described in Managing Software Packages in Configuration Manager
    - Configuring device drivers as described in Managing Device Drivers in Configuration Manager

3. Optionally, customize the MDT configuration files or the MDT DB to the needs of your organization as described in:

    - Configuring MDT Deployments
    - Performing Deployments Using the MDT DB

4. Customize the task sequence to the needs of your organization as described in Configuring ZTI Task Sequence Steps in Configuration Manager, which is the same process for UDI and ZTI deployments.

5. Configure the behavior of the UDI Wizard to the needs of your organization as described in Configuring UDI Wizard Behavior.

6. Optionally, enable monitoring of the MDT deployment process as described in Monitoring MDT Deployments.

7. Ensure that all packages, including operating system deployment packages, that the new UDI task sequence uses are properly distributed to the assigned distribution points as described in Managing Distribution Points in Configuration Manager, which is the same process for UDI and ZTI deployments.

    > ⓘ **Note**
    >
    > Most production networks have multiple distribution points. When performing this step in a production environment, select the appropriate distribution points for the network.

8. Update the distribution points so that any changes to the packages are distributed properly as described in Managing Distribution Points in Configuration Manager, which is the same process for UDI and ZTI deployments.

   > **Note**
   >
   > Most production networks have multiple distribution points. When performing this step in a production environment, select the appropriate distribution points for the network.

Prepare for the Replace Computer Deployment Scenario to Target Computers Using UDI

In the Replace Computer deployment scenario, one computer replaces another computer. Create a computer association record that associates the existing target computer and the new target computer. The existing user state migration data is saved from the existing target computer. Then, a new installation of Windows is deployed to a new computer. Finally, the user state data is restored to the new computer. There are no existing file systems to preserve.

> **Important**
>
> You must establish a computer association record for each existing target computer and each new target computer prior to performing the deployment to the target computer.

Use the:

- User Driven Installation Replace Task Sequence template to save the user state migration of the existing target computer

  > **Important**
  >
  > Run this task sequence before running the task sequence based on the User Driven Installation Task Sequence template on the new target computer.

- Client Task Sequence template to deploy the captured image of the reference computer to the new target computer and restore the user state migration data

  > **Important**

Run this task sequence after running the task sequence based on the User Driven Installation Replace Task Sequence template on the existing target computer.

To prepare for the Replace Computer deployment scenario to target computers using UDI

1. Create a computer association between the existing target computer and the new target computer as described in "How to Perform a Side-by-Side Operating System Deployment," in the section, "How to Deploy Operating Systems in Configuration Manager," in Configuration Manager Documentation Library, which is installed with Configuration Manager.

2. Create a new task sequence that will save the user state migration data of the existing target computer as described in Creating a UDI Task Sequence Using MDT Task Sequence Templates.

   > **Tip**
   >
   > Create the task sequence for capturing the user state migration data from the target computer based on the User Driven Installation Task Replace Sequence template included in MDT.

3. Create a new task sequence that will deploy the captured image of the reference computer to the target computer, and restore the user state migration data saved by the **User Driven Installation Replace Task Sequence** as described in Creating a UDI Task Sequence Using MDT Task Sequence Templates, but ensure that you specifically follow the configuration settings on the wizard pages listed in Table 140 and select the appropriate values on the other wizard pages for your organization's requirements.

   > **Tip**
   >
   > Create the task sequence for deploying to the target computer based on the ClientTask Sequence template included in MDT.

Table 140. Information for Completing the Create MDT Task Sequence Wizard for Performing the Replace Computer Deployment Scenario using UDI

Expand table

On this wizard page	Do this
OS Image	Select **Create a new OS image**, and specify the fully qualified UNC path to the WIM image captured from the reference computer.
Deployment Method	Select **Perform a "User-Drive Installation"**.

4. Configure the appropriate software for deployment to the target computer in the Configuration Manager console, including:
    - Configuring applications and operating system packages as described in Managing Software Packages in Configuration Manager, which is the same process for UDI and ZTI deployments.
    - Configuring device drivers as described in Managing Device Drivers in Configuration Manager, which is the same process for UDI and ZTI deployments.
5. Customize the MDT configuration files or the MDT DB to the needs of your organization as described in:
    - Configuring MDT Deployments
    - Performing Deployments Using the MDT DB
6. Customize the task sequences to the needs of your organization as described in Configuring ZTI Task Sequence Steps in Configuration Manager, which is the same process for UDI and ZTI deployments.
7. Optionally, enable monitoring of the MDT deployment process as described in Monitoring MDT Deployments.
8. Ensure that the distribution points for the packages and operating system images that the new ZTI task sequence uses are distributed properly as described in Managing Distribution Points in Configuration Manager, which is the same process for UDI and ZTI deployments.

> **Note**
>
> Most production networks have multiple distribution points. When performing this step in a production environment, select the appropriate

distribution points for the network.

9. Update the distribution points so that any changes to the packages are distributed properly as described in Managing Distribution Points in Configuration Manager, which is the same process for UDI and ZTI deployments.

> **Note**
>
> Most production networks have multiple distribution points. When performing this step in a production environment, select the appropriate distribution points for the network.

Deploying Captured Images to Target Computers Using UDI

The deployment of the captured images to the target computers is slightly different for each MDT deployment scenario using UDI. Deploy the captured image of the reference computer to target computers for each respective deployment scenario in your organization.

To deploy the capture image of the reference computer to the target computers using UDI

1. Add the target computer to the Configuration Manager site database:

   - Manually, as described in Manually Adding Computers to the Site Database in Configuration Manager, which is the same process for UDI and ZTI deployments.

   - Automatically in Configuration Manager as described in the section, "How to Manage Unknown Computer Deployments in Configuration Manager," in the Configuration Manager Documentation Library, which is installed with Configuration Manager.

> **Note**
>
> If the target computers already exist in the Configuration Manager Site database, then this step is not necessary.

2. Create a collection that contains the target computers as described in Managing Computer Collections in Configuration Manager, which is the same process for UDI and ZTI deployments.

   > **Note**
   >
   > Create a collection for each MDT deployment scenario to be performed, and ensure that the collection includes the target computers requiring the corresponding deployment scenario.

3. Create an advertisement for the target computer task sequences as described in Managing Task Sequence Deployment in Configuration Manager, which is the same process for UDI and ZTI deployments.

4. Provide a method for starting the target computers by doing any combination of the following:

   - Create a task sequence bootable media disk using the Task Sequence Media Wizard as described in Creating Task Sequence Bootable Media in Configuration Manager, which is the same process for UDI and ZTI deployments.

   - Prepare Windows Deployment Services to start the appropriate Windows PE images that will in turn start the UDI deployment process to the target computers as described in Preparing Windows Deployment Services for ZTI Deployments Using Configuration Manager, which is the same process for UDI and ZTI deployments.

5. Deploy the captured reference computer image to the target computers for each deployment scenario as described in:

   - Deploy Captured Images to Target Computers in the New Computer Deployment Scenario Using UDI

   - Deploy Captured Images to Target Computers in the Refresh Computer Deployment Scenario Using UDI

   - Deploy Captured Images to Target Computers in the Replace Computer Deployment Scenario Using UDI

     Depending on the target computers in your organization, any combination of deployments scenarios might be necessary. For more information about the MDT deployment scenarios, see Identifying Deployment Scenarios.

Deploy Captured Images to Target Computers in the New Computer Deployment Scenario Using UDI

Start the target computer with the task sequence bootable media created earlier in the process or from Windows Deployment Services. Either method starts Windows PE on the target computer and initiates the UDI deployment process. At the end of the process, the captured image of the reference computer is deployed on the target computer.

To deploy the capture images to the target computers in the New Computer Deployment Scenario using UDI

1. Start the target computer with the task sequence bootable media created earlier in the process or from Windows Deployment Services.

   The Task Sequence Wizard starts.

2. Complete the Task Sequence Wizard, ensuring that you specifically follow the configuration settings on the wizard pages listed in Table 141 and select the appropriate values on the other wizard pages for your organization's requirements.

   > **Note**
   >
   > This wizard will not appear if you configure UDI to perform a PXE boot and have configured a mandatory advertisement or if only one task sequence is advertised to the target computer.

Table 141. Information for Completing the Task Sequence Wizard in the New Computer Deployment Scenario using UDI

Expand table

On this wizard page	Do this
Select a Task Sequence	Select the task sequence you created for the target computer deployment for the New Computer deployment scenario.

The wizard starts, and the operating system deployment starts. Eventually the task sequence starts the UDI Wizard.

3. Optionally, view the MDT deployment process using the Monitoring node in the Deployment Workbench or using the **Get-MDTMonitorData** cmdlet.

   For more information about monitoring MDT deployments, see View MDT Deployment Progress.

4. Complete the UDI Wizard by selecting the appropriate values on the wizard pages for your organization's requirements as described in Running the UDI Wizard.

Deploy Captured Images to Target Computers in the Refresh Computer Deployment Scenario Using UDI

Start this scenario by running the Configuration Manager task sequence deployment (advertisement) for capturing the user state migration data that you created earlier in the process. This task sequence runs in the current operating system on the existing target computer.

To deploy the capture images to the target computers in the Refresh Computer Deployment Scenario Using UDI

1. On the target computer, run the Configuration Manager deployment (advertisement) for capturing the Refresh Computer deployment scenario that you created earlier in the deployment process.

   The task sequence starts. Eventually, the task sequence starts the UDI Wizard.

2. Complete the UDI Wizard by selecting the appropriate values on the wizard pages for your organization's requirements, as described in Running the UDI Wizard.

3. Optionally, view the MDT deployment process using the Monitoring node in the Deployment Workbench or using the **Get-MDTMonitorData** cmdlet.

   For more information about monitoring MDT deployments, see View MDT Deployment Progress.

   The task sequence runs in Windows PE to capture user state migration data. The task sequence restarts the computer, starts Windows PE, and then initiates installation of the new operating system. The task sequence restarts the computer, starts the new operating system, restores the user state migration data, installs any packages, installs any applications, and performs any other actions configured in the task sequence. Finally, the OSD Results program, OSDResults.exe, runs and displays the results of the deployment. The target computer is now deployed.

Deploy Captured Images to Target Computers in the Replace Computer Deployment Scenario Using UDI

The Replace Computer deployment scenario requires two separate steps to complete the migration. First, run the deployment (advertisement) for the task sequence you created to capture the user state migration data from the existing target computer (old computer). Second, run the UDI Wizard to deploy the captured image of the reference computer to the new target computer (new computer) and restore the user state saved earlier in the process.

To deploy captured images of the reference computer to target computers in the Replace Computer deployment scenario using UDI

1. Save the user state migration data from the existing target computer as described in Save the User State Migration Data in the Replace Computer Deployment Scenario Using UDI.

2. Deploy the captured image of the reference computer to the new target computer as described in Deploy the Captured Image and User State Migration Data in the Replace Computer Deployment Scenario Using UDI.

Save the User State Migration Data in the Replace Computer Deployment Scenario Using UDI

Start this scenario by running the task sequence deployment (advertisement) for capturing the user state migration data that you created earlier in the process. This task sequence runs in the current operating system on the existing target computer.

To save the user state migration data from the existing target computers in the Replace Computer Deployment Scenario using UDI

1. Run the task sequence deployment (advertisement) for capturing user state migration data that you created earlier in the process Refresh Computer deployment scenario.

2. Optionally, view the MDT deployment process using the Monitoring node in the Deployment Workbench or using the **Get-MDTMonitorData** cmdlet.

   For more information about monitoring MDT deployments, see View MDT Deployment Progress.

   The task sequence runs in the current operating system to capture user state migration data. At the end of the task sequence, the user state migration data of

the existing target computer is saved to the Configuration Manager state migration point.

Deploy the Captured Image and User State Migration Data in the Replace Computer Deployment Scenario Using UDI

Start the target computer with the ZTI bootable media created earlier in the process or from Windows Deployment Services. The ZTI bootable media starts Windows PE on the target computer and initiates the UDI deployment process. At the end of the deployment process, the captured image of the reference computer is deployed on the target computer, and the user state migration data is restored from the Configuration Manager state migration point.

To complete the Windows Deployment Wizard in the Replace Computer deployment scenario for deploying the captured image using UDI

1. Start the reference computer with the ZTI bootable media created earlier in the process or from Windows Deployment Services.

   Windows PE starts, and then the Windows Deployment Wizard starts.

2. Complete the Task Sequence Wizard, ensuring that you follow the configuration settings for the wizard pages listed in Table 142 and select values on the other wizard pages for your organization's requirements.

   > **Note**
   >
   > This wizard will not appear if you configure ZTI to perform a PXE boot and have configured a mandatory advertisement or if only one task sequence is advertised to the target computer.

Table 142. Information for Completing the Task Sequence Wizard for the Replace Computer Deployment Scenario Using UDI

Expand table

On this wizard page	Do this
Select a Task Sequence	Select the task sequence you created for the target computer deployment in the Replace Computer deployment scenario to deploy the captured image of the reference computer to the new target computer.

The task sequence starts. Eventually, the task sequence starts the UDI Wizard.

3. Complete the UDI Wizard by selecting the appropriate values on the wizard pages for your organization's requirements, as described in Running the UDI Wizard.

4. Optionally, view the MDT deployment process using the Monitoring node in the Deployment Workbench or using the **Get-MDTMonitorData** cmdlet.

   For more information about monitoring MDT deployments, see View MDT Deployment Progress.

   The task sequence starts Windows PE and then initiates installation of the new operating system. The task sequence restarts the computer, starts the new operating system, restores the user state migration data, installs any packages, installs any applications, and performs any other actions configured in the task sequence. Finally, the OSD Results program, OSDResults.exe, runs and displays the results of the deployment. The target computer is now deployed.

Managing UDI Deployments

You manage UDI deployments through the Configuration Manager console and the UDI Wizard Designer. You use the Deployment Workbench in UDI deployments only to configure the MDT DB. The wizard used to create UDI task sequences are integrated into the Configuration Manager console. You can use the UDI Wizard Designer to configure the behavior of the UDI Wizard.

Manage UDI deployments by:

- Reviewing UDI administration process as described in Overview of UDI Administration
- Creating a new task sequence for UDI deployments using the Create MDT Task Sequence Wizard as described in Creating a UDI Task Sequence Using MDT Task Sequence Templates
- Managing operating systems for UDI deployments in the Configuration Manager Console as described in Managing Operating Systems in Configuration Manager

which is the same process for UDI and ZTI deployments

- Managing device drivers for UDI deployments in the Configuration Manager Console as describe in Managing Device Drivers in Configuration Manager, which is the same process for UDI and ZTI deployments
- Deploying an operating system using task sequence bootable media as described in Deploying an Operating System Using Task Sequence Bootable Media in Configuration Manager, which is the same process for UDI and ZTI deployments
- Creating task sequence bootable media for UDI as described in Creating Task Sequence Bootable Media in Configuration Manager, which is the same process for UDI and ZTI deployments
- Creating boot images for use with UDI using the Create Image Using Microsoft Deployment Wizard as described in Creating ZTI Boot Images in Configuration Manager, which is the same process for UDI and ZTI deployments
- Managing software packages for UDI in the Configuration Manager console as described in Managing Software Packages in Configuration Manager, which is the same process for UDI and ZTI deployments
- Managing advertisements for UDI as described in Managing Task Sequence Deployment in Configuration Manager, which is the same process for UDI and ZTI deployments
- Manually adding computers to the site database for UDI as described in Manually Adding Computers to the Site Database in Configuration Manager, which is the same process for UDI and ZTI deployments
- Managing computer collections for UDI as described in Managing Computer Collections in Configuration Manager, which is the same process for UDI and ZTI deployments
- Managing distribution points for UDI as described in Managing Distribution Points in Configuration Manager, which is the same process for UDI and ZTI deployments
- Configuring individual UDI task sequence steps as described in Configuring ZTI Task Sequence Steps in Configuration Manager, which is the same process for UDI and ZTI deployments
- Configuring the UDI Wizard behavior by customizing the UDI Wizard configuration file as described in Configuring UDI Wizard Behavior

- Creating a custom wizard page to collect additional deployment information as described in Creating Custom Wizard Pages Using the Build Your Own Page Feature

Overview of UDI Administration

The goal of UDI administration is to configure the user experience in the UDI Wizard and ultimately control the deployment of Windows operating systems and applications to target computers. You configure the UDI user experience by using the UDI Wizard Designer and by customizing the Configuration Manager task sequences used with UDI in the Configuration Manager Console.

The primary tool for administering UDI is the UDI Wizard designer. The UDI Wizard Designer is installed as a part of MDT, which is installed on the same computer at the Configuration Manager Console. Because UDI is built-on the OSD feature in Configuration Manager, you will also use the Configuration Manager Console to administer specific aspects of UDI deployments.

Figure 6 illustrates the high-level overview of the UDI administrative process.

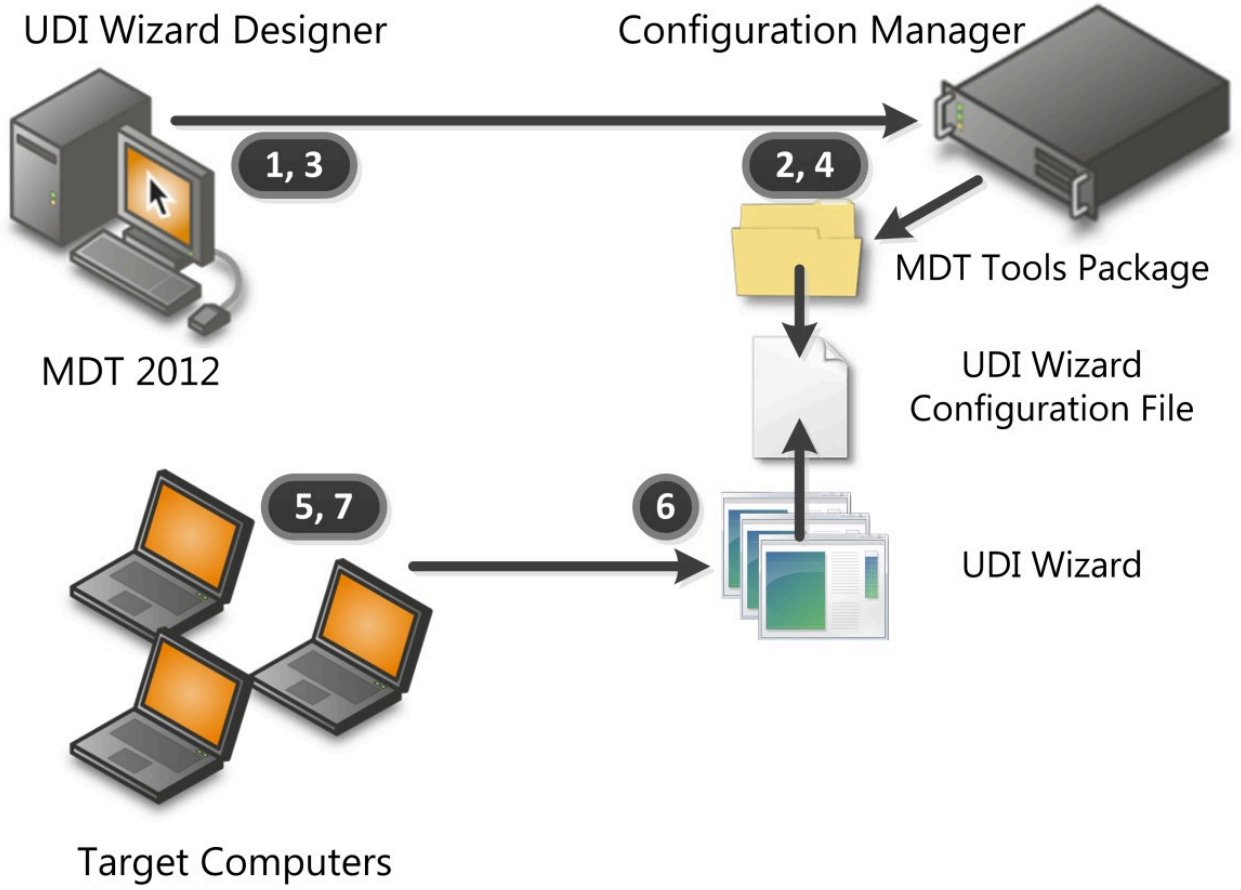


Figure 6. Overview of UDI administration process

The UDI administration process, as illustrated in Figure 6, is performed as follows:

1. Create a UDI task sequence based on the task sequence templates built-in to MDT.

   As a part of creating the task sequence, the **Create MDT Task Sequence** wizard creates an MDT toolkit package that contains the contents of the **installation_folder\Templates\Distribution** folder (where *installation_folder* is the folder where you installed MDT). The toolkit package is referenced by the **Use Toolkit Package** task sequence step.

   For more information about the built-in UDI tasks sequence templates in MDT, see Identify the UDI Task Sequence Templates in MDT.

2. Distribute the MDT toolkit package to Configuration Manager distribution points.

   The UDI Wizard and UDI Wizard configuration file are contained in the package. The UDI Wizard (UDIWizard.exe) is in the **Tools** folder in the package. The UDI Wizard configuration file (UDIWizard_Config.xml) is in the **Scripts** folder in the package.

3. Customize the UDI Wizard configuration file and application information file using the UDI Wizard Designer.

   The UDI Wizard configuration file (UDIWizard_Config.xml) and application information file (UDIWizard_Config.xml.app) are stored in the **Scripts** folder in the MDT toolkit package.

4. Update the distribution points with the modified version of the UDI Wizard configuration file and corresponding application information file in the MDT toolkit package.

5. The target computers initiate the UDI task sequence which runs the UDI Wizard at the appropriate point in the task sequence.

   The UDI Wizard is initiated using the **UDI Wizard** task sequence step.

6. The UDI Wizard runs and the deployment configuration information is collected from the user.

   The UDI Wizard reads the UDI Wizard configuration file to determine the wizard pages to display and the sequence of the pages. The user completes the UDI Wizard by providing the necessary deployment information. The UDI Wizard updates task sequence variables based on the information provided. The updated task sequence variables are used by the UDI task sequence to perform the balance of the deployment.

7. The remainder of the task sequence steps in the UDI task sequence complete and the **OSD Results** dialog is displayed at the end of the deployment. Any applications installed during the task sequences are identified the first time a user logs in using **AppInstaller**.

   AppInstaller enables Configuration Manager to identify any applications installed using the Application model during the task sequence. This allows Configuration Manager to use features such as the monitoring feature.

Creating a UDI Task Sequence Using MDT Task Sequence Templates

Use the Create MDT Task Sequence Wizard in the Configuration Manager console to create task sequences in Configuration Manager that are integrated with MDT. MDT includes task sequence templates that you can use to deploy the reference and target computers.

Create UDI task sequences using the MDT task sequence templates by:

- Identifying the UDI task sequence templates that are a part of MDT as described in Identify the UDI Task Sequence Templates in MDT
- Identifying the packages and images that the UDI task sequence templates require as described in Identify the Packages and Images That the UDI Task Sequence Templates Require
- Creating UDI task sequences as described in Create UDI Task Sequences Using the Create MDT Task Sequence Wizard
- Configure UDI task sequences to deploy different operating systems as described in Configure UDI Task Sequences to Deploy Different Operating Systems

Identify the UDI Task Sequence Templates in MDT

MDT includes task sequence templates that are used to create MDT task sequences in Configuration Manager. The task sequence templates included in MDT are described in Identify the Task Sequence Templates in MDT in Configuration Manager.

Of the templates described in Identify the Task Sequence Templates in MDT in Configuration Manager, the following are used in MDT deployment scenarios using UDI:

- **Client Task Sequence**. This task sequence template is used for the MDT New Computer, Refresh Computer, and Replace Computer deployment scenarios. This

task sequence template is also used to build and capture images of the reference computers.

- **User Driven Installation Replace Task Sequence**. This task sequence template is the first step in a two-step process in the MDT Replace Computer deployment scenario and is used to capture user state migration data. The second step in the two step process is the User Driven Installation Task Sequence task sequence template, which is used to:
  - Deploy the target applications and operating system
  - Restore the user state migration data saved during the User Driven Installation Replace task Sequence task sequence template in the first step of the process

  For more information on the MDT deployment scenarios, see Identifying Deployment Scenarios.

> **Note**
>
> Always use the Create MDT Task Sequence Wizard to create task sequences. Although you can manually create the task sequences, doing so is not recommended.

Identify the Packages and Images That the UDI Task Sequence Templates Require

The UDI task sequence templates require the same packages and images as required by ZTI deployments, as described in Identify the Packages and Images That the MDT Task Sequence Templates in Configuration Manager Require.

Create UDI Task Sequences Using the Create MDT Task Sequence Wizard

The Create MDT Task Sequence Wizard substitutes the packages and images selected for the placeholders in the task sequence templates. After completing the wizard, the new task sequence references the appropriate packages and images.

> **Note**
>
> Always use the Create MDT Task Sequence Wizard to create task sequences based on the MDT task sequence templates. Although you can manually import the task

sequence templates, doing so is not recommend.

Create UDI task sequences using the same process for creating ZTI task sequences using the Create MDT Task Sequence Wizard as described in Create ZTI Task Sequences Using the Create MDT Task Sequence Wizard in Configuration Manager

Select the appropriate UDI task sequence template based on the deployment scenario being performed. For more information about the UDI task sequence templates in MDT, see Identify the UDI Task Sequence Templates in MDT.

Configure UDI Task Sequences to Deploy Different Operating Systems

The **VolumePage** page in built-in UDI stage groups allows you to select from any operating system images that you have configured in the UDI Wizard Designer. However, the task sequence created by the Create MDT Task Sequence wizard references only one specific operating system image in the **Apply Operating System Image** task sequence step.

When you select the operating system image on the **VolumePage** page, the UDI Wizard sets the **OSDImageName** task sequence variable to the value of the image name that was selected. The value of the **OSDImageName** task sequence variable corresponds to the name of the operating system image in the **Operating System Images** or **Operating System Installers** nodes in the Configuration Manager console.

You can configure a UDI task sequence to support the operating systems you have added to the **VolumePage** page by performing the following steps:

1. Rename the existing **Apply Operating System Image** task sequence step to reflect the name of the operating system image being deployed.
2. Configure a condition for the existing **Apply Operating System Image** task sequence step that will only run the step when the **OSDImageName** task sequence variable is equal to the name of the operating system image being deployed.
3. For each operating system that has been added to the **VolumePage** page, perform the following steps:
    a. Add a new **Apply Operating System Image** task sequence step that reflects the name of the operating system image to be deployed.
    b. Configure a condition for the new **Apply Operating System Image** task sequence step that will only run the step when the **OSDImageName** task

sequence variable is equal to the name of the operating system image to be deployed.

After performing these steps, when the user selected an operating system image on the **VolumePage** page, the corresponding **Apply Operating System Image** task sequence step will be run and deploy the appropriate operating system image.

Configuring UDI Wizard Behavior

The User-Driven Installation Task Sequence and User-Driven Installation Replace Task Sequence templates include task sequence steps that run the UDI Wizard. When a task sequence step runs the UDI Wizard, the step also references the UDIWizard_Config.xml file, which controls the behavior of the UDI Wizard and is stored in the Scripts folder of the MDT files package. You can customize the UDIWizard_Config.xml file using the UDI Wizard Designer.

Configure the UDI Wizard behavior by performing the following steps in the UDI Wizard Designer:

- Review the UDI Wizard Designer concepts as described in Review UDI Wizard Designer Concepts.
- Identify the UDI components used in performing UDI deployments and the relationship between those components as described in Identify UDI Deployment Process Components.
- Review the relationship between the UDI wizard pages, UDI wizard page editors, and the UDI Wizard configuration file as described in Review the Relationship Among UDI Wizard Pages, Wizard Page Editors, and the UDI Wizard Configuration File.
- Review the UI of the UDI Wizard Designer as described in Review the UDI Wizard Designer User Interface.
- Create a new UDI Wizard configuration file as described in Create a New UDI Wizard Configuration File.
- Open an existing UDI Wizard configuration file as described in Open an Existing UDI Wizard Configuration File.
- Save changes in the UDI Wizard Designer to a UDI Wizard configuration file as described in Save UDI Wizard Configuration Updates.

- Override the location and name of the UDI configuration file used by a task sequence as described in Override the Configuration File That the UDI Wizard Uses.
- Configure the title and banner image to be displayed in the UDI Wizard as described in Configure the UDI Wizard Title and Banner Image.
- Add a wizard page to a stage as described in Add a Wizard Page to a Stage.
- Remove a wizard page from a stage as described in Remove a Wizard Page from a Stage.
- Change the sequence of a wizard page within a stage as described in Change the Wizard Page Sequence Flow Within a Stage.
- Allow or prevent users from entering information in a control on a wizard page as described in Allow or Prevent Users from Entering Information in a Control on a Wizard Page.
- Configure the user experience for a wizard page as described in Configure the User Experience for a Wizard Page.
- Preview how the wizard pages and wizard page sequence flow the UDI Wizard as described in Preview Wizard Pages and the Wizard Page Sequence Flow.
- Add a wizard page to the page library as described in Add a Wizard Page to the Page Library.
- Remove a wizard page from the page library as described in Remove a Wizard Page from the Page Library.
- Change the sequence of a stage group or a stage within a stage group as described in Change the Sequence of a Stage Group or a Stage.
- Prepare for language pack deployment using the UDI Wizard as described in Prepare for Language Pack Deployment in UDI.
- Skip (remove) a wizard page from a stage as described in Skip a Wizard Page.

Review UDI Wizard Designer Concepts

The UDI Wizard Designer is a console in MDT that allows you to easily configure the UDI Wizard configuration file. The UDI Wizard Designer can update an existing UDI Wizard configuration file or create a new UDI Wizard configuration file.

> **Note**
>
> If you are unfamiliar with UDI, review the UDI terms and terminology in "UDI Concepts". Familiarizing yourself with these terms and terminology will help you be more successful in applying this guide to your organization.

At a high-level, the UDI Wizard Designer allows you to configure the:

- Types of wizard pages that are displayed in the UDI Wizard
- Sequence of the wizard pages as they will appear in the UDI Wizard
- Configuration settings for each wizard page

  For more detailed information about the tasks that can be performed in the UDI Wizard Designer, see Configuring UDI Wizard Behavior.

Identify UDI Deployment Process Components

The UDI deployment process is based on ZTI deployments in MDT and requires Configuration Manager. The UDI process runs as any other MDT task sequence, except that the UDI-specific task sequences run the UDI Wizard at the appropriate steps in the task sequence.

Table 143 lists the UDI deployment process components and a brief description of how they work together in a UDI deployment.

Table 143. UDI Deployment Process Components

Expand table

Component	Description
UDI Wizard	The UDI Wizard is the UI that allows customization of the deployment process based on configuration settings made in the wizard. The configuration settings are used to modify the task sequence variables and unattended installation files used in the operating system and application deployment process. The UDI Wizard is initiated by the appropriate task sequence steps in task sequences created using UDI task sequence templates. The wizard pages that are displayed in the UDI Wizard and the controls that

Component	Description
	are active for each wizard page is controlled by the UDI Wizard configuration file.
UDI Wizard Designer	The UDI Wizard Designer is used to customize the UDI Wizard configuration file. You can use the UDI Wizard Designer to: - Determine the wizard pages that are displayed in the UDI Wizard - Determine the sequence of the wizard pages that are displayed in the UDI Wizard - Configure default values for controls on the wizard pages - Enable or disable individual controls on the wizard pages
UDI Wizard configuration file	The UDI Wizard configuration file is read by the UDI Wizard to determine the wizard pages that are displayed, the sequence of the wizard pages, any default values for controls, and whether controls are enabled or disabled for user interaction. The UDI Wizard configuration file is customized using the UDI Wizard Designer. The default UDI Wizard configuration file is name UDIWizard_Config.xml and is stored in the Scripts folder in the MDT files package.
UDI task sequences	The UDI task sequences are created using UDI-related MDT task sequence templates. The UDI task sequence templates include the task sequence step to run the UDI Wizard at the appropriate time in the UDI deployment process. For more information about UDI task sequence templates, see Identify the UDI Task Sequence Templates in MDT.

Review the Relationship Among UDI Wizard Pages, Wizard Page Editors, and the UDI Wizard Configuration File

For each wizard page displayed in the UDI Wizard, there is a corresponding wizard page editor that can be used to configure that wizard page using the UDI Wizard Designer. The UDI Wizard configuration file (UDIWizard_Config.xml) is used to store the configuration settings for each wizard page. Figure 7 illustrates the relationship between UDI wizard pages, UDI wizard page editors, and the UDI Wizard configuration file.

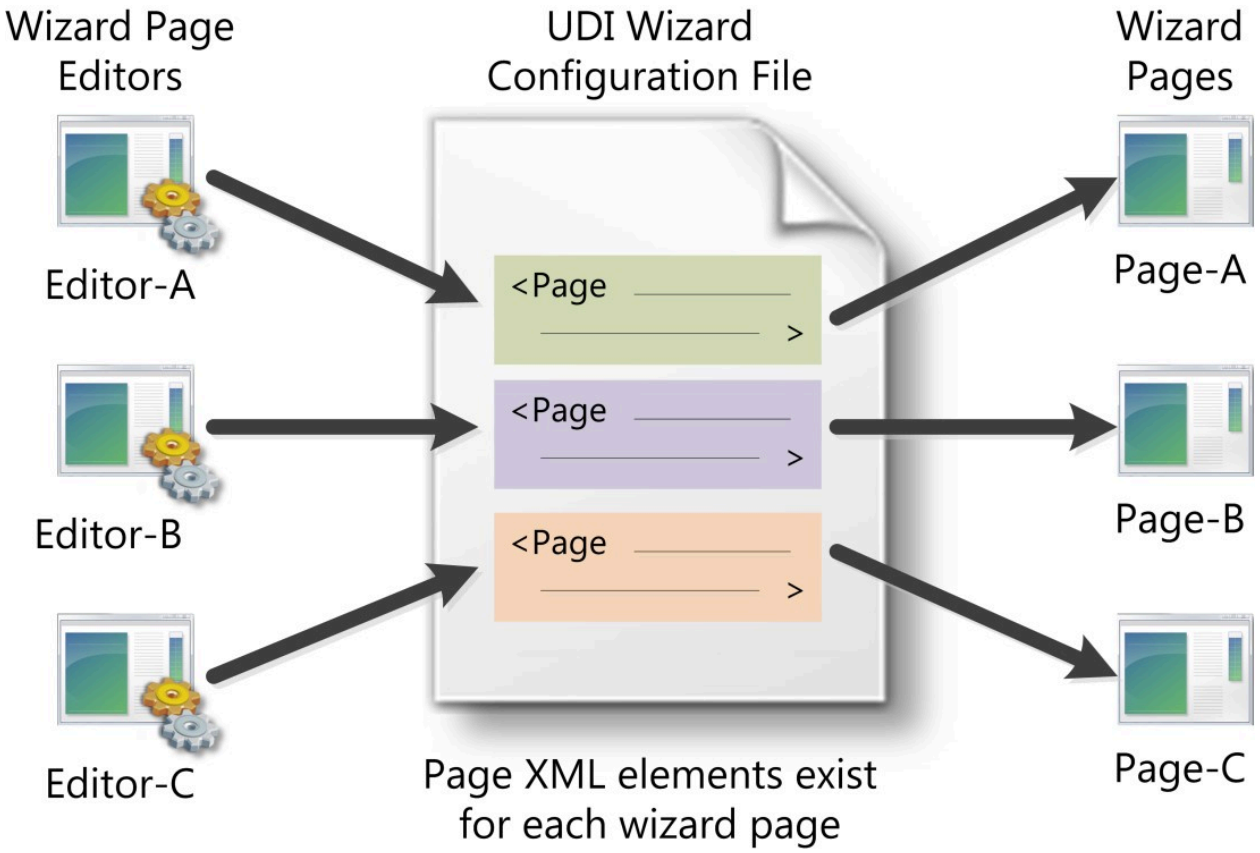


Figure 7. Relationship between UDI wizard pages, UDI wizard page editors, and the UDI Wizard configuration file

In the UDI Wizard configuration file, there is a separate **Page** XML element for each wizard page that is displayed in the UDI Wizard. When you add a wizard page using the UDI Wizard Designer a corresponding **Page** XML element is created in the UDI Wizard configuration file. Similarly, when you remove a wizard page, the corresponding **Page** XML element is removed.

Each **Page** XML element has child XML elements for each configuration setting for the wizard page. These child XML elements are also configured using the wizard page editors in the UDI Wizard Designer.

For a complete list of the **Page** XML elements, see the "UDI Wizard Configuration File Schema Reference".

> ⓘ **Note**
>
> Do not directly modify the UDI Wizard configuration file. Instead, use the appropriate wizard page editors in the UDI Wizard Designer.

Review the UDI Wizard Designer User Interface

The UDI Wizard Designer is used to customize the user experience in the UDI Wizard, including the:

- Wizard pages that are displayed in the UDI Wizard (wizard pages can be added or removed)
- Sequence of the wizard pages as they are displayed in the UDI Wizard
- Controls on each wizard page, such as:
  - Enabling a control for user interaction
  - Disabling a control for user interaction
  - Specifying a default value

  Review the UDI Wizard Designer UI by completing the following steps:

1. Review the UDI Wizard Designer high-level UI elements as described in Review the UDI Wizard Designer High-Level User Interface Elements.
2. Review the **Page Library** pane UI elements in the UDI Wizard Designer as described in Review the Page Library Pane in the UDI Wizard Designer.
3. Review the **Flow** tab in the details pane the UDI Wizard Designer as described in Review the Flow Tab in the UDI Wizard Designer.
4. Review the **Configure** tab in the details pane the UDI Wizard Designer as described in Review the Configure Tab in the UDI Wizard Designer.

Review the UDI Wizard Designer High-Level User Interface Elements

Figure 8 illustrates the UDI Wizard Designer high-level UI elements.

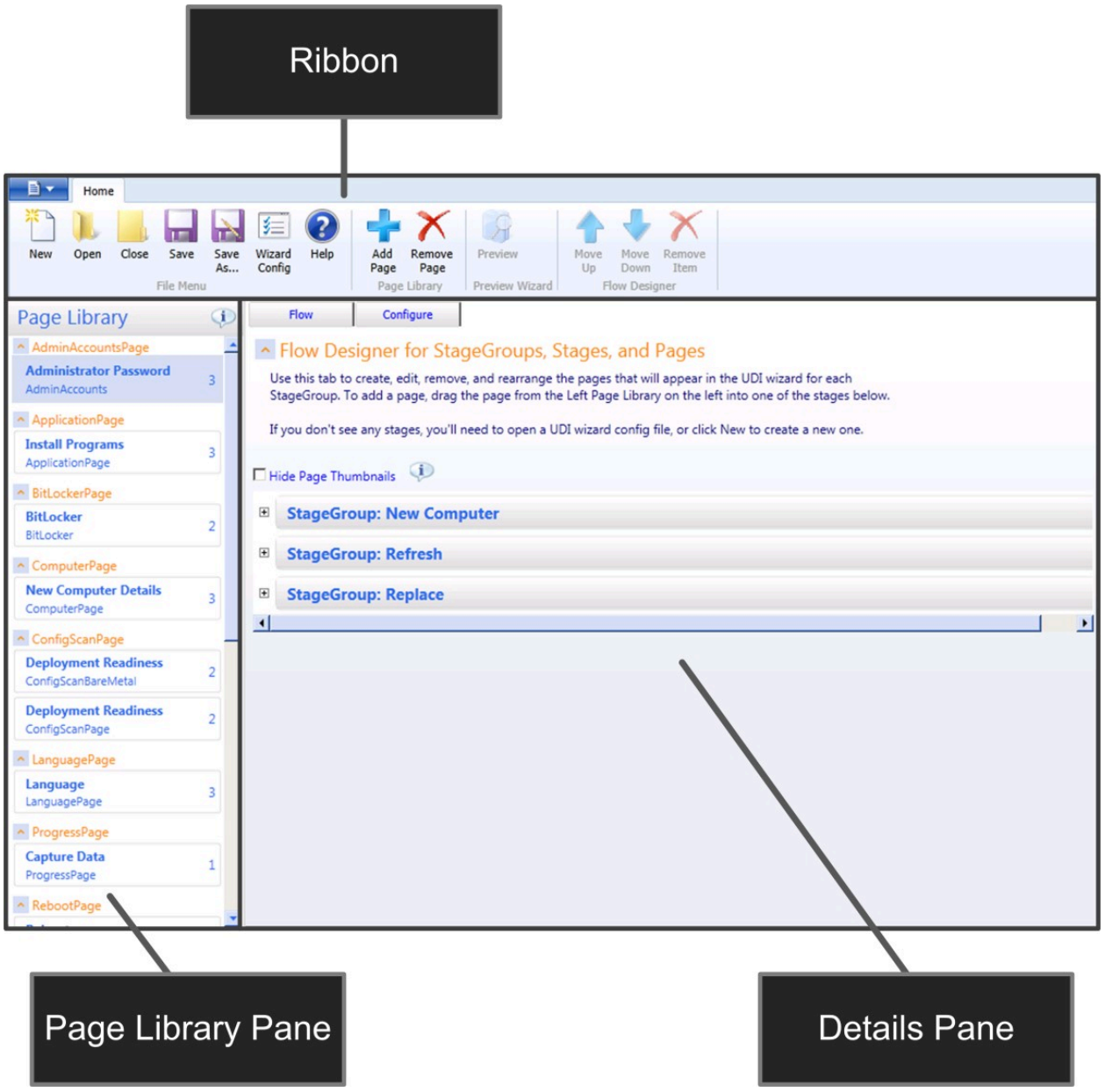


Figure 8. UDI Wizard Designer high-level UI elements

Table 144 lists the high-level UI elements illustrated in Figure 8 and provides a brief description of each element.

Table 144. UDI Wizard Designer High-Level UI Elements

Expand table

UI element	Description
Ribbon	Provides access to task-based actions that can be performed within the UDI Wizard Designer. The actions are combined into groups, such as the **File Menu** group or the **Flow Designer** group.
Page Library	Contains the wizard pages that are available for use within the UDI Wizard Designer. The number of times that each page is used within the UDI Wizard configuration file

UI element	Description
pane	is displayed on the far right portion of the wizard page entry. For example, the **Language** page is used three times while the **Scan Configuration** page is used twice.
Details pane	Provides access to the configuration details of the UDI Wizard configuration file being customized.

Review the Page Library Pane in the UDI Wizard Designer

Figure 9 illustrates the UI elements in the Page Library pane in the UDI Wizard Designer.

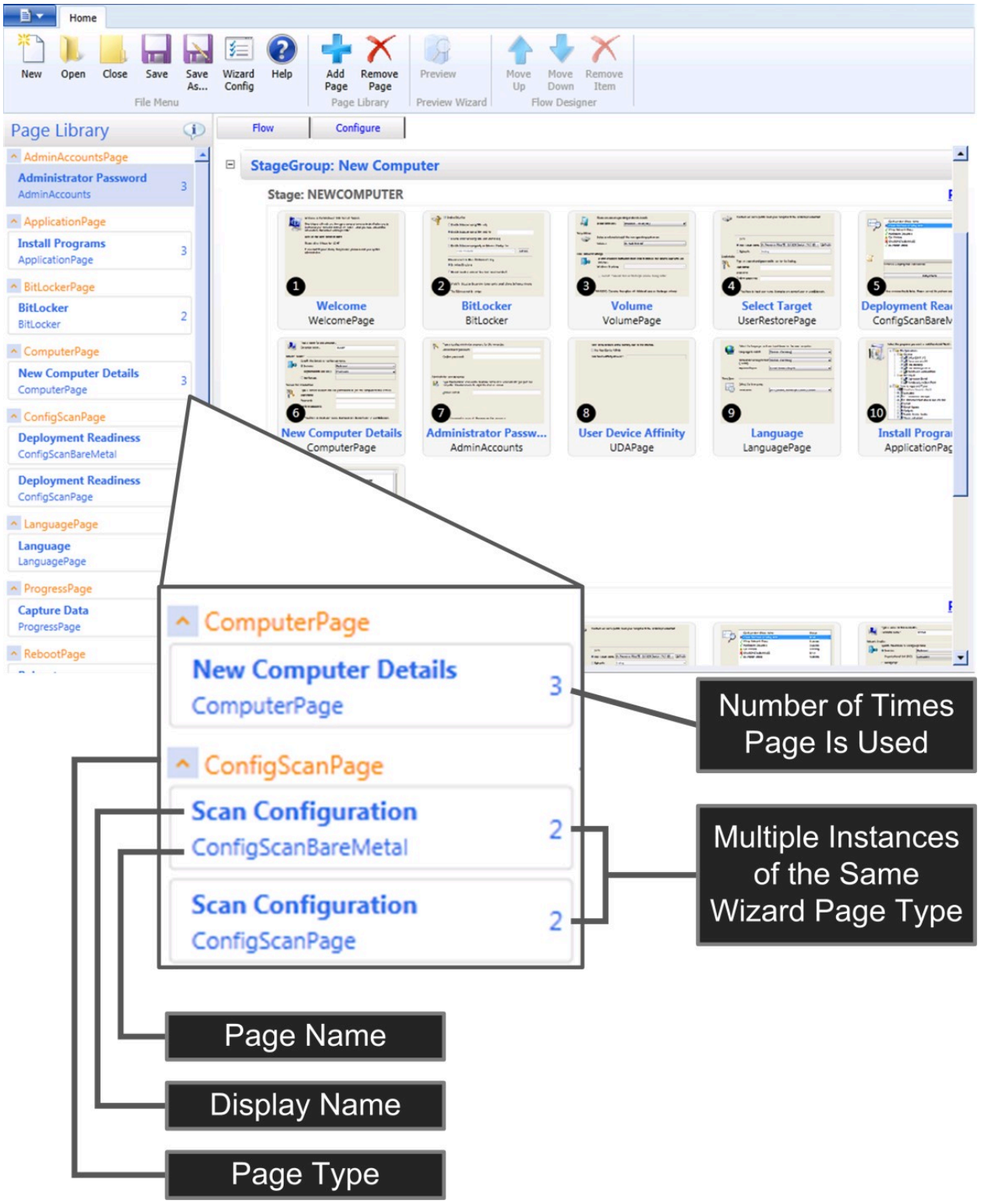


Figure 9. UI elements in the Page Library pane

Table 145 lists the UI elements illustrated in Figure 9 and provides a brief description of each element.

Table 145. UI Elements in the Page Library Pane

Expand table

UI element	Description
Page instance	Each wizard page in the Page Library is an instance of a specific wizard page type. As shown in REF _Ref307996589 \h Figure 9, there are two instances of the **ConfigScanPage** wizard page type with the page name of **ConfigScanBareMetal** and **ConfigScanPage**. Configuration settings are made to a wizard page instance in the Page Library and affect all stages to which the wizard page is added.
Page type	A *page type* defines a template that can be used to create instances of the wizard page type. The **Add New Page** dialog box has a list of the wizard page types available in the UDI Wizard Designer.
Display name	This is the user-friendly name for the instance of the wizard page type. This value does not have to be unique within the Page Library.
Page name	This is the name of the instance of the wizard page type. This value must be unique within a page library.
Number of times page is used	This element maintains a count of the number of times that a wizard page instance is used in the stages. For example, as shown in REF _Ref307996589 \h Figure 9, the **ComputerPage** wizard page instance is used in two different places within the UDI Wizard configuration file.

Review the Flow Tab in the UDI Wizard Designer

Figure 10 illustrates the UI elements in the **Flow** tab in the details pane. The **Flow** tab is used to configure the:

1. Wizard pages that will be displayed in the UDI Wizard for a specific stage within a specific stage group
2. Sequence of the wizard pages as they are displayed in the UDI Wizard

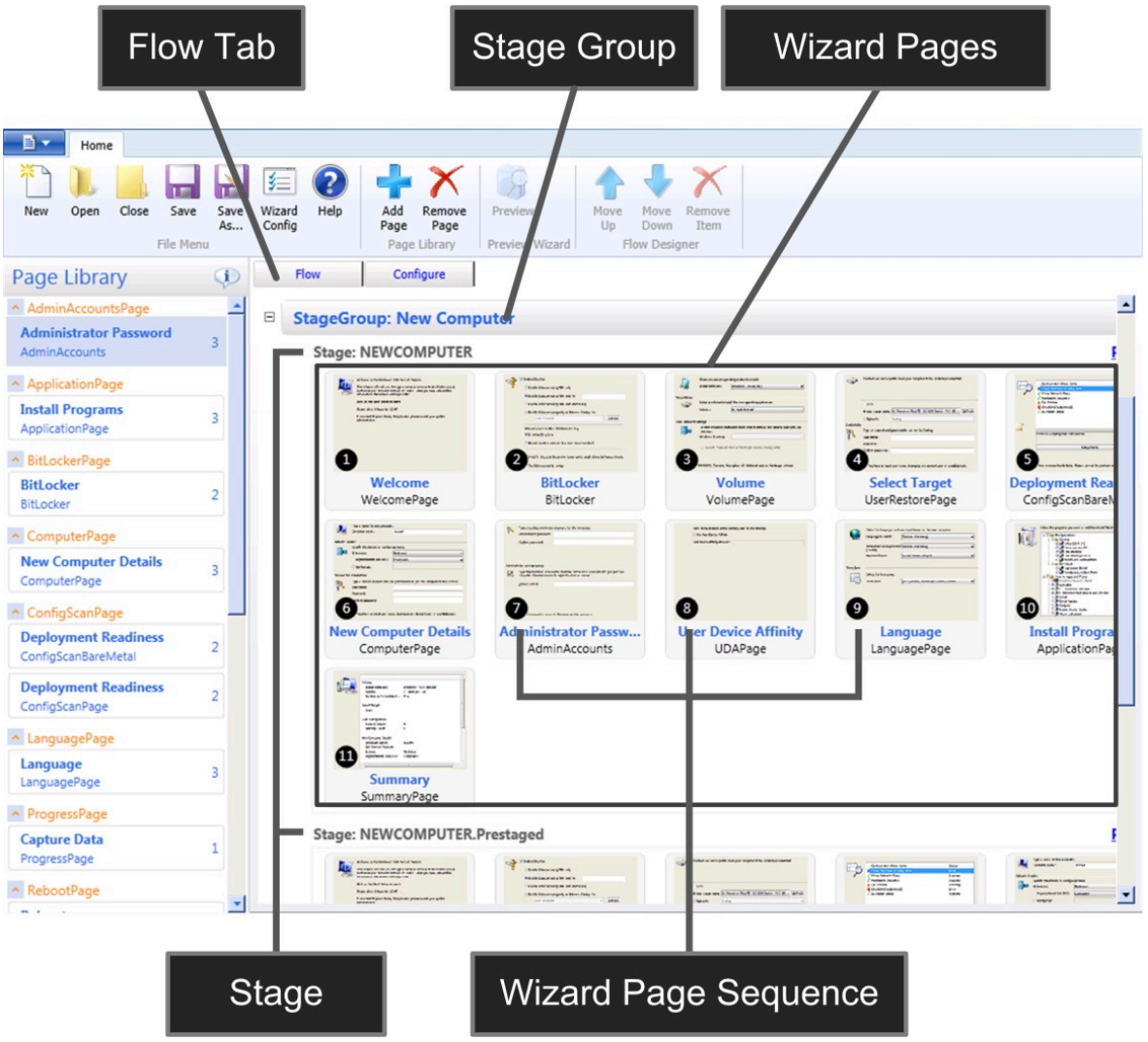


Figure 10. Flow tab in the UDI Wizard Designer

Table 146 lists the UI elements on the **Flow** tab, which is illustrated in Figure 10, and provides a brief description of each element.

Table 146. UDI Wizard Designer Flow Tab UI Elements

Expand table

UI element	Description
Stage group	Collection of one or more sets of wizard page groupings (stages) that are used by the deployment scenarios that UDI supports, including the New Computer, Refresh Computer, and Replace Computer MDT deployment scenarios. The stage groups are predefined in UDI. Adding or removing stage groups is not supported.
Stage	Collection of one or more wizard pages used at a specific time within a stage group. For the New Computer stage group, MDT includes the following stages:

UI element	Description
	- **NEWCOMPUTER**. This stage is used for new computer deployments. - **NEWCOMPUTER.Prestaged**. This stage is used for prestaged media deployments in Configuration Manager. For the Replace Computer stage group, MDT includes the following stages: 1. **REPLACE**. This stage is used for the portion of the Replace Computer stage group performed in the original operating system running on the target computer. 2. **REPLACE.WinPE**. This stage is used for the portion of the Replace Computer stage group performed in Windows PE. The stages are predefined in UDI. Adding or removing stages is not supported.
Wizard page	The wizard page that is to be displayed in the UDI Wizard for a specific stage within a specific stage group. A wizard page is based on an instance of the wizard page in the Page Library. An instance of a wizard page may appear in multiple stages and stage groups. Configuration settings for a wizard page affect the wizard page instance, not the individual pages that appear in the stages and stage groups. Create a unique instance of a wizard page in the Page Library for each set of unique configuration settings that you want to manage for a specific type of wizard page.
Wizard page sequence	The sequence in which the wizard page is displayed in the UDI Wizard for a specific stage within a specific stage group.

Review the Configure Tab in the UDI Wizard Designer

Figure 11 illustrates the UI elements in the **Configure** tab in the details pane. You use the **Configure** tab to configure the individual controls on the wizard page.

> ⓘ **Note**
>
> Any changes made to the settings on the **Configure** tab affect the instance of that wizard page in the Page Library. The result is any stage groups or stages that contain the same instance of that wizard page will also reflect the changes in the configuration settings.

Figure 11. Configure tab in the UDI Wizard Designer

Table 147 lists the UI elements on the **Flow** tab, which is illustrated in Figure 11, and provides a brief description of each element.

Table 147. UDI Wizard Designer Configure Tab UI Elements

Expand table

UI element	Description
Page screenshot	Expand to show a sample of the wizard page as displayed in the UDI Wizard.
Section	Logical grouping of one or more user controls.
Subsection	Logical grouping of one or more user controls within a section. Expand to show the user controls contained within the configuration details.

If you expand a subsection, you can see the controls within that subsection. Figure 12 illustrates the UI elements for a control beneath a subsection. A subsection may contain multiple controls.

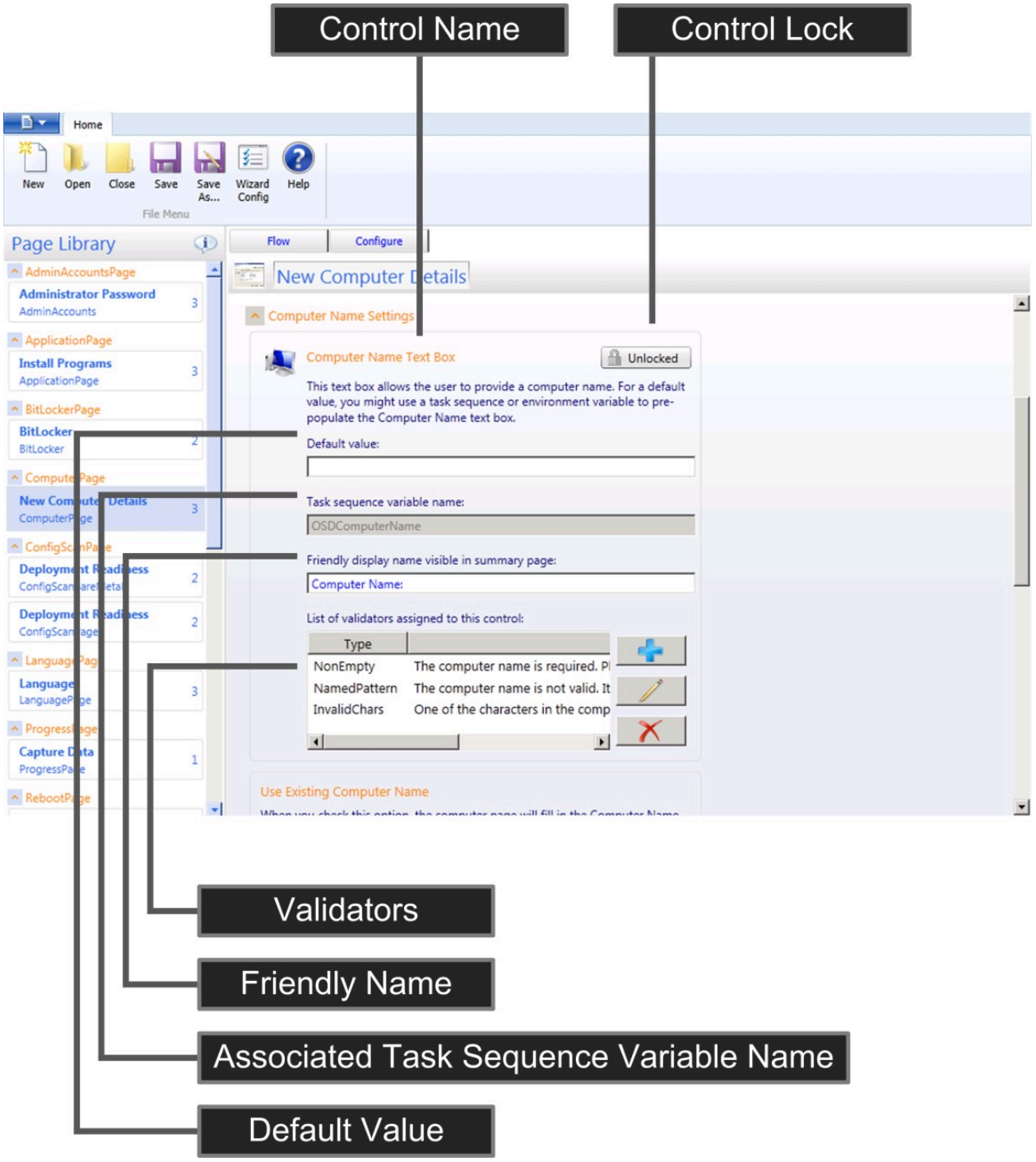


Figure 12. UI elements for a control on the Configure tab in the UDI Wizard Designer

Table 148 lists the UI elements for a control on the **Configure** tab, which is illustrated in Figure 12, and provides a brief description of each element.

> **Note**

Each control on a wizard page is unique and has different UI elements. The control illustrated in Figure 12 is provided as an example for generalized discussion.

Table 148. UI Elements for a Control on the Configure Tab

Expand table

UI element	Description
Control lock	Allows you to enable (unlocked) or disable (lock) the control to allow or prevent users from entering information in the control. The status in the button can be one of the following values: - **Unlocked**. Users are able to interact with the control in the UDI Wizard. - **Locked**. Users are unable to interact with the control in the UDI Wizard. If you disable a control, you must provide the information collected by the control either as a default value in the control or in the CustomSettings.ini file or in MDT DB.
Default value	The value that is displayed in the control by default. If the control is enabled, then the user can override the default value.
Associated task sequence variable name	The task sequence variable name associated with the control. The UDI Wizard sets the specified task sequence variable with the value provided in the UDI Wizard.
Friendly name	The name which appears on the **Summary** wizard page (if included) and shows the configuration setting collected by this control.
Validators	A list of validation checks performed on the information entered in the control. You can add or remove validation checks to this list.

For more information about configuring specific controls on specific wizard pages, see the corresponding section for that wizard page in the MDT document *Toolkit Reference*.

Create a New UDI Wizard Configuration File

The UDI Wizard displays wizard pages based on the configuration options specified in the UDIWizard_Config.xml file in the Scripts folder of the MDT files package specified in the task sequence. Create a new UDI Wizard configuration file using the UDI Wizard Designer.

Tip

Create a new MDT files package and corresponding package source for each unique configuration of the UDIWizard_Config.xml file.

To create a new UDI Wizard configuration file using the UDI Wizard Designer

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **UDI Wizard Designer**.

   The UDI Wizard Designer starts.

2. In the **UDI Wizard Designer** console, on the Ribbon, in the **File Menu** group, select **New**.

3. Make the appropriate changes in the UDI Wizard Designer console.

4. In the **UDI Wizard Designer** console, on the Ribbon, in the **File Menu**group, select **Save As**.

   The **Save As** dialog box opens.

5. In the **Save As** dialog box, go to ***folder_path*** (where *folder_path* is the fully qualified path to the Scripts folder in the MDT files package source), in **File name**, type ***file_name*** (where *file_name* is the file name for the configuration file), and then select **Save**.

   After creating the new UDI Wizard configuration file, create a new task sequence or modify an existing task sequence steps to use the appropriate MDT files package. You also need to update the distribution points with the modified MDT files package as described in Managing Distribution Points in Configuration Manager, which is the same process for UDI and ZTI deployments.

Open an Existing UDI Wizard Configuration File

The UDI Wizard displays pages based on the configuration options specified in the UDIWizard_Config.xml file in the Scripts folder of the MDT files package specified in the task sequence. Open an existing UDI Wizard configuration file using the UDI Wizard Designer.

To open an existing UDI Wizard configuration file using the UDI Wizard Designer

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **UDI Wizard Designer**.

   The UDI Wizard Designer starts.

2. In the **UDI Wizard Designer** console, on the Ribbon, in the **File Menu** group, select **Open**.

3. In the **Open** dialog box, go to ***folder_path*** (where *folder_path* is the fully qualified path to the Scripts folder in the MDT files package source), select ***file_name*** (where *file_name* is the file name for the configuration file), and then select **Open**.

Save UDI Wizard Configuration Updates

After you have updated the UDI Wizard configuration, you need to save the changes to the UDI wizard configuration file. Save the UDI Wizard configuration file in the Scripts folder of the MDT files package specified in the task sequence.

To save the UDI Wizard configuration updates using the UDI Wizard Designer

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **UDI Wizard Designer**.

   The UDI Wizard Designer starts.

2. In the **UDI Wizard Designer** console, on the Ribbon, in the **File Menu** group, select **Open**.

3. In the **Open** dialog box, go to ***folder_path*** (where *folder_path* is the fully qualified path to the Scripts folder in the MDT files package source), select ***file_name*** (where *file_name* is the file name for the configuration file), and then select **Open**.

4. Make the appropriate changes in the UDI Wizard Designer console.

5. In the **UDI Wizard Designer** console, on the Ribbon, in the **File Menu** group, select **Save**.

   The **File Save** dialog box opens, notifying you that the file Save operation is complete.

6. In the **File Save** dialog box, select **OK**.

Override the Configuration File That the UDI Wizard Uses

The UDI Wizard defaults to using the UDIWizard_Config.xml file in the Scripts folder in the MDT Files package for configuration. You can override the default configuration file that the wizard uses by modifying the **UDI Wizard**task sequence step to use the **/definition** parameter.

To override the configuration file that the UDI Wizard uses

1. In the Configuration Manager console, in the navigation pane, select **Software Library**.
2. In the Software Library, go to Overview/Operating Systems/Task Sequences.
3. In the preview pane, select ***task_sequence*** (where *task_sequence* is the name of the task sequence you want to edit).
4. On the Ribbon, on the **Home** tab, in the **Task Sequence** group, select **Edit**.

   The ***task_sequence*** *task_sequence_name TaskTask Sequence Editor** dialog box opens (where *task_sequence* is the name of the task sequence you want to edit).
5. In the ***task_sequence*** **Task Sequence Editor** dialog box (where *task_sequence* is the name of the task sequence you want to edit), in the task sequence hierarchy, go to the **State Capture** phase.
6. Beneath the **State Capture** phase, select the **UDI Wizard** task sequence step.
7. On the **Properties** tab for the **UDI Wizard** task sequence step in **Command line**, modify the text as follows (where *path* is the path to the configuration file, which is relative to the Scripts folder and *file_name* is the name of the configuration file):

   Windows Command Prompt

   ```
   cscript.exe "%DeployRoot%\Scripts\UDIWizard.wsf" /definition:
   <path\file_name>.xml.
   ```

   > **Note**
   >
   > The above text appears on one line. The line wrap seen here is the result of document formatting constraints.

8. Repeat steps 3 and 4, substituting **State Capture** with **Preinstall/New Computer Only**.
9. Repeat steps 3 and 4 for any custom task sequence steps that run UDIWizard.wsf.

10. Select **OK**.

Configure the UDI Wizard Title and Banner Image

The UDI Wizard displays a title and a banner at the top of the wizard pages. You can configure the UDI Wizard title and banner image for your organization in the UDI Wizard Designer.

To configure the UDI Wizard title and banner image using the UDI Wizard Designer

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **UDI Wizard Designer**.

   The UDI Wizard Designer starts.

2. In the **UDI Wizard Designer** console, on the Ribbon, in the **File Menu** group, select **Open**.

3. In the **Open** dialog box, go to ***folder_path*** (where *folder_path* is the fully qualified path to the Scripts folder in the MDT files package source), select ***file_name*** (where *file_name* is the file name for the configuration file), and then select **Open**.

4. On the Ribbon, on the **Home** tab, in the **File Menu** group, select **Wizard Config**.

   The **Edit Wizard Settings** dialog box appears.

5. Complete the **Edit Wizard Settings** dialog box by performing the following steps:

   a. In **Wizard Title**, type ***wizard_title*** (where *wizard_title* is the title that you want displayed at the top of the UDI Wizard).

   b. In **Banner Image**, type ***image_name*** (where *image_name* is the name of the file that contains the image that you want displayed at the top of the UDI Wizard).

   > ⓘ **Note**
   >
   > Your custom image file needs to be placed in the Tools\x64, Tools\x86, and Tools\OSDResults folders in the MDT files package source.

   c. Select **OK**.

6. Make other appropriate changes in the UDI Wizard Designer console.

7. In the **UDI Wizard Designer** console, on the Ribbon, in the **File Menu** group, select **Save**.

   The **File Save** dialog box opens, notifying you that the file Save operation is complete.

8. In the **File Save** dialog box, select **OK**.

9. Close all open windows and dialog boxes.

Add a Wizard Page to a Stage

The UDI Wizard displays a sequence of wizard pages that are used to collect the necessary information to complete the operating system and application deployment. You can configure the wizard pages and the sequence of wizard pages displayed in the UDI Wizard using the UDI Wizard Designer.

The list of available wizard pages is displayed in the Page Library pane. You can add pages from the Page Library pane by dragging the wizard page from the Page Library pane to the stage in the details pane.

To add a wizard page to a UDI stage using the UDI Wizard Designer

1. In the **UDI Wizard Designer** console, in the details pane, expand ***stage*** (where *stage* is the stage you want to customize).

2. In the Page Library pane, select ***wizard_page***, and then drag ***wizard_page*** to the details pane (where *wizard_page* is the wizard page you want to add).

   The wizard page appears in the details pane in location where you dragged it in the stage.

   > **Tip**
   >
   > Remember to save the UDI Wizard configuration file after making any changes.

Remove a Wizard Page from a Stage

The UDI Wizard displays a sequence of wizard pages that are used to collect the necessary information to complete the operating system and application deployment. You can configure the wizard pages and the sequence of wizard pages displayed in the

UDI Wizard using the UDI Wizard Designer. As a part of this process, you can remove wizard pages within a stage. Removing a wizard page from a stage does not remove a wizard page from the Page Library pane.

> **Note**
>
> If you remove a wizard page, you must provide the values for the task sequence variables that the wizard page configured. For more information, see the corresponding wizard page in the MDT document *Toolkit Reference*.

To remove a wizard page from a stage using the UDI Wizard Designer

1. In the **UDI Wizard Designer** console, in the details pane, expand ***stage*** (where *stage* is the stage you want to customize).
2. In the details pane, select ***wizard_page*** (where *wizard_page* is the wizard page you want to remove).
3. On the Ribbon, on the **Home** tab, in the **Flow Designer** group, select **Remove Item**.

   The **Delete Item Confirmation** dialog box appears.
4. In the **Delete Item Confirmation** dialog box, select **Yes**.

   In the details pane, the wizard page is removed from the stage.

   > **Tip**
   >
   > Remember to save the UDI Wizard configuration file after making any changes.

Change the Wizard Page Sequence Flow Within a Stage

The UDI Wizard displays a sequence of wizard pages used to collect the necessary information to complete the operating system and application deployment. You can configure the wizard pages and the sequence of wizard pages displayed in the UDI Wizard using the UDI Wizard Designer. As a part of this process, you can the sequence of wizard pages within a stage.

To change the wizard page sequence flow within a stage using the UDI Wizard Designer

1. In the **UDI Wizard Designer** console, in the details pane, expand ***stage*** (where *stage* is the stage you want to customize).
2. In the details pane, select ***wizard_page***, and then drag ***wizard_page*** to the location in the stage flow where you want the page to appear (where *wizard_page* is the wizard page you want to move within the stage).

   The wizard page appears in the details pane in the location to which you dragged it.

> **Tip**
>
> Remember to save the UDI Wizard configuration file after making any changes.

Allow or Prevent Users from Entering Information in a Control on a Wizard Page

Each wizard page displayed by UDI Wizard either displays information about the UDI deployment process or collects information to be used in the UDI deployment process. Then wizard pages that collect information have one or more controls used to collect the information.

By default, all controls are enabled on all wizard pages. Using the UDI Wizard Designer, you can disable individual controls on each wizard page to prevent users from entering information using those controls. The UDI Wizard designer has button that displays the following status:

1. **Unlocked**. This status indicates that the control is enabled and users can enter information using it.
2. **Locked**. This status indicates that the control is disabled and users are unable to enter information using it.

> **Note**
>
> If you disable (lock) a control, you must provide the information the control collected by configuring MDT properties in CustomSettings.ini or in the MDT DB. Otherwise, the UDI Wizard will not collect the necessary information, and the UDI deployment will fail.

To allow or prevent users from entering formation in a control on a wizard page using the UDI Wizard Designer

1. In the **UDI Wizard Designer** console, in the details pane, expand ***stage*** (where *stage* is the stage you want to customize).
2. In the details pane, select ***wizard_page*** (where *wizard_page* is the wizard page you want to remove).
3. In the details pane, on the **Configure** tab, expand ***section***, go to ***control***(where *section* is the section where the control is located and *control* is the control you want to allow or prevent users from entering information), and then select the corresponding button with one of the following status indicators:
    - **Unlocked**. Selecting the button changes the status to **Locked** and prevents users from entering information in the control.
    - **Locked**. Selecting the button changes the status to **Unlocked** and allows users to enter information in the control.

> **Tip**
>
> Remember to save the UDI Wizard configuration file after making any changes.

Configure the User Experience for a Wizard Page

Each wizard page collects unique information that helps configure the UDI deployment process. You can configure the user experience for each wizard page.

To configure the user experience for a specific wizard page using the UDI Wizard Designer

1. In the **UDI Wizard Designer** console, in the details pane, expand ***stage*** (where *stage* is the stage that contains the wizard page you want to customize).
2. In the details pane, on the **Flow** tab, select ***wizard_page*** (where *wizard_page* is the name of the wizard page you want to customize).
3. In the details pane, select the **Configure** tab.
4. In the details pane, configure the user experience based on the type of wizard page being configured.

For more information about configuring the user experience for each type of wizard page included in MDT, see the corresponding section for the wizard page in the *User-Driven Installation Developers Guide*.

> **Tip**
>
> Remember to save the UDI Wizard configuration file after making any changes.

Preview Wizard Pages and the Wizard Page Sequence Flow

After you have the appropriate wizard pages in the correct sequence for a stage, you can preview how the pages will appear in the UDI Wizard using the Preview feature in the UDI Wizard Designer. The Preview feature allows you to visualize the user experience and to make any changes to the user experience prior to performing actual deployments.

To preview the wizard pages and wizard page sequence flow for a stage using the UDI Wizard Designer

1. In the **UDI Wizard Designer** console, in the details pane, expand ***stage*** (where *stage* is the stage you want to customize).

2. On the Ribbon, on the **Home** tab, in the **Preview Wizard** group, select **Preview**.

   The UDI Wizard opens and allows you to navigate through (preview) the wizard pages. For each wizard page, you can see the controls displayed and whether the controls are enabled.

3. After reviewing the wizard pages, close the UDI Wizard.

   You can also preview the wizard pages and the wizard page sequence flow for a stage by selecting the **Preview** hyperlink on the stage within a stage.

Add a Wizard Page to the Page Library

The Page Library in the UDI Wizard Designer contains a list of the wizard pages that you can add to stages. Each wizard page in the Page Library maintains a count of the number of instances in which the wizard page is used in the current version of the UDI Wizard configuration file.

You can add a wizard page to the Page Library so that it can be added to stages.

To add a wizard page to the Page Library using the UDI Wizard Designer

1. In the **UDI Wizard Designer** console, on the Ribbon, on the **Home** tab, in the **Page Library** group, select **Add Page**.

   The **Add New Page** dialog box appears.

2. Complete the **Add New Page** dialog box by performing the following steps:

   a. In **Select the page that you want to add**, select ***page_type*** (where *page_type* is the page type that you want to add to the stage).

   b. In **Page Name**, type ***unique_name*** (where *unique_name* is a unique name for the wizard page).

   > **Tip**
   >
   > An error message appears directly above **Display Name** if the **Page Name** is not unique.

   c. Select **OK**.

   The wizard page appears in the list of wizard pages in the Page Library.

> **Tip**
>
> Remember to save the UDI Wizard configuration file after making any changes.

You can also add a wizard page by right-clicking anywhere in the stage in the details pane, and then selecting **Add Page**.

Remove a Wizard Page from the Page Library

The Page Library in the UDI Wizard Designer contains a list of the wizard pages that you can add to stages. Each wizard page in the Page Library maintains a count of the number of instances in which the wizard page is used in the current version of the UDI Wizard configuration file.

You can remove a wizard page from the Page Library so that it can no longer be added to stages.

> **Note**

You cannot remove wizard pages from the Page Library that are currently in use in any stage. Verify that the wizard page is not used in any stages by viewing the in use count in the Page Library.

To remove a wizard page from the Page Library using the UDI Wizard Designer

1. In the **UDI Wizard Designer** console, in **Page Library**, select ***wizard_page*** (where *wizard_page* is the name of the wizard page you want to remove from the page library).
2. On the Ribbon, on the **Home** tab, in the **Page Library** group, select **Remove Page**.
3. If the wizard page is:

    - In use in any stage, the **Page In Use** dialog box is displayed, notifying you that the wizard page is currently in use and cannot be removed. In the **Page In Use** dialog box, select **OK**.
    - Not in use by any stage, the **Delete Item Confirmation** dialog box is displayed, confirming that you want to remove the wizard page. In the **Delete Item Confirmation** dialog box, select **Yes**.

      The wizard page is deleted from the Page Library.

> **Tip**
>
> Remember to save the UDI Wizard configuration file after making any changes.

Change the Sequence of a Stage Group or a Stage

The details pane contains a list of the stage groups and stages that the UDI Wizard configuration file (UDIWizard_Config.xml) supports. Each stage group listed in the details pane is used in one or more of the following MDT deployment scenarios:

- New Computer
- Refresh Computer
- Replace Computer

  You can change the sequence of a stage group or the stages within in the details pane.

To change the sequence of a stage group using the UDI Wizard Designer

1. In the **UDI Wizard Designer** console, in the details pane, select ***stage_group*** (where *stage_group* is the name of the stage group for which you want to change the sequence).

2. On the Ribbon, on the **Home** tab, in the **Flow Designer** group, select one of the following:

   - **Move Up** to make the stage group appear earlier in the list of scenarios
   - **Move Down** to make the stage group appear later in the list of scenarios

     The stage group moves up or down in the list of stage group based on the option you select.

   You can also change the sequence of a stage group by right-clicking the stage group, and then selecting **Move Up** or **Move Down** based on the desired result.

> **Tip**
>
> Remember to save the UDI Wizard configuration file after making any changes.

To change the sequence of a stage within a stage group using the UDI Wizard Designer

1. In the **UDI Wizard Designer** console, in the details pane, expand ***stage_group***, and then select ***stage*** (where *stage_group* is the name of the stage group that contains the stage and *stage* is the name of the stage for which you want to change the sequence).

2. On the Ribbon, on the **Home** tab, in the **Flow Designer** group, select one of the following:

   - **Move Up** to make the stage appear earlier in the list of stages within the stage group
   - **Move Down** to make the stage appear later in the list of stages within the stage group

     The stage moves up or down in the list of stages within the stage group based on the option you select.

You can also change the sequence of a stage by right-clicking the stage, and then selecting **Move Up** or **Move Down** based on the desired result.

> **Tip**
>
> Remember to save the UDI Wizard configuration file after making any changes.

Prepare for Language Pack Deployment in UDI

One the UDI Wizard page types available in the Page Library in the UDI Wizard Designer is the **LanguagePage** wizard page type. The **LanguagePage** wizard page type allows you to select the:

- Default language
- Time and currency format (locale)
- Keyboard layout
- Time zone

  Specifically, the **LanguagePage** wizard page allows you select the default language of the target operating system. However, additional steps must be performed to support the selection and subsequent deployment of the language pack for the target operating system.

To support the deployment of multiple languages as the default language for target computers

1. Add a language pack that you want to support in UDI using the **Install Language Packs Offline** task sequence as described in Add Language Packs in Configuration Manager
2. For the **Install Language Packs Offline** task sequence step created in step 1, configure the task sequence step to run on the condition that the **UILanguage** task sequence variable is equal to the language pack added in step 1

   > **Tip**
   >
   > You configure conditions on the **Options** tab of the task sequence step. Add a **Task Sequence Variable** condition.

For example, if you add a Japanese language pack, then the condition for the corresponding **Install Language Packs Offline** task sequence step will be for the **UILanguage** task sequence variable to be equal to **JA-JP**.

For more information on the **LanguagePage** wizard page type in the UDI Wizard Designer, see the corresponding section in the *User-Driven Installation Developers Guide*.

Skip a Wizard Page

In some instances, you may want to further control and simplify the UDI Wizard experience by skipping (removing) wizard pages. Skipping a wizard page allows you to provide configuration values usually provided by the user for the wizard page. Also, skipping a wizard page is simpler and less confusing than disabling (locking) all the controls on a wizard page.

To skip a wizard page

1. Identify the variables that are written by the wizard page in a specific stage.

   To identify the variables written by a specific wizard page, see the corresponding section for the wizard page in "UDI Wizard Page Reference".

2. Configure the variables identified in the previous step in the CustomSettings.ini or the MDT DB.

3. Remove the wizard page from the stage within a stage by using the **Remove Item** action on the Ribbon in the **Flow Designer** group.

Creating Custom Wizard Pages Using the Build Your Own Page Feature

There may be instances in which you want to collect additional deployment information to be used in UDI. You must collect this additional information in the UDI Wizard using a custom wizard page. You can create custom wizard pages using the:

- **Build Your Own Page feature**. This feature allows you to create a custom wizard page for collecting deployment information without requiring you to write code or have developer skills. Use this feature if you need to collect basic information without advanced user interaction. For example, you cannot add any code or customize UI fonts using this feature.

- **UDI SDK and Visual Studio**. Use this SDK if you want to create an advanced, fully customized wizard page in Visual Studio for collecting deployment information. Although the UDI SDK allows you to create customized wizard pages, such as adding custom code or changing fonts, this method requires developer skills.

  For more information on using the UDI SDK to create custom wizard pages, see "Creating Custom UDI Wizard Pages" in the *User-Driven Installation Developers Guide*.

  Create custom wizard pages using the Build Your Own Page Feature in UDI by performing the following tasks:

- Create a new custom wizard page in a stage group as described in Create a New Custom Wizard Page.
- Add a control to a custom wizard page as described in Add a Control to a Custom Wizard Page.
- Position a control on a custom wizard page as described in Position a Control on a Custom Wizard Page.
- Change the size of a control on a custom wizard page as described in Change the Size of a Control on a Custom Wizard Page.
- Remove a control from a custom wizard page as described in Remove a Control from a Custom Wizard Page.
- Edit the properties of a control on a custom wizard page as described in Edit Custom Wizard Page Control Properties.
- Show or hide the gridlines on a custom wizard page as described in Show or Hide Custom Wizard Page Gridlines.
- Verify and test the custom wizard page after you create it as described in Verify and Test a Custom Wizard Page.

Create a New Custom Wizard Page

UDI custom wizard pages created using the Build Your Own Page feature allow you to collect deployment information in addition to the information collected on other UDI Wizard pages. You create custom wizard pages based on the **Build Your Own Page wizard page** type. After you create the custom wizard page, you can add controls to the wizard page and configure the task sequence variables that the controls set.

To create a new custom wizard page

1. Select **Start**, point to **All Programs**, point to **Microsoft Deployment Toolkit**, and then select **UDI Wizard Designer**.

   The UDI Wizard Designer starts.

2. Open the .xml file

3. On the Ribbon, on the **Home** tab, in the **Page Library** group, select **Add Page**.

   The **Add New Page** dialog box appears.

4. In the **Add New Page** dialog box, in the **Page Type** column, select **Build Your Own Page**.

5. In **Display Name**, type ***display_name*** (where *display_name* is the user-friendly name of the wizard page and appears in the wizard navigation progress pane).

6. In **Page Name**, type ***page_name*** (where *page_name* is name of the wizard page and must be unique in the Page Library), and then select **OK**.

   The new custom wizard page appears in the Page Library.

7. In the details pane, select the **Flow** tab.

8. On the **Flow** tab, expand *stage_group* (where *stage_group* is the name of the stage group to which you want to add the new custom wizard page).

   The list of wizard pages in the stage group is displayed.

9. In the Page Library, select ***display_name***. Drag the page to the appropriate place in ***stage_group*** on the **Flow** tab (where *display_name* is the user-friendly name of the wizard page and *stage_group* is the name of the stage group to which you want to add the new custom wizard page).

Add a Control to a Custom Wizard Page

After a new UDI custom wizard page is added to a stage group, you must add the appropriate controls to the new custom wizard page. You add these controls from the Build Your Own Page toolbox, which is displayed when you view the custom wizard page on the **Configure** tab in the UDI Wizard Designer.

Table 149 lists the types of controls to your custom wizard page, which is illustrated in Figure 13.

Table 149. Types of Controls in the UDI Build Your Own Page Toolbox

Expand table

Control type	Description
Checkbox	This control allows you select or clear a configuration option and behaves as a traditional UI check box. This control has a corresponding label that you can use to describe the purpose of the check box. The state of this control is True when the check box is selected and False when the check box is cleared. The state of the check box is stored in the task sequence variable configured for this control. For more information on this control, see "Checkbox Control" in the MDT document, *Toolkit Reference*.
Combobox	This control allows you to select an item from a list of items and behaves as a traditional UI drop-down list. This control allows you to add or remove items from the list and provide a corresponding value that will be set in the task sequence variable configured for this control. For more information on this control, see "Combobox Control" in the MDT document, *Toolkit Reference*.
Line	This control allows you to add a horizontal line to divide one portion of the custom wizard page from another. This control does not collect any configuration values but rather is used to visually enhance the UI. For more information on this control, see "Line Control" in the MDT document, *Toolkit Reference*.
Label	This control allows you to add descriptive, read-only text to the wizard page. This control does not collect any configuration values but rather is used to visually enhance the UI. For more information on this control, see "Label Control" in the MDT document, *Toolkit Reference*.
Radio	This control allows you to select one configuration option from a group of two or more options. As with traditional radio buttons, two or more of these controls can be grouped together, and then the user can select one of the options in the radio button group. A unique value is assigned to each option. The value assigned to the selected option control is saved in the task sequence variable configured for this control. For more information on this control, see "Radio Control" in the MDT document, *Toolkit Reference*.
Bitmap	This control allows you to add a bitmap graphic (.bmp file) to the custom wizard page. This control does not collect any configuration values but rather is used to visually enhance the UI. The path to the .bmp file is relative to the location of the UDI Wizard (OSDSetupWizard.exe). For more information on this control, see "Bitmap Control" in the MDT document, *Toolkit Reference*.
Textbox	This control allows you to enter text on the custom wizard page. The text typed into this control is saved in the task sequence variable configured for this control. For

Control type	Description
	more information on this control, see "Textbox Control" in the MDT document, *Toolkit Reference*.

You can add any combination of these controls to your custom wizard page based on the information you want to collect. In addition, you can use the **Show Gridlines** check box to show or hide gridlines that can be used to assist in visually designing the custom wizard page.

Figure 13 provides an example of a custom wizard page and the Build Your Own Page toolbox.

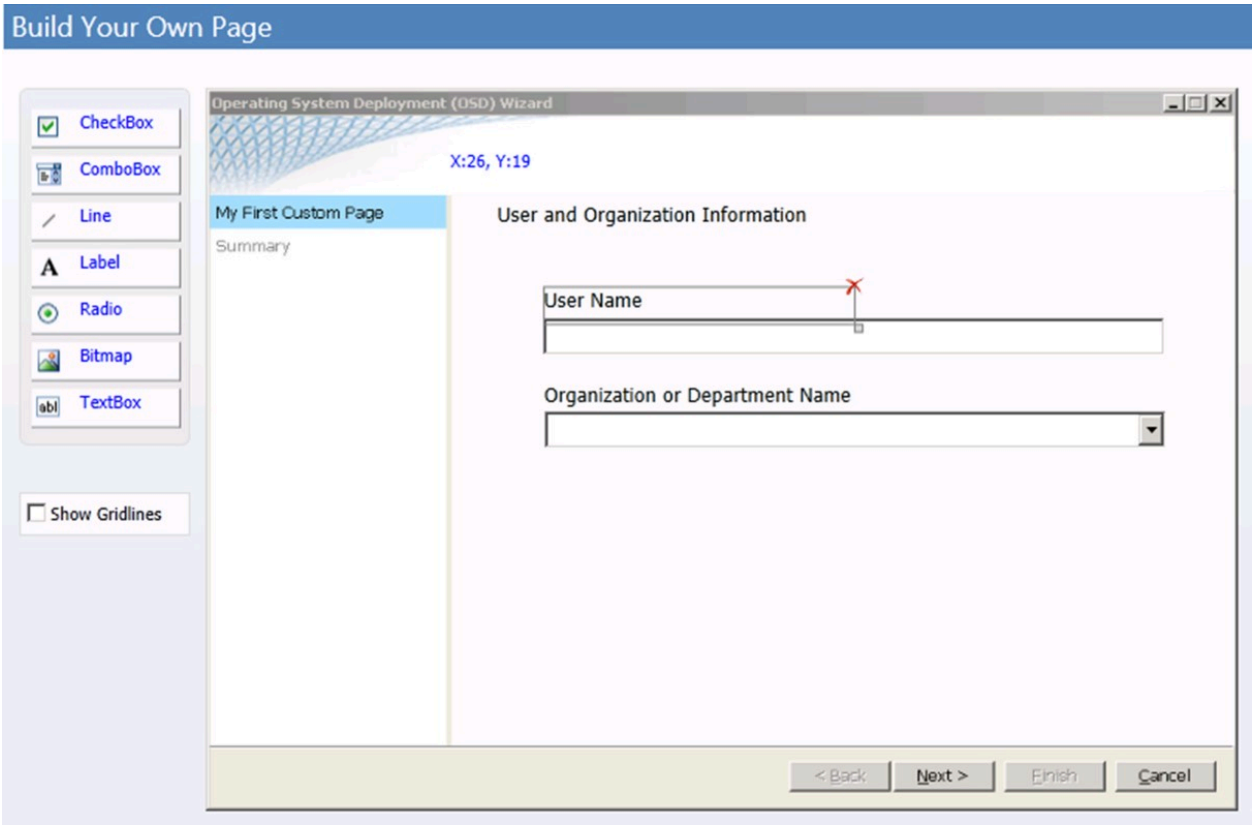


Figure 13. Example custom wizard page

To add a control to a custom wizard page

1. In the UDI Wizard Designer, in the Page Library, select ***custom_wizard_page*** (where *custom_wizard_page* is the name of the custom wizard page to which you want to add the control).

   If you have not already added a custom wizard page based on the Build Your Own Page wizard page type to the Page Library, add a custom wizard page. For more information on how to add a custom wizard page based on the **Build Your Own Page wizard page** type to the Page Library, see Create a New Custom Wizard Page.

2. In the details pane, select the **Configure** tab.

   The custom wizard page is displayed in the details pane.

3. In the Build Your Own Page toolbox, select ***toolbox_control*** (where *toolbox_control* is the type of control you want to add to the custom wizard page), and drag it to the custom wizard page.

   The control is added to the custom wizard page.

Position a Control on a Custom Wizard Page

After a control has been added to a custom wizard page, you can position the control by performing one of the following tasks:

- Position a control on a custom wizard page using drag and drop as described in Position a Control on a Custom Wizard Page Using Drag and Drop.
- Position a control on a custom wizard page using control properties as described in Position a Control on a Custom Wizard Page Using Control Properties.

Position a Control on a Custom Wizard Page Using Drag and Drop

You can position a control on a custom wizard page using drag and drop for one of the following situations:

1. Initially placing the control from the Build Your Own Page to the custom wizard page
2. Moving the control to an approximate location on the custom wizard page

   To position a control more precisely, position the control using the **X** and **Y** properties on the **Layout** properties of the control. For more information on positioning a control on a custom wizard page using control properties, see Position a Control on a Custom Wizard Page Using Control Properties.

To position a control on a custom wizard page using drag and drop

1. In the UDI Wizard Designer, in the Page Library, select ***custom_wizard_page*** (where *custom_wizard_page* is the name of the custom wizard page to which you want to position the control).

   If you have not already added a custom wizard page based on the **Build Your Own Page wizard page** type to the Page Library, add a custom wizard page. For more

information about how to add a custom wizard page based on the **Build Your Own Page wizard page** type to the Page Library, see Create a New Custom Wizard Page.

2. In the details pane, select the **Configure** tab.

   The custom wizard page is displayed in the details pane.

3. In the details pane, select ***toolbox_control*** (where *toolbox_control* is the control you want to position on the custom wizard page), and then drag it to the desired location on the custom wizard page.

   > **Tip**
   >
   > You can use the *x* and *y* coordinate locations displayed at the top of the custom wizard page to help you position the control.

Position a Control on a Custom Wizard Page Using Control Properties

Position a control on a custom wizard page when you want to control the placement of the control so that all your controls are aligned precisely. You position the control using the **X** and **Y** properties on the **Layout** properties of the control.

To position a control approximately, such as when you are doing your initial layout, do so using drag and drop. For more information on positioning a control on a custom wizard page using drag and drop, see Position a Control on a Custom Wizard Page Using Drag and Drop.

To position a control on a custom wizard page using control properties

1. In the UDI Wizard Designer, in the Page Library, select ***custom_wizard_page*** (where *custom_wizard_page* is the name of the custom wizard page to which you want to position the control).

   If you have not already added a custom wizard page based on the **Build Your Own Page wizard page** type to the Page Library, then add a custom wizard page. For more information about how to add a custom wizard page based on the **Build Your Own Page wizard page** type to the Page Library, see Create a New Custom Wizard Page.

2. In the details pane, select the **Configure** tab.

The custom wizard page is displayed in the details pane.

3. In the details pane, select ***toolbox_control*** (where *toolbox_control* is the control you wish to position on the custom wizard page), and then select the **Layout** tab.

4. On the **Layout** tab, configure the values for the properties listed in Table 150 based on the coordinates at which you want the control to be located.

Table 150. Control Position Layout Properties

Expand table

Property	Description
X	This property controls the horizontal position of the control.
Y	This property controls the vertical position of the control.

After the properties are configured, the control is positioned at the coordinates specified by these properties.

Change the Size of a Control on a Custom Wizard Page

Change the size of a control on a custom wizard page so that the contents of the control are properly displayed. You change the size of the control using the **Width** and **Height** properties on the **Layout** properties of the control.

To change the size of a control on a custom wizard page

1. In the UDI Wizard Designer, in the Page Library, select ***custom_wizard_page*** (where *custom_wizard_page* is the name of the custom wizard page to which you want to position the control).

2. In the details pane, select the **Configure** tab.

   The custom wizard page is displayed in the details pane.

3. In the details pane, select ***toolbox_control*** (where *toolbox_control* is the control you wish to change in size on the custom wizard page), and then select the **Layout** tab.

4. On the **Layout** tab, configure the values for the properties listed in Table 151 based on the size you want the control to be.

Table 151. Control Size Layout Properties

Expand table

Property	Description
Width	This property controls the width of the control. If the text or graphic displayed in the control is wider than the width of the control, the text or graphic is clipped and not displayed.
Height	This property controls the height of the control. If the text or graphic displayed in the control is higher than the height of the control, the text or graphic is clipped and not displayed.

After the properties are configured, the size of the control reflects the values in these properties.

Remove a Control from a Custom Wizard Page

Remove a control from a custom wizard page when the control is no longer needed on the custom wizard page. Once you remove a control from a custom wizard page, all **Layout** and **Settings** properties associated with the control are also removed. Once the control has been removed and the UDI Wizard configuration file has been saved, the removal cannot be undone.

> **Tip**
>
> If you want to undo removal of a control, close the UDI Wizard without saving changes.

To remove a control from a custom wizard page

1. In the UDI Wizard Designer, in the Page Library, select ***custom_wizard_page*** (where *custom_wizard_page* is the name of the custom wizard page from which you want to remove the control).

2. In the details pane, select the **Configure** tab.

   The custom wizard page is displayed in the details pane.

3. In the details pane, select ***toolbox_control*** (where *toolbox_control* is the control you want to remove from the custom wizard page), and then select the red **X** in the upper right corner of the control.

   The control is removed from the custom wizard page.

Edit Custom Wizard Page Control Properties

Each control that you place on your custom wizard page has properties. These properties are used to configure the appearance of the control and how the UDI Wizard processes the information the control collects.

The following types of properties are available for Build Your Own Page toolbox controls:

- **Layout properties**. Use these properties to configure the UI characteristics of the control. Every control has **Layout** properties, such as the **Y**, **X**, **Width**, and **Height** properties.

  For more information about the **Layout** properties for a specific control, see the corresponding section for each control in "UDI Build Your Own Page Toolbox Control Reference" in the MDT document, *Toolkit Reference*.

- **Settings properties**. Use these properties to configure the data that is initially shown in a control (default value) and where the information collected from the user is saved. Only controls that collect information have **Settings** properties, such as the **Task sequence variable name** and **Friendly display name visible in summary page** properties.

  For more information about the **Settings** properties for a specific control, see the corresponding section for each control in "UDI Build Your Own Page Toolbox Control Reference" in the MDT document, *Toolkit Reference*.

To edit custom wizard page control properties

1. In the UDI Wizard Designer, in the Page Library, select ***custom_wizard_page*** (where *custom_wizard_page* is the name of the custom wizard page on which you want to position the control).

2. In the details pane, select the **Configure** tab.

   The custom wizard page is displayed in the details pane.

3. In the details pane, select ***toolbox_control*** (where *toolbox_control* is the control you want to position on the custom wizard page).

4. Select the **Layout** tab to configure the **Layout** properties.

   For more information about the **Layout** properties for a specific control, see the corresponding sections for each control in "UDI Build Your Own Page Toolbox Control Reference" in the MDT document, *Toolkit Reference*.

5. Select the **Settings** tab to configure the **Settings** properties.

   For more information about the **Settings** properties for a specific control, see the corresponding sections for each control in "UDI Build Your Own Page Toolbox Control Reference" in the MDT document, *Toolkit Reference*.

Show or Hide Custom Wizard Page Gridlines

You can show or hide gridlines on your custom wizard pages. The gridlines help you place controls so that they are aligned properly to each other.

To show or hide custom wizard page gridlines

1. In the UDI Wizard Designer, in the Page Library, select ***custom_wizard_page*** (where *custom_wizard_page* is the name of the custom wizard page on which you want to position the control).

2. In the details pane, select the **Configure** tab.

   The custom wizard page is displayed in the details pane.

3. In the details pane, select or clear the **Show Gridlines** check box.

   The **Show Gridlines** check box determines whether the gridlines are displayed on the custom wizard page. If the**Show Gridlines** check box is:

   - Selected, then the gridlines are displayed
   - Cleared, then the gridlines are not displayed

Verify and Test a Custom Wizard Page

After you create your custom wizard page and configure the appropriate controls, verify that your custom wizard page behaves as expected. You can verify and test your custom wizard page using the preview feature in the UDI Wizard Designer.

The preview feature allows you to visualize the user experience and make any changes to the user experience prior to performing actual deployments. You can interact with your custom wizard page as though you were the user running the UDI Wizard.

For more information on how to preview wizard pages and the wizard page sequence flow, see Preview Wizard Pages and the Wizard Page Sequence Flow.

Running the UDI Wizard

The UDI Wizard is automatically initiated when you run a UDI-based task sequence. Initiate the UDI-based task sequence automatically by using Windows Deployment Services or manually by using a deployed (advertised) task sequence in the Configuration Manager Client. Each MDT deployment scenario (New Computer, Refresh Computer, or Replace Computer) uses a different process. Initiate the deployment from Windows Deployment Services or using task sequence bootable media. The deployment process prompts for any configuration settings not already specified.

The UDI Wizard displays wizard pages based on the MDT deployment scenario you selected and the configuration options you saved in UDI Wizard configuration file (UDIWizard_Config.xml) in the Scripts folder of the MDT files package. The controls that are enabled and their default values are also controlled by the configuration options you saved in the UDI Wizard configuration file.

To run the UDI Wizard

1. Initiate the task sequence created using a UDI-based task sequence template and one of the following methods:

   - Task sequence bootable media disk using the Task Sequence Media Wizard as described in Creating Task Sequence Bootable Media in Configuration Manager.

   - Windows Deployment Services to start the appropriate Windows PE images that will in turn start the UDI deployment process to the target computers as described in Preparing Windows Deployment Services for UDI Deployments.

     Windows PE starts, and then the Task Sequence Wizard starts.

2. Complete the Task Sequence Wizard by selecting the appropriate UDI-based task sequence.

   At the appropriate task sequence step, the UDI Wizard starts.

3. Complete the UDI Wizard based on the wizard pages selected and the sequence of the wizard pages.

   > **Tip**
   >
   > Ensure that you preview the user experience of the wizard pages in the UDI Wizard using the Preview feature in the UDI Wizard Designer prior to performing deployments in your production environment.

   After you complete the UDI Wizard, the deployment of the new operating system begins. When the deployment process is complete, the **OSD Results** page is displayed just prior to the first user logging on to the target computer. For more information about how to configure the **OSD Results** page, see the section, "OSDResults.exe.config File Element Values", in the MDT document *Toolkit Reference*.

Configuring MDT Deployments

Configure MDT deployments by:

- Customizing the CustomSettings.ini and Bootstrap.ini files as described in Customizing MDT Configuration Files
- Customizing the MDT properties as described in Configuring the Appropriate MDT Properties
- Applying the MDT properties to groups of computers as described in Applying MDT Properties to Groups of Computers
- Applying the MDT properties to individual computers as described in Applying MDT Properties to Individual Computers
- Configuring the MDT processing rules as described in Configuring MDT Processing Rules
- Preparing disks on the target computers as described in Preparing Disks on Target Computers
- Saving and restoring the user state migration data using USMT as described in Saving and Restoring User State Information
- Joining target computers to AD DS domains as described in Joining Target Computers to AD DS Domains

- Deploying software updates to the target computers as described in Deploying Software Updates to Target Computers
- Managing device drivers in MDT deployments as described in Managing Device Drivers
- Running Microsoft System Center 2012 Orchestrator runbooks from MDT as described in Running Orchestrator Runbooks
- Running Windows PowerShell scripts in a task sequence as described in Running Windows PowerShell Scripts During Deployment
- Applying security and compliance configuration settings using Group Policy Object Packs as described in Applying Group Policy Object Packs
- Enabling participation in Windows Customer Experience Improvement Program (CEIP) and Windows Error Reporting (WER) as described in Enabling Participation in CEIP and WER
- Configuring the task sequence steps that configure Windows roles and features on the target computer as described in Configuring Roles and Features Task Sequence Steps
- Configuring server roles for Windows Server operating systems in MDT deployments as described in Configuring Server Role Task Sequence Steps
- Copying content to the target computers for MDT deployments as described in Copying Content to the Target Computer
- Creating custom scripts that integrate with the MDT deployment processes as described in Creating Custom Scripts for MDT

Customizing MDT Configuration Files

MDT is flexible and highly customizable with the MDT configuration files. The following sections contain configuration samples that demonstrate how to customize the deployment process.

Customize the MDT configuration files by:

- Identifying the syntax of the CustomSettings.ini file as described in Identify the CustomSettings.ini File Syntax
- Identifying the sections of the CustomSettings.ini file as described in Sections in the CustomSettings.ini File

- Configuring the properties in the CustomSettings.ini file as described in Properties in the CustomSettings.ini File
- Configuring subsections in the CustomSettings.ini file as described in Subsections in the CustomSettings.ini File
- Configuring the CustomSettings.ini file to run user exit scripts using the **UserExit** directive as described in User Exit Scripts in the CustomSettings.ini File
- Configuring the basic configuration settings for the CustomSettings.ini file for LTI deployments as described in Basic CustomSettings.ini File for LTI Deployments
- Configuring the basic configuration settings for the CustomSettings.ini file for ZTI deployments in Configuration Manager as described in Basic CustomSettings.ini File for ZTI Deployments Using Configuration Manger
- Identifying the syntax of the BootStrap.ini file as described in Identify the BootStrap.ini File Syntax

Identify the CustomSettings.ini File Syntax

The syntax of the CustomSettings.ini file is similar to many .ini files. A CustomSettings.ini file includes:

- Sections
- Properties
- Settings

  Listing 1 shows a CustomSettings.ini file customized for ZTI for Configuration Manager. For more information about the CustomSettings.ini file in Listing 1, see Basic CustomSettings.ini File for ZTI Deployments Using Configuration Manger, later in this guide.

  Listing 1. CustomSettings.ini File Customized for ZTI Deployment for Configuration Manager

```ini
[Settings]
Priority=Default, MACAddress
Properties=CustomProperty

[Default]
OSInstall=Y
```

```
ScanStateArgs=/v:5 /o /c
LoadStateArgs=/v:5 /c /lac
UserDataLocation=NONE

[00:0F:20:35:DE:AC]
CustomProperty=TRUE

[00:03:FF:FE:FF:FF]
CustomProperty=FALSE
```

Sections in the CustomSettings.ini File

Sections are identified by brackets (`[]`) that surround the section name (for example, `[Settings]`). In Listing 1, the sections include `[Settings]`, `[Default]`, `[00:0F:20:35:DE:AC]`, and `[00:03:FF:FE:FF:FF]`.

The sections in the CustomSettings.ini file include the:

- Required sections as described in Required Sections
- Optional sections as described in Optional Sections

Required Sections

Only the `[Settings]` section is required. All other sections are optional. The MDT scripts require the `[Settings]` section in CustomSettings.ini to locate the reserved properties (**Priority** and **Properties**).

Optional Sections

You use the optional sections in the CustomSettings.ini file to assign a group of configuration settings to:

- **A group of computers**. In Listing 1, the configuration settings in the `[Default]` section are applied to more than one computer. For more information, see Applying MDT Properties to Groups of Computers, later in this guide.
- **An individual computer**. In Listing 1, the configuration settings in the `[00:0F:20:35:DE:AC]` and `[00:03:FF:FE:FF:FF]` sections are applied to the corresponding computer (in this case, identified by the media access control [MAC] address of the target computer). For more information, see Applying MDT Properties to Individual Computers, later in this guide.

Properties in the CustomSettings.ini File

Properties are variables to which values must be assigned. Properties are followed by an equal sign (=). The scripts scan the CustomSettings.ini file to locate the properties.

The types of properties that you can use in deploying target computers include properties that are:

- **Automatically declared in ZTIGather.wsf**. These predefined properties are declared in the ZTIGather.wsf code and are documented in the MDT document *Toolkit Reference*. In addition, the ZTIGather.wsf file automatically sets the values for these properties. These properties are not configured in CustomSettings.ini and should be treated as read only.

- **Declared in the ZTIGather.xml file**. These predefined properties are listed in the ZTIGather.xml file and are documented in the MDT document *Toolkit Reference*. The ZTIGather.wsf file retrieves these properties by scanning the ZTIGather.xml file. Divide the properties in this file into properties that:

  - **ZTIGather.wsf automatically assigns values to**. ZTIGather.wsf automatically sets the values for these properties, which must be treated as read only.

  - **Must be assigned values in CustomSettings.ini**. Ensure that the value for any property to be used is set in CustomSettings.ini and is considered modifiable.

- **Declared in the Properties property**. These are custom properties that can be declared, and they are in addition to the properties automatically declared in ZTIGather.wsf and in ZTIGather.xml.

  The way you use properties for ZTI and LTI are identical. However, some properties are unique to ZTI or LTI deployment. Like ZTI deployments, LTI deployments also have unique properties. Most of the LTI-specific properties relate to the Deployment Wizard (such as **SkipAdministratorPassword**, **SkipCapture**, or **SkipUserData**). Although these properties use the same syntax as other properties, the reserved properties perform specific functions in the deployment processing rules.

> ⓘ **Note**
>
> Property values must be specified in upper case so that the deployment scripts can properly identify them—for example, YES, TRUE, or FALSE. This is true for property values specified in the CustomSettings.ini file, BootStrap.ini file, and MDT DB.

Configure the CustomSettings.ini file by:

- Configuring the **Priority** reserved property as described in Priority Reserved Property
- Configuring the **Properties** reserved property as described in Properties Reserved Property
- Configuring the values for properties as described in Values in the CustomSettings.ini File

Priority Reserved Property

The **Priority** reserved property determines the sequence and section in which you can find configuration values. Each section is searched in the order specified. When a property value is found, the remaining sections are not used for that property. In Listing 1, the `[Default]` section is parsed first, and then the section that corresponds to the MAC address of the target computer (in this case, `[00:0F:20:35:DE:AC]` or `[00:03:FF:FE:FF:FF]`).

Table 152 lists the types of sections that you can reference in the **Priority** property.

Table 152. Section Types for the Priority Property

Expand table

Type	You can base sections on
MDT properties	Any property known MDT. For example, specifying the **HostName** property causes MDT to scan for a section with the target computer host name. Other properties, like **MACAddress**, can result in multiple section names being checked (because a computer can have multiple MAC addresses).
Literal section name	A literal name that you specify in the **Priority** property. For example, if `MySection` is included in the **Priority** property, MDT would search for properties not previously found in the `[MySection]` section.
Indirect reference	A literal name that references a section, which in turn references other sections. For example, if the **DefaultGateway** property is included in the **Priority** property, MDT would search for the `[DefaultGateway]` section. If the `[DefaultGateway]` section references other sections (based on the IP address of the default gateway), this is an example of an indirect reference. For an example of indirect reference using the **DefaultGateway** property, see "Example: Computer Groupings Selected by Woodgrove Bank" in Select the Method for Grouping Computers.

Properties Reserved Property

The **Properties** reserved property (shown in Listing 1) defines any custom, user-defined properties to be used in the deployment. These user-defined properties are located by ZTIGather.wsf script in the CustomSettings.ini file (or configuration database). These properties are in addition to the predefined properties in MDT.

In Listing 1, **CustomProperty** is a user-defined property, and **ScanStateArgs** is a predefined property. For a list of the predefined properties in MDT, see the section, "Properties", in the MDT document *Toolkit Reference*.

You can also define custom properties to which you can assign multiple values by adding numerical suffixes, such as **ListProperty001**, **ListProperty002**, and so on. You create these types of custom properties by adding "`(*)`" to the end of the property name. For example, **ListProperty(*)** defines the custom property as a list of property values instead of a single-valued property. Consider the following excerpt from a CustomSettings.ini file in which **ListProperty(*)** is defined:

ini

```ini
[Settings]
Priority=Default
Property=CustomProperty, ListProperty(*)

[Default]
CustomProperty=TRUE
ListProperty001=New York City
ListProperty002=Chicago
```

Values in the CustomSettings.ini File

Values are the configuration settings assigned to the properties. Values are preceded by an equal sign (=). The scripts scan the CustomSettings.ini file to locate the values. In Listing 1, the value assigned to the **LoadStateArgs** property is:

ini

```ini
/v:5 /c /lac
```

> ⓘ **Note**
>
> the CustomSettings.ini file is different from traditional INI files in that you do not place quotation marks around values, even if the value contains spaces.

Subsections in the CustomSettings.ini File

You can create subsections in the Customsettings.ini file based on the value of a property using the **Subsection** directive. The value of the **Subsection** directive can be used to dynamically reference subsections that can be used to group configuration settings.

Listing 2 illustrates an excerpt of a CustomSettings.ini file that uses the **Subsection** directive to dynamically reference subsections based on the computer model, which is specified in the **Model** property.

Listing 2. Using the Subsection Directive to Dynamically Reference Subsections in the CustomSettings.ini File

ini

```ini
[Settings]
Priority=Make, Default

[Default]

[Contoso Computer Corporation]
Subsection=Contoso-%Model%

[Contoso-MDT 6600]
Packages001=XXX00009:Program9
Packages002=XXX0000A:Program10

[Contoso-MDT 2431]
Packages001=XXX00003:Program2
Packages002=XXX00003:Program4
```

In Listing 2, the **Priority** line contains the **Make** property, which is used to references subsections based on the value of the **Make** property. The "Contoso Computer Corporation" subsection is referenced when the value of the **Make** property is equal to "Contoso Computer Corporation".

The "Contoso Computer Corporation" subsection contains a **Subsection** line that references other subsections based on the value of the **Model** property. In this example, the "Contoso-MDT 6600" and "Contoso-MDT 2431" sections will be processed by MDT depending on the value of the **Model** property.

User Exit Scripts in the CustomSettings.ini File

A user exit script is effectively a function library that can be called during the processing of the CustomSettings.ini file using the **UserExit** directive. A user exit script contains one or more functions that can be called during the process of the CustomSettings.ini file.

A user exit script is called by specifying the **UserExit** directive and assigning the property name of the script to be called—for example, **UserExit=TrimAssetTag.vbs**. A function within the user exit script is called by specifying the name of a function enclosed in the # characters. For example, if the user exit script contains a function called **TrimAssetTag()**, it would be called by specifying **#TrimAssetTag()#**.

Parameters can be passed to the function in the user exit script in the usual way by specifying the parameter while calling the function. For example, to pass the variable **%ASSETTAG%** to the function **TrimAssetTag()**, the function would be called by specifying **#TrimAssetTag("%ASSETTAG%")#**.

The value returned by the function can be assigned to a variable by assigning the function to that variable. For example, to take the asset tag of a computer and trim it using the function **TrimAssetTag()**, and to then reassign the trimmed asset tag to the variable **AssetTag**, the CustomSettings.ini file would read **AssetTag=#TrimAssetTag("%ASSETTAG%")#**.

An example of how this could be used is to determine the task sequence to be run based on a rule that sets the **TaskSequenceID** property. Listing 3 is an example user exit script that determines the task sequence to be run based on the amount of available RAM. This script also uses the **ZTIUtility** logging class.

Listing 3. Example User Exit Script

VB

```
Function UserExit(sType, sWhen, sDetail, bSkip)
  UserExit = Success
End Function

Function SetTaskSequence(vMemory)

  oLogging.CreateEntry "UserExit - Determining Task " & _
    "Sequence to run based on available RAM", LogTypeInfo

  If vMemory <= 2048 Then
    SetTaskSequence = "Win7_X86"
    oLogging.CreateEntry "UserExit - Available RAM: " & _
      vMemory & ". Selecting Win7_X86 TS.", LogTypeInfo
  Else
    SetTaskSequence = "Win8_X86"
    oLogging.CreateEntry "UserExit - Available RAM: " & _
      vMemory & ". Selecting Win8_X86 TS.", LogTypeInfo
  End If
```

```
End Function
```

The user exit script should be placed in the Scripts folder on the deployment share (for example, D:\Production Deployment Share\Scripts).

To create the user exit script

1. Create and test the custom script to be used.
2. Locate the MDT Scripts folder (for example, D:\Production Deployment Share\Scripts).
3. Copy the custom script to the Scripts folder.

   With the user exit script added to the deployment share (in this case, Z-*RAMTest.wsf*), it must then be referenced in the CustomSettings.ini file for the deployment share so it is called during deployment.

To call the user exit script from CustomSettings.ini

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share* (where *deployment_share* is the name of the deployment share to configure).
3. In the Actions pane, select **Properties**.
4. Select the **Rules** tab to display the CustomSettings.ini file.
5. Add sections to UserExit.vbs to call the required functionality using the principles described in the previous section. An example CustomSetting.ini file is shown in Listing 4.
6. Select **OK** to submit the changes.
7. In the details pane, select *deployment_share* (where *deployment_share* is the name of the deployment share to configure).
8. In the Actions pane, select **Update Deployment Share**.

   The Update Deployment Share Wizard starts.

9. On the **Options** page, select **Optimize the boot image updating process**, and then select **Next**.
10. On the **Summary** page, verify the details are correct, and then select **Next**.
11. On the **Confirmation** page, select **Finish**.

    Another common use for the user exit script is to dynamically set the computer name from known MDT properties such as **SerialNumber**, **Model**, or **Product**.

    Listing 4. Example CustomSettings.ini for Calling the User Exit Script

ini

```
[Settings]
Priority=Default

[Default]
OSInstall=Y
TaskSequenceID=#SetTaskSequence("%MEMORY%")#
UserExit=Z-RAMTest.vbs

UserDataLocation=NONE
SkipCapture=YES
SkipAdminPassword=NO
SkipProductKey=YES
```

Basic CustomSettings.ini File for LTI Deployments

For LTI deployments, the Deployment Workbench uses a template version of the CustomSettings.ini file (stored in *installation_folder*\Templates, where *installation_folder* is the folder in which MDT is installed) as a basis for a customized version of CustomSettings.ini. The template version of the CustomSettings.ini file is illustrated in Listing 5. The template version in Listing 5 does not contain sufficient settings to successfully deploy Windows to a target computer. However, the file will be further customized using the Deployment Workbench.

Listing 5. Unmodified CustomSettings.ini File in the Templates Folder

ini

```
[Settings]
Priority=Default
Properties=MyCustomProperty

[Default]
```

```
OSInstall=Y
ScanStateArgs=/v:5 /o /c
LoadStateArgs=/v:5 /c /lac
```

The New Deployment Share Wizard in the Deployment Workbench modifies this template of the CustomSettings.ini file based on the responses provided. Listing 6 shows the customized version of the CustomSettings.ini file after completing the New Deployment Share Wizard.

Listing 6. Customized CustomSettings.ini File Modified by the Deployment Workbench

ini

```
[Settings]
Priority=Default
Properties=MyCustomProperty

[Default]
OSInstall=Y
ScanStateArgs=/v:5 /o /c
LoadStateArgs=/v:5 /c /lac
UserDataLocation=NONE
SkipCapture=NO
SkipAdminPassword=YES
SkipProductKey=YES
```

The CustomSettings.ini file in Listing 6 contains the property values for all the target computers to be migrated using this version of the file. This version of the file contains no values unique to a specific target computer. For LTI, the target computer-specific configuration values are manually provided during the installation process.

Table 153 explains the properties and corresponding values used in Listing 6.

Table 153. Explanation of CustomSettings.ini Properties in Listing 6

Expand table

Line in CustomSettings.ini	Purpose
`[Settings]`	Indicates the start of the `[Settings]` section.
`Priority=Default`	Establishes the sequence in which the process parses subsections to locate values for the variables. In this example, the `[Default]` section is the only subsection that is parsed for variables.

Line in CustomSettings.ini	Purpose
`Properties=MyCustomProperty`	Indicates any additional properties to locate. The properties listed here are in addition to the properties listed in ZTIGather.xml. ZTIGather.wsf parses ZTIGather.xml to obtain a list of the properties.
`[Default]`	Indicates the start of the `[Default]` section.
`OSInstall=Y`	Indicates that the computer is supposed to perform an operating system deployment.
`ScanStateArgs=/v:5 /o /c`	Parameters passed to the Scanstate.exe tool in the USMT. These parameters are passed to Scanstate.exe during state capture.
`LoadStateArgs=/v:5 /c /lac`	Parameters passed to the Loadstate.exe tool in the USMT. These parameters are passed to Loadstate.exe during state restore.
`UserDataLocation=NONE`	Indicates where the user state migration data should be saved. The value `NONE` indicates that the user state migration data should not be saved. Indicates where the user state migration data should be saved. The value `NONE` indicates that the user state migration data should not be saved.
`SkipCapture=YES`	Indicates whether the **Specify whether to prompt for image capture** page in the Deployment Wizard is displayed. If the property is set to YES, the wizard page is skipped and not displayed. Indicates whether the **Specify whether to prompt for image capture** page in the Deployment Wizard is displayed. If the property is set to YES, the wizard page is skipped and not displayed.
`SkipAdminPassword=YES`	Indicates whether the **Allow user to set Administrator Password** page in the Deployment Wizard is displayed. If the property is set to YES, the wizard page is skipped and not displayed.
`SkipProductKey=YES`	Indicates whether the **Allow user to specify a product key** page in the Deployment Wizard is displayed. If the property is set to YES, the wizard page is skipped and not displayed.

For more information on the individual properties, see the corresponding reference section in the MDT document *Toolkit Reference*.

Basic CustomSettings.ini File for ZTI Deployments Using Configuration Manger

For ZTI deployments using Configuration Manager, the Deployment Workbench uses a template version of the CustomSettings.ini file (stored in *installation_folder*\Templates,

where *installation_folder* is the folder in which MDT is installed) as a basis for a customized version of CustomSettings.ini. The template version of the CustomSettings.ini file is illustrated in Listing 7. The template version in Listing 7 does not contain sufficient settings to successfully deploy Windows to a target computer. However, the file will be further customized using the Deployment Workbench.

Listing 7. Unmodified CustomSettings.ini File in the Templates Folder

ini

```
[Settings]
Priority=Default
Properties=MyCustomProperty

[Default]
OSInstall=Y
ScanStateArgs=/v:5 /o /c
LoadStateArgs=/v:5 /c /lac
```

The New Deployment Share Wizard in the Deployment Workbench modifies this template version of the CustomSettings.ini file based on the responses provided. Listing 8 shows the customized version of the CustomSettings.ini file after completing the New Deployment Share Wizard.

Listing 8. Customized CustomSettings.ini File Modified by the Deployment Workbench

ini

```
[Settings]
Priority=Default
Properties=MyCustomProperty

[Default]
OSInstall=Y
ScanStateArgs=/v:5 /o /c
LoadStateArgs=/v:5 /c /lac
UserDataLocation=NONE
```

The CustomSettings.ini file shown in Listing 8 contains the property values for all of the target computers to be deployed using this version of the file. This version of the file contains no values that are unique to a specific target computer.

For ZTI using Configuration Manager, the Create MDT Task Sequence Wizard copies an unmodified version of the CustomSettings.ini template with no modifications to the file. Modify the version of the template in the package source folder that you specified in

Package source to be created on the **MDT Package** wizard page in the Create MDT Task Sequence Wizard as described in Create ZTI Task Sequences Using the Create MDT Task Sequence Wizard in Configuration Manager.

Modify this version of the CustomSettings.ini file to include the target computer-specific configuration values. After modifying the file, update the distribution points for the Microsoft Deployment Files package so that the changes are available to the task sequences. Listing 9 shows a modified version of the CustomSettings.ini file, which includes target computer-specific settings.

Listing 9. Customized CustomSettings.ini File with Target Computer Settings

ini

```ini
[Settings]
Priority=Default, MACAddress
Properties=MyCustomProperty

[Default]
OSInstall=Y
ScanStateArgs=/v:5 /o /c
LoadStateArgs=/v:5 /c /lac
UserDataLocation=NONE

[00:0F:20:35:DE:AC]
MyCustomProperty=TRUE

[00:03:FF:FE:FF:FF]
MyCustomProperty=FALSE
```

Table 154 explains the properties and corresponding values used in Listing 9.

Table 154. Explanation of CustomSettings.ini Properties in Listing 9

Expand table

Line in CustomSettings.ini	Purpose
`[Settings] [Settings]`	Indicates the start of the `[Settings]` section.
`Priority=Default, MACAddress`	Establishes the sequence in which the process parses subsections to locate values for the variables. In this example, the `[Default]` section is parsed first, and then the section that corresponds to the MAC address of the target computer (`MACAddress`) is parsed. The sections for the target computers (`[00:0F:20:35:DE:AC]` and `[00:03:FF:FE:FF:FF]`) contain computer-specific settings.

Line in CustomSettings.ini	Purpose
	Establishes the sequence in which the process parses subsections to locate values for the variables. In this example, the `[Default]` section is parsed first, and then the section that corresponds to the MAC address of the target computer (`MACAddress`) is parsed. The sections for the target computers (`[00:0F:20:35:DE:AC]` and `[00:03:FF:FE:FF:FF]`) contain computer-specific settings.
`Properties=MyCustomProperty`	Indicates any additional properties to locate. The properties listed here are in addition to the properties listed in ZTIGather.xml. ZTIGather.wsf parses ZTIGather.xml to obtain a list of the properties.
`[Default]`	Indicates the start of the `[Default]` section.
`OSInstall=Y`	Indicates whether the target computer is authorized to have the operating system installed.
`ScanStateArgs=/v:5 /o /c`	Parameters passed to the Scanstate.exe tool in the USMT. These parameters are passed to Scanstate.exe during the State Capture Phase.
`LoadStateArgs=/v:5 /c /lac`	Parameters passed to the Loadstate.exe tool in the USMT. These parameters are passed to Loadstate.exe during state restore.
`UserDataLocation=NONE`	Indicates where the user state migration data should be saved. The value `NONE` indicates that the user state migration data should not be saved.
`[00:0F:20:35:DE:AC]`	Section that contains all the properties and settings specific to the target computer with the matching MAC address. In this sample, the target computer has a MAC address of `[00:0F:20:35:DE:AC]`.
`[00:03:FF:FE:FF:FF]`	Section that contains all the properties and settings specific to the target computer with the matching MAC address. In this sample, the target computer has a MAC address of `[00:03:FF:FE:FF:FF]`.

For more information on the individual properties, see the corresponding reference section in the MDT document *Toolkit Reference*.

Identify the BootStrap.ini File Syntax

In LTI deployments, use the BootStrap.ini file to specify property settings before accessing the CustomSettings.ini file. Use the BootStrap.ini file to provide distribution point information, logon credentials, and Windows PE keyboard locale settings. The properties configured in BootStrap.ini help the MDT scripts locate the appropriate MDT distribution share.

The syntax of the BootStrap.ini file is identical to the CustomSettings.ini file. The BootStrap.ini file contains a subset of the properties used in CustomSettings.ini as follows:

- `DeployRoot`

  > ⓘ **Note**
  >
  > Ensure that the **DeployRoot** property is specified in the BootStrap.ini file, as the LTI deployment process will not be able to locate the CustomSettings.ini file in the deployment share. If not configured correctly, errors and warnings are displayed when running the Deployment Wizard.

- `SkipBDDWelcome`
- `UserDomain`
- `UserID`
- `UserPassword`
- `KeyboardLocale`

  For more information about each of these properties, see the corresponding section in the MDT document *Toolkit Reference*.

  The Deployment Workbench creates the BootStrap.ini file when you create a deployment share. After the initial creation, make all further customizations manually.

  Listing 10 shows the BootStrap.ini file that the Deployment Workbench creates automatically for a deployment share. You may need to customize the BootStrap.ini file in Listing 10.

  Listing 10. BootStrap.ini File As Created by the Deployment Workbench for Deployment Shares

```ini
[Settings]
Priority=Default

[Default]
DeployRoot=\\NYC-MDT-01\Distribution$
```

Configuring the Appropriate MDT Properties

MDT uses wizards to create and manage configuration files. For more information about the standard MDT configuration files, CustomSettings.ini and BootStrap.ini, see Customizing MDT Configuration Files. However, you can customize configuration files to meet the needs of your organization.

Before configuring the deployment process, select the properties to reference from the predefined or user-defined properties. The properties selected must include all the configuration settings to be supplied during the deployment process.

For ZTI deployments using Configuration Manager, provide all configuration settings required to deploy the target operating system. For LTI deployments, provide a subset of the configuration settings that are provided automatically; the remainder of the settings can be provided manually during the deployment process.

The MDT process occurs in the phases defined in the TS.xml file. The Task Sequencer parses the TS.xml file to identify the appropriate sequence for performing the deployment process. The phases defined in the TS.xml file include:

- **Validate Phase**. Performs validation checks to make sure that the operating system installation can proceed; specifically blocks installation on server operating systems.
- **State Capture Phase**. Gathers information from the configuration file, databases, and the local machine to determine how the image installation process should proceed, including whether there is enough space to do a local USMT state backup. The scripts also invoke the USMT Scanstate.exe command as appropriate.
- **Preinstall Phase**. Confirms that the necessary information has been gathered in the State Capture Phase for the Refresh Computer scenario. In the New Computer and Replace Computer scenarios, the script gathers the necessary information in this phase, because these scenarios do not perform the State Capture Phase. Also, a backup of the computer can be optionally performed for the Refresh Computer scenario.
- **Install Phase**. Installs the target operating system on the target computers.
- **Post Install Phase**. Updates the Unattend.xml with information gathered in the previous custom actions based on the operating system being deployed.
- **State Restore Phase**. Invokes the USMT Loadstate.exe command to restore the user state that was previously backed up.

The TS.xml file identifies the appropriate steps in each phase based on each type of deployment scenario (Refresh Computer, Replace Computer, and New Computer). Select the properties required during each phase of the deployment process.

For more information on each of the properties used in each phase, see the section, "Properties", in the MDT document *Toolkit Reference*.

Applying MDT Properties to Groups of Computers

Whenever possible, use group-based rules to apply most computer configuration settings. Group-based rules allow the same configuration settings to be applied to a group of client computers. After applying group-based rules, you can supply computer-specific configuration settings using computer-based rules.

Apply properties to groups of computers by performing the following steps:

1. Select the method of grouping multiple computers as described in Select the Method for Grouping Computers.
2. Apply the properties to the groupings of computers as described in Apply the Properties to the Groups.

Select the Method for Grouping Computers

Different methods can be used to group client computers. After determining how to group the computers, select the appropriate properties to help group them.

Using the processing rules in MDT, group computers based on any property that might be applied to a group of computers (such as **Make**, **Model**, or **DefaultGateway**). Table 155 lists methods of grouping computers, a description of the method, and the properties that can be used to group the computers.

Table 155. Methods for Grouping Computers

Expand table

Grouping method	Description	Properties
Geographically	Group configuration settings based on resources located within a geographic region (such as a shared	**DefaultGateway**

Grouping method	Description	Properties
	folder on a computer within a geographic region).	
Target computer hardware attributes	Group configuration settings based on hardware attributes (such as the make of the computer or processor architecture of the target computer).	**Architecture**CapableArchitectureMake
Target computer software attributes	Group configuration settings based on hardware attributes (such as the operating system version of the target computer).	**OSVersion**
Default attributes	Apply configuration settings to all target computers when the properties are not in other sections.	**Default**

In most instances, computer groupings can be nested. For example, you can use the **DefaultGateway** property to designate the IP subnets on which a computer resides within a geographic location. Define locations using the user-defined properties in the `[DefaultGateway]` section, as shown in Listing 11.

> ⓘ **Note**
>
> A variety of methods can be used to group computers by hardware configuration, and the script will search for the substituted value regardless. For instance, if you specify `Priority=Make`, the script substitutes the value for **Make** that it determines through a Windows Management Instrumentation (WMI) call and will look for the corresponding section—for example, `[Dell Computer Corporation]`.

Example: Computer Groupings Selected by Woodgrove Bank

Listing 11 shows an example of how the fictional company, Woodgrove Bank, used `[DefaultGateway]` to designate the configuration settings for a specific location. Three subnets (172.16.0.3, 172.16.1.3, and 172.16.2.3) reside within the NYC location. A separate section, `[NYC]`, includes the configuration settings specific to the NYC location. Similar sections exist for the DALLAS and WASHINGTON locations. This is a special case that allows multiple default gateways to point to the same section. In many environments, a one-to-one mapping might be expected between the `[DefaultGateway]` section and a corresponding section.

Listing 11. Using [DefaultGateway] to Designate Location-Specific Configuration Settings

ini

```ini
[Settings]
Priority=DefaultGateway

[DefaultGateway]
172.16.0.3=NYC
172.16.1.3=NYC
172.16.2.3=NYC
172.16.111.3=DALLAS
172.16.112.3=DALLAS
172.16.116.3=WASHINGTON
172.16.117.3=WASHINGTON

[NYC]
UDShare=\\NYC-AM-FIL-01\MigData
SLShare=\\NYC-AM-FIL-01\Logs
Packages1=NYC00010-Install
Packages2=NYC00011-Install
Administrator1=WOODGROVEBANK\NYC Help Desk Staff

[DALLAS]
UDShare=\\DAL-AM-FIL-01\MigData
SLShare=\\DAL-AM-FIL-01\Logs
Administrator1=WOODGROVEBANK\DAL Help Desk Staff
```

Apply the Properties to the Groups

After identifying how to group configuration settings, determine which properties and corresponding configuration settings to apply to each group. Properties that can be grouped are those you can apply to multiple computers.

Some examples of properties that are typically applied to groups of computers include:

- **BackupDir**
- **BackupShare**
- **CaptureGroups**
- **ComputerBackupLocation**
- **Packages**
- **SLShare**

- **UDDir**
- **UDShare**
- **UDProfiles**

  Properties that are not applied to groups of computers are those specific to a particular computer. Examples of properties that are not appropriate to apply to groups of computers include:

- **OSDAdapter0IPAddress**
- **OSDNewMachineName**

  Example: Group-based Configuration Settings Selected by Woodgrove Bank

  Listing 11 showed an example in which Woodgrove Bank selects group-based configuration settings:

- In the NYC and DALLAS locations, `UDShare`, `SLShare`, and `Administrator1` are specified for each location.
- The servers that `UDShare` and `SLSShare` (`NYC-AM-FIL-01` and `DAL-AM-FIL-01`) reference are within each respective location.
- The Administrator accounts that `Administrator1` (`WOODGROVEBANK\NYC Help Desk Staff` and `WOODGROVEBANK\DAL Help Desk Staff`) reference are unique to each respective location.
- In NYC, location-specific packages are designated by `Packages1` and `Packages2`.

Applying MDT Properties to Individual Computers

After determining the groupings of target computers and configuration settings to be applied to each group, determine the method for identifying individual computers and the configuration settings to assign to each computer. The rules for target computers allow the override or augmentation of group-based processing rules based on the priority of the computer-based rules.

For more information about determining the priority of processing rules, see Priority Reserved Property, earlier in this guide.

Whenever possible, use group-based rules for most client computer configuration settings. Group-based rules allow the same configuration settings to be applied to a

group of computers. After applying group-based rules, you can apply computer-specific configuration settings using computer-based rules.

As when grouping computers, more than one method is available for identifying individual computers. After selecting the method for identifying an individual target computer, select the appropriate properties.

The processing rules allow the identification of computers based on any property that you might apply to an individual of computer (such as **AssetTag**, **MACAddress**, **UUID**, and so on).

Table 156 lists the methods of identifying individual computers, a description of the method, and the properties that you can use to identify the individual computers.

Table 156. Methods for Identifying Individual Computers

Expand table

Identification method	Description	Properties
Target computer hardware attributes	Identify the target computer using the hardware configuration.	**AssetTag**, **MACAddress**, **SerialNumber**, **UUID**, **Product**, **Make**, and **Model**
Target computer software attributes	Identify the target computer using the software or firmware configuration.	**OSVersion**, **IsServerOS**, and **OSSKU**
Target computer user-defined attributes	Identify the target computer using attributes that are assigned to the computer but not part of the hardware or software configuration.	**AssetTag SerialNumber**

Example: Computer Identification Method Selected by Woodgrove

Listing 12 shows an example of how Woodgrove Bank identified computer-based configuration settings. In this instance, Woodgrove used the MAC address of the computer to identify the corresponding configuration settings for the computer (for example, `00:03:FF:CB:4E:C2` and `00:0F:20:35:DE:AC`). The configuration settings for each computer are listed immediately after the section that corresponds to the computer's MAC address.

Listing 12. How Woodgrove Identified Client Computers

```
ini
```

```
[00:03:FF:CB:4E:C2]
ComputerName=WasW2K
OverRideProductKey=TTTTT-VVVVV-WWWWW-XXXXX-YYYYY

[00:0F:20:35:DE:AC]
ComputerName=HPD530-1
OverRideProductKey=AAAAA-BBBBB-CCCCC-DDDDD-EEEEE

[00:03:FF:FE:FF:FF]
ComputerName=BVMXP
OverRideProductKey=11111-22222-33333-44444-55555
```

Example: Computer-based Configuration Settings Selected by Woodgrove

Listing 12 also shows the computer-based configuration settings that Woodgrove Bank selected. Table 157 lists the computer-specific configuration settings applied to each computer.

Table 157. Woodgrove Client Computers and the Corresponding Configuration Settings

Expand table

Target computer	Settings and description
`[00:03:FF:CB:4E:C2]`	`ComputerName` is the name of the computer after deployment—in this case, WasW2K. `OverRideProductKey` is the product key to be assigned to the computer—in this case, TTTTT-VVVVV-WWWWW-XXXXX-YYYYY.
`[00:0F:20:35:DE:AC]`	`ComputerName` is the name of the computer after deployment—in this case, HPD530-1. `OverRideProductKey` is the product key to be assigned to the computer—in this case, AAAAA-BBBBB-CCCCC-DDDDD-EEEEE.
`[00:03:FF:FE:FF:FF]`	`ComputerName` is the name of the computer after deployment—in this case, BVMXP. `OverRideProductKey` is the product key to be assigned to the computer—in this case, 11111-22222-33333-44444-55555.

Configuring MDT Processing Rules

MDT scripts configure computer settings based on rules and configuration settings stored in the CustomSettings.ini file or in the MDT DB. Configure the MDT processing rules by completing the following tasks:

- Configure the processing rules as described in Configure the Rules in the CustomSettings.ini File.
- Configure the processing rules as described in Configure the Rules in the MDT DB.

Configure the Rules in the CustomSettings.ini File

Configure rules in the CustomSettings.ini file. The template version of the CustomSettings.ini file, along with the organization's rules, becomes the customized CustomSettings.ini file.

For LTI deployments, configuring group-based settings might be sufficient, because computer-specific settings can be provided during the MDT installation process. For ZTI deployments using Configuration Manager, add configuration settings unique to a specific client computer, because ZTI assumes that all configuration settings necessary for deployment are configured in advance. These configuration settings can be in addition to or instead of the group-based rules.

Configure the Rules in the MDT DB

Use the Deployment Workbench to configure the rules for LTI and ZTI deployments in the MDT DB. The benefits of using the MDT DB include:

- **It has a more generic version of CustomSettings.ini**. Storing the configuration settings in the MDT DB removes most of the detail from the CustomSettings.ini file. This change helps make the CustomSettings.ini file more generic so that you can use the same file in multiple deployment shares.
- **It is a centralized repository for all property configuration settings**. Centralizing the configuration for all property settings ensures consistency across all deployment shares.

  For information about the MDT DB and using it to perform deployments, see Performing Deployments Using the MDT DB.

Preparing Disks on Target Computers

Prior to deploying the target operating system on a target computer, the MDT deployment process prepares the disks on the target computer for deployment. The disk-preparation process includes the following steps:

1. Create partitions on one or more drives on the target computers.

2. Format one or more partitions on the target computers.
3. Prepare the disks on the target computers for BitLocker.

   Disk preparation occurs during the State Capture and Preinstall phases in the MDT deployment process. The disk-preparation step completed in the State Capture Phase is the disabling of BitLocker on the target computer. The disk-preparation steps completed in the Preinstall Phase create and format the partitions on the target computer.

To prepare the disks on target computers in MDT

1. Review the default partition configuration created by MDT as described in Review the Default Partition Configuration Created by MDT.
2. Prepare for deployment to virtual hard disks (VHDs) with native boot as described in Prepare for Deployment to Virtual Hard Disks with Native Boot.
3. Configure task sequence steps based on the Create Virtual Hard Disk task sequence step type as described in Configure the Create VHD Disk Task Sequence Step Type.
4. Deploy to computers that support the Unified Extensible Firmware Interface specification as described in Deploy to Computers with UEFI.
5. Review the task sequence steps used for saving and restoring user state information as described in Configure Disk Preparation Task Sequence Steps.
6. Configure the MDT properties used in saving and restoring user state information as described in Configure Disk Preparation Properties.

Review the Default Partition Configuration Created by MDT

The MDT deployment processes automatically create the necessary disk partitions to take full advantage of the features provided by the target computer and operating system. By default, MDT creates the partition configuration for BIOS-based computers as described in Table 158.

Table 158. Default Partition Configuration Created by MDT for BIOS-based Computers

Expand table

Partition	File system	Size	Description
BDEDrive	NTFS	512 MB	Unencrypted partition used for starting Windows, also known as the *system partition*. This partition is used to initially start Windows until BitLocker is loaded and the operating system drive can be read. This partition can also be shared with the recovery partition.
OSDisk	NTFS	Remaining space	Partition on which Windows is located, also known as the *operating system drive*. If BitLocker is used, this is the partition that is encrypted.

By default, MDT creates the partition configuration for UEFI-based computers as described in Table 159.

Table 159. Default Partition Configuration Created by MDT for UEFI-based Computers

Expand table

Partition	File system	Size	Description
Description	FAT32	512 MB	Unencrypted partition used for starting Windows. For more information, see Deploy to Computers with UEFI. This partition can also be shared with the recovery partition.
MSR	Unformatted	128 MB	The *Microsoft Reserved Partition* (MSR) is a partition on a data storage device that is labeled with a GUID equal to E3C9E316-0B5C-4DB8-817D-F92DF00215AE. The containing storage device has to use the new GUID Partition Table (GPT) format, not the traditional master boot record (MBR) partition table format.An MSR partition is required on every GPT disk and should be created as the disk is initially partitioned. It should be located after the EFI System Partition (ESP) and any OEM service partitions, but—most importantly—the first data partition must immediately follow it.
OSDisk	NTFS	Remaining space	Partition on which Windows is located, also known as the *operating system drive*. If BitLocker is used, this is the partition that is encrypted.

In addition to the default MDT partition configurations, you can create custom partition configurations. For example, the default MDT partition configurations do not include

other utility partitions or recovery images. For more information, see Understanding Disk Partitions.

Prepare for Deployment to Virtual Hard Disks with Native Boot

Native boot allows VHDs to run on a computer without a VM or hypervisor.

> ⓘ **Note**
>
> Only LTI supports deployment to VHDs with native boot.

Native VHD boot has the following dependencies:

- The local disk must have at least two partitions: a system partition that contains the Windows boot environment files and Boot Configuration Data (BCD) store and a partition to store the VHD file.
- The local disk partition that contains the VHD file must have enough free disk space for expanding a dynamic VHD to its maximum size and for the page file created when booting the VHD. The page file is created outside of the VHD file, unlike in a VM, where the page file is contained inside the VHD.

  You can create LTI task sequences based on the following LTI task sequence templates to deploy to VHDs with native boot:

1. **Deploy to VHD Client Task Sequence**. Select this LTI task sequence template to deploy Windows in a VHD with native boot.
2. **Deploy to VHD Server Task Sequence**. Select this LTI task sequence template to deploy Windows Server in a VHD with native boot.

   These tasks sequences contain the following task sequence steps, which are used to perform deployment to VHDs:

- **Create VHD Disk**. The step creates the VHD file after the physical partition has been created and formatted. For more information about this task sequence step, see Configure the Create VHD Disk Task Sequence Step Type.
- **Format and Partition VHD**. This step formats the .vhd file and is built using the **Format and Partition Disk** task sequence step type. For more information about task sequence step type, see Configure Disk Preparation Task Sequence Steps.
- **Clear OSDDiskIndexVariable**. This step clears the **OSDDiskIndexVariable** task sequence variable that was set during the **Create VHD Disk** task sequence step.

Clearing the **OSDDiskIndexVariable** task sequence variable allows other disks to be partitioned and formatted as a part of the task sequence.

For more information about VHDs with native boot, see Understanding Virtual Hard Disks with Native Boot.

Configure the Create VHD Disk Task Sequence Step Type

The **Create VHD Disk** task sequence step type creates a .vhd file in preparation to performing a deployment to a VHD with native boot support. Table 160 describes how to configure the **Create VHD Disk** task sequence step type.

Table 160. Configure Create VHD Disk Task Sequence Step Type

Expand table

Setting	Description
VHD filename path	Specifies the path to the folder where the .vhd file will be created and can contain one of the following value: - **Random**. This value specifies that the task sequence step will automatically create a unique folder and .vhd file name. The task sequence step automatically places the .vhd file in the folder. - ***vhd_path***. In this case, *vhd_path* is the path to the folder where you want the task sequence step to create the .vhd file, including the name of the .vhd file. The default value for this setting is **Random**.
Diff filename	Specifies the file name for a differencing VHD file: - **Random**. This value specifies that the task sequence step will automatically create a unique folder and file name for the differencing .vhd file. - ***vhd_path***. In this case, *vhd_path* is the path to the folder where you want the task sequence step to create the differencing .vhd file, including the name of the .vhd file. - ***Blank***. In this case, no differencing .vhd file is created. The default value for this setting is **blank**, which indicates that no differencing .vhd file is created.

Setting	Description
VHD size	Specifies the capacity of the .vhd file, which can be specified in megabytes or as a percentage of the available disk space.
Dynamically expanding	This option creates a dynamically expanding .vhd file, which will physically increase in size as more data is stored in the .vhd file. The other option is **Fixed size**. This setting is the default selection.
Fixed size	This option creates a fixed sized .vhd file, which is created as the size specified in VHD size and does not automatically grow in size. The other option is **Dynamically expanding**, which is the default selection.
Retrieve the destination drive for the VHD file from a variable	This specifies the task sequence variable name used to designate the destination drive for the task sequence variable. You can specify any valid task sequence variable in this setting. The default value is the **VHDTargetDisk** task sequence variable. For more information on the **VHDTargetDisk** task sequence variable, see the **VHDTargetDisk** property in the in the MDT document *Toolkit Reference*.
Assign the disk index created from the VHD to a variable	This setting specifies the task sequence variable name used to designate the disk index to be used in deploying the operating system. You can specify any valid task sequence variable in this setting. The default value is the **OSDDiskIndex** task sequence variable. For more information on the **OSDDiskIndex** task sequence variable, see the **OSDDiskIndex** property in the in the MDT document *Toolkit Reference*.

Deploy to Computers with UEFI

The UEFI is a specification that defines a software interface between an operating system and platform firmware. UEFI is a more secure replacement for the older BIOS firmware interface, present in some personal computers, which is vulnerable to malware that performs attacks during startup or power on self-test (POST) processes.

Windows operating systems support firmware revisions that are based on the UEFI version 2.0 or later specification on 64-bit platforms and Intel Itanium platforms. Windows also supports firmware revisions that are based on the EFI Version 1.10 specification on Intel Itanium platforms.

Windows supports a subset of the functionality that is defined in the UEFI 2.0 specification. Windows implementations do not explicitly check against higher revisions of the firmware. The operating system supports higher revisions of the firmware if they contain the necessary support for Windows.

> ⓘ **Note**

The UEFI partitions must be formatted using the FAT32 file system. The NTFS file system is not supported for UEFI boot.

By default, MDT creates the appropriate partitions to support UEFI. If you create custom partition configurations, ensure that you follow the recommendations described in the Review the Default Partition Configuration Created by MDT section.

For more information, see the following resources:

- UEFI Support and Requirements for Windows Operating Systems
- Recommended UEFI-Based Disk-Partition Configurations

Configure Disk Preparation Task Sequence Steps

MDT includes task sequence templates for LTI and ZTI deployments. These task sequence templates include the task sequence steps listed in Table 161, which are used to perform disk-preparation steps.

Table 161. Disk Preparation Task Sequence Steps

Expand table

Task sequence step	Description
Convert Disk to Dynamic	Converts a physical disk from a basic disk type to a dynamic disk type; available in ZTI task sequences only
Enable BitLocker	Configures BitLocker on the target computer; available for LTI and ZTI task sequences
Format and Partition Disk	Creates partitions and formats disks on the target computer; available for LTI and ZTI task sequences
Disable BitLocker	Disables BitLocker on the current operating system drive or on a specific drive; available in ZTI task sequences only
Disable BDE Protectors	Disables the BitLocker protectors on the target computer; available in LTI task sequences only
Create Virtual Hard Disk (VHD)	Creates a .vhd file in preparation for deploying Windows to a VHD with native boot support.

Configure Disk Preparation Properties

Table 162 lists the MDT properties that control the preparation of disks on the target computers. You can configure these properties in CustomSettings.ini or in the MDT DB. For more information about the properties in Table 162, see the corresponding section for each property in the MDT document *Toolkit Reference*.

Table 162. Disk Preparation Properties

Expand table

Property	Description
BDEDriveLetter	The drive letter for the partition that is not encrypted with BitLocker, also known as the SYSVOL
BDEDriveSize	The size of the BitLocker system partition in megabytes
BDEInstall	The type of BitLocker installation you are performing
BDEInstallSuppress	Indicates whether the deployment process should skip BitLocker installation
BDEKeyLocation	The location for storing the BitLocker recovery key and startup key
BDEPin	The PIN to be assigned to the target computer when configuring BitLocker and the **BDEInstall** or **OSDBitLockerMode** properties are set to a value of **TPMPin**. This property can contain numeric only or alphanumeric values based on the value of the **BDEAllowAlphaNumericPin** property.
BDERecoveryKey	A Boolean value that indicates whether the process creates a recovery key for BitLocker
BDEWaitForEncryption	Indicates whether the deployment process should not proceed until BitLocker has finished the encryption process for all specified drives
DestinationDisk	Disk number to which the image will be deployed
DestinationLogicalDrive	The logical drive to which the image will be deployed
DestinationPartition	Disk partition to which the image will be deployed
OSDPartitions	The number of defined partition configurations (The maximum number of partitions you can configure is two; the default is none.)

Property	Description
OSDPartitionsxBootable	The partition that should be set to bootable (The default first partition is set to bootable.)
OSDPartitionsxFileSystem	The type of file system for the partition (Valid values are **NTFS** or **FAT32**.)
OSDPartitionsxQuickFormat	Indicates whether the partition should be quick formatted (The default is TRUE.)
OSDPartitionsxSize	The size of the partition
OSDPartitionsxSizeUnits	The units of measure for specifying the size of the partition (Valid values are **MB**, **GB**, or **%**. The default value is **MB**.)
OSDPartitionsxType	The type of partition to be created
OSDPartitionsxVolumeLetterVariable	The drive letter to be assigned to the partition
OSDPartitionsxVolumeName	The volume name that will be assigned to the partition
WipeDisk	Indicates whether the disk should be wiped

Saving and Restoring User State Information

User state information consists of the user profile information, Internet Explorer favorites, data files, and other user-specific data stored on the target computer. The MDT deployment process can automatically capture and restore user state information on the target computers.

The MDT deployment process uses USMT to save and restore user state information. During the State Capture Phase in the MDT deployment process, USMT saves the user state information to a desired location. Later, during the State Restore Phase, USMT restores this user state information.

To save and restore user state information on target computers in MDT

1. Review the task sequence steps used for saving and restoring user state information as described in Review User State Information Task Sequence Steps.
2. Configure the MDT properties used in saving and restoring user state information as described in Configure User State Information Properties.

3. Customize the USMT XML control files as described in Configure User State Migration XML Control Files.

4. Configure MDT to perform user state capture in Windows PE (offline) or in the existing operating system (online) as described in Configure USMT Offline User State Migration.

Review User State Information Task Sequence Steps

MDT includes task sequence templates for LTI and ZTI deployments for Configuration Manager. These task sequence templates include the task sequence steps listed in Table 163, which are used to save and restore user state information.

Table 163. User State Information Task Sequence Steps

Expand table

Task sequence step	Description
Generate Application Migration Files	Generates an XML file used to identify documents created by applications installed on the target computer
Capture User State	Captures user state information based on the application migration files that the **Generate Application Migration Files** task sequence step generates and the user state information properties in CustomSettings.ini or the MDT DB for LTI deployments
Capture Groups	Captures the group membership of the local groups on the target computer based on the user state information properties in CustomSettings.ini or the MDT DB
Restore User State	Restores the user state information that the **Capture User State** task sequence step saved to the target computer
Restore Groups	Restores the group membership information that the **Capture Groups** task sequence step saved to the target computer
Offline User State Capture	Captures user state information while running in Windows PE (offline) instead of the orignial operating system (online). This task sequence step runs the ZTIUserState.wsf script and is run when the following conditions are met: - The **_SMSTSMediaType** property is not equal to **"OEMMedia"**. - The **OSDDiskPart** property is not equal to **"TRUE"**.

Task sequence step	Description
	This task sequence step is a part of the Offline USMT group that is run when the **USMTOfflineMigration** equals **"TRUE"**.

Configure User State Information Properties

Table 164 lists the MDT properties for LTI deployments that control saving and restoring user state information. You can configure these properties in the CustomSettings.ini file or in the MDT DB. For more information about the properties in Table 164, see the corresponding section for each property in the MDT document *Toolkit Reference*.

Table 164. User State Information Properties for LTI Deployments

Expand table

Property	Description
LoadStateArgs	List of parameters passed to the Loadstate.exe tool
ScanStateArgs	List of parameters passed to the Scanstate.exe tool
UserDataLocation	Indicates where the user state migration data should be saved
UDDir	The folder in which the user state migration data is stored (This folder exists beneath the network shared folder specified in the **UDShare** property.)
UDProfiles	A comma-delimited list of user profiles that the Scanstate.exe tool must save during the State Capture Phase
UDShare	The network share in which user state migration data is stored
USMTOfflineMigration	Indicates whether an USMT offline migration should be performed. An offline migration is performed while the computer is started in Windows PE instead of the operating system currently installed on the target computer.

> **Important**
>
> USMT will by default capture all local and domain user accounts unless explicitly excluded. Any captured local accounts will then, by default, be included in the

restore process. In some circumstances the restore step will fail without the inclusion of the /lae parameter to set the password for these local accounts.

Configure User State Migration XML Control Files

USMT uses the default versions of the migration XML files unless the path to the custom XML control files is indicated. Customize the user state migration XML control files for USMT by performing the following tasks:

- Configure the XML control files for USMT for LTI deployments as described in Configure User State Migration XML Control Files for LTI Deployments.
- Configure the XML control files for USMT for ZTI deployments as described in Configure User State Migration XML Control Files for ZTI Deployments.

Configure User State Migration XML Control Files for LTI Deployments

For LTI deployments, insert one or more lines in the CustomSettings.ini file that contain the **USMTMigFiles** property for each of the USMT migration XML control files that you want to specify. The XML files need to be copied into either the USMT folder or the Scripts folder in the distribution share.

Use the following format for these lines:

ini

```
USMTMigFiles1=MigApp.xml
USMTMigFiles2=MigUser.xml
USMTMigFiles3=MigSys.xml
USMTMigFiles4=MigCustom.xml
USMTConfigFile=Config.xml
```

> **Note**
>
> See the MDT document *Toolkit Reference* for details on configuration settings.

Configure User State Migration XML Control Files for ZTI Deployments

For ZTI deployments for Configuration Manager, insert a line in the CustomSettings.ini file that contains the **OSDMigrateConfigFiles** task sequence variable for the USMT

migration XML control file that you want to specify. If you specify the **OSDMigrateConfigFiles** property, insert another line that sets the **OSDMigrateMode** task sequence variable to Advanced.

Use the following format for these lines:

```ini
OSDMigrateMode=Advanced
OSDMigrateConfigFiles=MigApp.xml,MigUser.xml
```

The path to the XML control files is relative to the current folder, which will be the location of the USMT package. If you keep the XML control files in the USMT package, update this package each time you modify any of the XML control files. Otherwise, you can store the XML control files in a separate package or network shared folder and specify a fully qualified UNC path to the package or network shared folder.

Configure USMT Offline User State Migration

USMT can perform offline migration of user state from a computer. In an offline migration, the capture is performed in Windows PE instead of the existing operating system. The advantages of performing an offline user state migration are:

- You do not need to log on to the computer on which you are capturing user state.
- Hardware resources and files are more readily accessible by ScanState and other USMT tools.
- Performance may increase on older computers that have limited hardware resources and numerous software applications.
- Doing so may help avoid conflicts where a file is in use by another application or service.
- You may be able to use an offline migration to recover files and settings if a computer no longer starts properly.

  The disadvantage of performing an offline user state migration is that some user settings are not captured but rather can only be captured while running USMT in the target operating system. For a list of the settings that are not captured when performing an offline user state capture, see What Does USMT Migrate?

  You can perform USMT offline user state migration in the MDT:

- In a Refresh Computer deployment scenario using LTI, ZTI, or UDI when the value of the **USMTOfflineMigration** property is set to **"TRUE"**
- In a New Computer deployment scenario using LTI with the **Move Data and Settings** wizard page in the **Deployment Wizard** or if the value of the **USMTOfflineMigration** property is set to **"TRUE"**

> ⓘ **Note**
>
> You cannot perform USMT offline user state migration in the MDT New Computer deployment scenario using ZTI.8

Joining Target Computers to AD DS Domains

One of the final steps in completing the deployment of a target operating system to the target computers is joining the computer to an AD DS domain. Although you can complete this process manually, MDT supports the following automated methods for joining target computers to AD DS domains:

- Using the Deployment Wizard as described in Join Domains Using the Deployment Wizard
- Modifying CustomSettings.ini as described in Join Domains by Modifying the CustomSettings.ini File
- Modifying Unattended.xml as described in Join Domains by Modifying the Unattended.xml File
- Using the **Recover from Domain Join Failure** task sequence step type as described in Join Domains Using the Recover from Domain Join Failure Task Sequence Step Type
- Using the Windows offline domain join feature as described in Join Domains Using Offline Domain Join

Join Domains Using the Deployment Wizard

For LTI deployments, the **Join the computer to a domain or workgroup** wizard page in the Windows Deploy Wizard in MDT allows you to interactively provide the configuration settings necessary to join a domain. Table 165 lists the configuration settings on this wizard page used in joining a domain.

Table 165. Configuration Settings on the Join the computer to a domain or workgroup wizard page for Joining Domain

Expand table

Setting	Description
Join a domain	Select to configure the Deployment Wizard to join the target computer to a domain.
Domain	Specifies the domain to which the target computer is to be joined. This text box is enabled only when you select **Join a domain**.
User Name	Specifies the account to be used in joining the target computer to the domain specified in **Domain**. This text box is enabled only when you select **Join a domain**.
Password	Specifies the password for the account specified in **User Name**. This text box is enabled only when you select **Join a domain**.
Domain	Specifies the domain in which the account specified in **User Name** is located. This text box is enabled only when you select **Join a domain**.
Organizational Unit	Specifies the OU in which the computer account will be created in the domain specified in **Domain**. This text box is enabled only when you select **Join a domain**.

For more information about completing the **Join the computer to a domain or workgroup** wizard page in the Windows Deploy Wizard, see Complete the Deployment Wizard.

Join Domains by Modifying the CustomSettings.ini File

You can automate the domain-join process for LTI or ZTI deployments by modifying the properties listed in Table 166 in the CustomSettings.ini file used in the MDT deployment process.

Table 166. Properties in CustomSettings.ini to Modify for Joining a Domain

Expand table

Property	Description
DomainAdmin	The user account credentials used to join the target computer to the domain specified in **JoinDomain**; specify as *domain\user_name* or *user_name@domain.com*
DomainAdminDomain	The domain in which the user's credentials specified in **DomainAdmin** reside
DomainAdminPassword	The password used for the domain Administrator account specified in the **DomainAdmin** property to join the computer to the domain
JoinDomain	The domain that the target computer joins after the target operating system is deployed (This is the domain in which the computer account for the target computer is created. The **JoinDomain** property can contain alphanumeric characters, hyphens [-], and underscores [_]. The **JoinDomain** property cannot be blank or contain spaces.)
MachineObjectOU	The AD DS OU in the target domain in which the computer account for the target computer is created

Join Domains by Modifying the Unattended.xml File

You can automate the domain-join process for LTI or ZTI deployments by modifying the settings listed in Table 167 in the Unattended.xml file used in the MDT deployment process.

Table 167. Settings in Unattended.xml to Modify for Joining a Domain

Expand table

Setting	Description
Username	The user account credentials used to join the target computer to the domain specified in **JoinDomain**
Domain	The domain in which the user's credentials specified in **Username** reside
Password	The password used for the domain Administrator account specified in the **Username** setting to join the computer to the domain
JoinDomain	The domain that the target computer joins after the target operating system is deployed
MachineObjectOU	The AD DS OU in the target domain in which the computer account for the target computer is created

For more information about these settings, see Microsoft-Windows-UnattendedJoin.

Join Domains Using the Recover from Domain Join Failure Task Sequence Step Type

Task sequence steps based on the **Recover from Domain Join Failure** task sequence step type retry the domain-join process using the configuration information specified in CustomSettings.ini. You can configure the **Recover from Domain Join Failure** task sequence step type to recover using one of the following methods:

- **Auto Recover (Rerun Join Domain)**. This method automatically retries the domain-join process without intervention. Select this method when you want the MDT process to automatically retry the domain-join process.
- **Manual Recover (Allow user to Join Domain)**. This method allows the user running the Deployment Wizard to retry the domain-join process. Select this method when you want the MDT process to allow the user to retry the domain-join process.
- **No Recover (Stop script execution)**. This method automatically terminates the task sequence if the computer has not successfully joined the domain. Select this method when you want MDT to stop running the task sequence if the computer has not successfully joined the domain.

To configure task sequence steps based on the Recover from Domain Join Failure task sequence step type

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/Task Sequences (where *deployment_share* is the name of the deployment share in which you will configure the task sequence).
3. In the details pane, select ***task_sequence_name*** (where *task_sequence_name* is the name of the task sequence you want to configure).
4. In the Actions pane, select **Properties**.

   The ***task_sequence_name*** **Properties** dialog box opens (where *task_sequence_name* is the name of the task sequence you want to configure).

5. On the **Task Sequence** tab, in the task sequence hierarchy, go to *task_sequence_step*, and then select the **Properties** tab.

6. On the **Properties** tab, configure the settings listed in Table 168 based on the requirements of your organization, and then select **OK**.

Table 168. Configuration Settings on the Properties Tab of the Recover from Domain Join Failure Task Sequence Step Type

Expand table

Setting	Description
Type	Contains the task sequence type, which is always set to **Recover from Domain Join Failure**
Name	Contains name of the task sequence step displayed in the task sequence
Comments	Provides descriptive information about the task sequence step
Auto Recover (Rerun Join Domain)	Select to configure the task sequence step to automatically retry the domain-join process without intervention
Manual Recover (Allow user to Join Domain)	Select to configure the task sequence step to allow the user to retry the domain-join process
No Recover (Stop script execution)	Select to configure the task sequence step to stop the task sequence if the computer has not successfully joined the domain

Join Domains Using Offline Domain Join

Offline domain join is a process to join a domain without contacting a domain controller. This process makes it possible to join computers to a domain in locations where there is no connectivity to a corporate network.

Using offline domain join, target computers can be joined to the domain when they initially start after the installation of the target operating system. No additional restart is required to complete the domain-join process, which can significantly reduce the overall time required for wide-scale VM deployments.

The offline domain join is a variant on joining domains by modifying the Unattend.xml file. The Unattend.xml file includes the `Microsoft-Windows-UnattendJoin/Identification/Provisioning` section, which includes the configuration settings for performing a domain join.

For more information about:

- The offline domain join process, see Offline Domain Join (Djoin.exe) Step-by-Step Guide
- Configuring the Unattended.xml file to perform offline join, see the section, "Performing an offline domain join by using an unattended operating system installation," in Offline Domain Join (Djoin.exe) Step-by-Step Guide

Deploying Software Updates to Target Computers

In addition to the target operating system, applications, device drivers, and other software components, you may need to apply software updates to all these software components. These software updates are required to ensure a consistent configuration baseline for all the target computers.

Deploy software updates to target computers in MDT by:

- Selecting the appropriate strategies for deploying software updates as described in Select the Software Update Deployment Strategy
- Deploying software updates using Windows Update Agent-based technologies for LTI deployments as described in Deploy Software Updates with Windows Update Agent for LTI Deployments
- Deploying software updates using the Deployment Workbench for LTI deployments as described in Deploy Software Updates with the Deployment Workbench for LTI Deployments
- Deploying software updates using Configuration Manager for ZTI deployments as described in Deploy Software Updates with Configuration Manager for ZTI Deployments

Select the Software Update Deployment Strategy

The software update deployment strategies are based on when the software updates are to be installed. You can install software updates:

- As a part of the image deployed to the target computers

- After the target operating system is deployed to the target computers

Deploy Software Updates with Windows Update Agent for LTI Deployments

In LTI deployments, you can install software updates from Windows Update or from WSUS using a task sequence step that runs the ZTIWindowsUpdate.wsf script. Some of the LTI task sequence templates provided in MDT include the **Windows Update (Pre-Application Installation)** task sequence step and the **Windows Update (Post-Application Installation)** task sequence step.

You can also create a custom task sequence step based on the **Run Command Line** task sequence step type that runs the following command line:

Windows Command Prompt

```
Cscript.exe "%SCRIPTROOT%\ZTIWindowsUpdate.wsf"
```

Deploy Software Updates with the Deployment Workbench for LTI Deployments

In LTI deployments, you can install software updates for Windows in the Packages node in the Deployment Workbench using a task sequence step based on the **Install Updates Offline** task sequence step type. Some of the LTI task sequence templates provided in MDT include the **Apply Patches** task sequence step, which is based on the **Install Updates Offline** task sequence step type.

You can control the software updates deployed to the target computers by this method using selection profiles. The **Install Updates Offline** task sequence step allows you to specify a selection profile so that you can specify which software updates to deploy. If you want to deploy software updates based on multiple selection profiles, create a task sequence step for each selection profile, and then specify the corresponding selection profile in the task sequence step.

For more information on creating selection profiles, see Create a New Selection Profile in the Deployment Workbench.

Deploy Software Updates with Configuration Manager for ZTI Deployments

In ZTI deployments using Configuration Manager, you can initiate software updates using a task sequence step based on the **Install Software Updates** task sequence step type. The **Install Software Updates** task sequence type allows you to install only mandatory or all software updates in a single task sequence step using one of the configuration options listed in Table 169.

Table 169. Configuration Settings on the Properties Tab of the Install Software Updates Type Task Sequence Step

Expand table

Setting	Description
Name	Configures the name of the task sequence step displayed in the task sequence hierarchy
Description	Configures the description text for the task sequence step
Mandatory Software Updates	Selecting configures the task sequence step to install only mandatory software updates
All Software Updates	Selecting configures the task sequence step to install all software updates, including mandatory software updates

For more information about the **Install Software Updates** task sequence type, see the section, "Install Software Updates," in the section, "Task Sequence Steps in Configuration Manager," in the Configuration Manager Documentation Library, which is installed with Configuration Manager.

Managing Device Drivers

Device driver management is a critical component in deploying operating systems to target computers. The proper device drivers must be available to Windows PE and to the target operating system for the deployment to be successful.

Manage device drivers using MDT by:

- Selecting the appropriate strategies for managing device drivers as described in Select the Device Driver Management Strategy
- Managing device drivers using the Deployment Workbench for LTI deployments as described in Control Device Driver Deployments for LTI

- Managing device drivers using Configuration Manager for ZTI deployments as described in Control Device Driver Deployments Using Configuration Manager for ZTI
- Resolving device driver signing issues as described in Resolve Device Driver Signing Issues

Select the Device Driver Management Strategy

The following are the high-level strategies for performing device driver management:

- **Include all device drivers**. This is the default behavior for LTI and ZTI deployments. In this strategy, all the drivers are deployed to the target computer. Then, Windows PE and the target operating system use Plug-and-Play IDs to identify the device drivers needed for the devices on the target computers.
- **Include only the device drivers specifically required for the target computer**. In this strategy, only the device drivers specific to the target computer are deployed to the target computer. This requires that you configure the ZTI and LTI process to control which device drivers are deployed to the target computer.

  Table 170 lists the advantages and disadvantages of these device driver management strategies.

Table 170. Advantages and Disadvantages of Device Driver Management Strategies

Expand table

Strategy	Advantages	Disadvantages
Include all device drivers	- Requires less initial time and effort to identify the appropriate drivers. - Works well when there are fewer device drivers to manage.	- Images are larger. - Images require more frequent version updates. - Ongoing management of device drivers in the image requires more effort, because there are more drivers in the image. - Does not work well if there are a large number of device drivers to manage.

Strategy	Advantages	Disadvantages
Include only the device drivers specifically required for the target computer	- Ongoing management of device drivers in the image requires less effort, because there are fewer drivers in the image. - Images are smaller. - Images require less frequent version updates. - Works well when there is a large number of device drivers to manage.	- Requires more initial time and effort to identify the appropriate drivers. - Can introduce unnecessary management overhead when there are fewer device drivers to manage.

Either of these strategies can cause problems if taken to the extreme. The "Include all device drivers" strategy might cause problems when trying to manage tens of thousands of device drivers because of the larger download sizes and a higher probability of detecting the wrong device drivers. A complex management structure used in the "Include only the device drivers specifically required for the target computer" strategy might be too complex for managing small environments with only two or three different types of computers and cause you to spend a lot of unnecessary time up front.

In most instances, select a device driver management strategy that is a hybrid of these strategies and best fits your organization.

Control Device Driver Deployments for LTI

The goal of managing device drivers for LTI deployments is to help ensure that only the appropriate device drivers are deployed to the target computers without introducing unnecessary effort and management overhead. The high-level approach to device driver management using the Deployment Workbench for LTI deployments is as follows:

1. Create a folder structure in the Out-of-Box Drivers node of the Deployment Workbench to organize the device drivers as described in Create Folders to Organize Device Drivers for LTI Deployments.

2. Create selection profiles used to select the device drivers for deployment based on the folder structure you created in the previous step as described in Create Selection Profiles to Select the Device Drivers for LTI Deployments.

3. Configure tasks sequences to deploy the device drivers in the selection profiles as described in Configure Task Sequences to Deploy Device Drivers in Selection

Profiles for LTI Deployments.

Create Folders to Organize Device Drivers for LTI Deployments

Create folder structures in the Out-of-Box Drivers node in the Deployment Workbench to provide the level of control you want for deploying device drivers to target computers. The folder structure groups or categorizes device drivers so that you can select specific groupings or categories of drivers using selection profiles.

Select any combination of the following methods for creating folder structures:

- **Target operating system**. Select this method to deploy only the device drivers to the target computer based on the respective target operating system.
- **Processor architecture of the target computers**. Select this method to deploy only the device drivers to the target computers based on the respective processor architecture (32-bit or 64-bit).
- **Manufacturer (make) and model of the target computers**. Select this method to deploy device drivers to the target computer based on the respective make and model of the target computer.

  > **Tip**
  >
  > If the name of the folder matches the value that the BIOS returned for make and model, you can select the appropriate folder structure based on task sequence variables later in the process.

  The complexity of the folder structure that you create is based on the level of precision you want for deploying the device drivers. The more precision you want, the more complex the folder structure.

  You can also select a method for creating folder structures based on specific problems you may be having or to mitigate an existing problem. For example, you may have or expect to have one of the following problems:

- **Device drivers are being selected for the wrong operating system**. In this case, you could organize your device drivers by creating the following folder structure in the Out-of-Box Drivers node in the Deployment Workbench based on operating systems, and then place the device drivers in the corresponding folders:
  - Windows 7

- Windows Server 2008 R2
- Windows Server 2012
- Windows 8

- **Device drivers are being selected for the wrong processor architecture**. In this case, you could organize your device drivers by creating the following folder structure in the Out-of-Box Drivers node in the Deployment Workbench based on processor architecture, and then place the device drivers in the corresponding folders:
  - x86
  - x64
- **Device drivers are being selected for the wrong make and model of target computer**. In this case, you could organize your device drivers by creating the following folder structure in the Out-of-Box Drivers node in the Deployment Workbench based on the make and model of the target computer, and then place the device drivers in the corresponding folders:
  - make_01\

    ...\model_01

    ...\model_02

    ...\model_03
  - make_02\

    ...\model_aa

    ...\model_ab
  - make_03\

    ...\model_xx

    ...\model_xy

  Example: Woodgrove Bank Device Driver Folder Structure for LTI

  Woodgrove Bank has decided that it wants to maintain precise control over the device drivers deployed to target computers. So, its IP pros create a folder structure in the Out-of-Box Drivers node in the Deployment Workbench that

organizes the device drivers by make and model, operating system, and processor architecture:

make\model\operating_system\architecture

Figure 14 illustrates the folder structure Woodgrove Bank created.

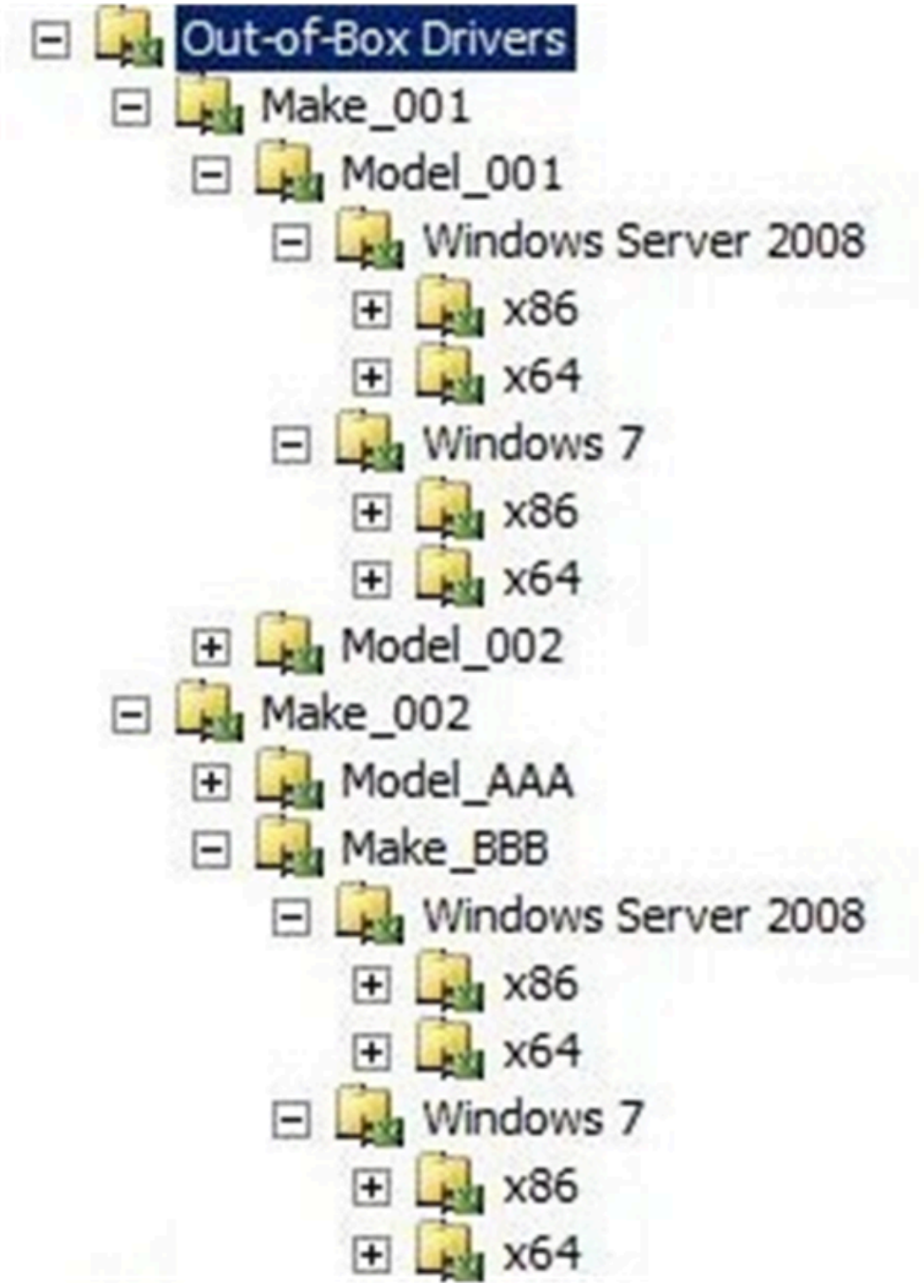


Figure 14. Device driver folder structure created by Woodgrove Bank

Create Selection Profiles to Select the Device Drivers for LTI Deployments

Create selection profiles to identify the combination of device drivers that you want to deploy to specific target computers based on the folder structure you created in the Out-of-Box Drivers node in the Deployment Workbench. The LTI deployment process uses selection profiles to determine the device drivers to deploy in the **Inject Drivers** task sequence step type, in CustomSettings.ini, and in the MDT DB.

By default, selection profiles deploy the device drivers in the selected folder and subfolders. Create selection profiles based on the level of control you want to have over the device drivers being deployed. If you create selection profiles on folders:

- Higher in the folder structure, more device drivers are included, and you have less granular control over the device drivers deployed
- Lower in the folder structure, fewer device drivers are included, and you have more granular control over the device drivers deployed

> **Tip**
>
> Use selection profile names that allow you to easily identify the device drivers included in them, such as Windows 7 32-bit and 64-bit Device Drivers, Windows 8 64-bit Device Drivers, or Fabrikam - Model A532- 32-bit Device Drivers.

Example: Woodgrove Bank Device Driver Selection Profiles for LTI Deployments

Woodgrove Bank has decided that it wants to maintain precise control over the device drivers deployed to target computers. So, its IT pros create a selection profile for each leaf-level folder in following folder structure:

make\model\operating_system\architecture

Woodgrove Bank named the selection profiles based on the folder structure in the format as follows:

make-model-operating_system-architecture

The following is an example of the Woodgrove Bank selection profile naming convention for 64-bit device drivers for Windows 8 running on a computer with "Fabrikam" as the make and "FK5323" as the model:

"Fabrikam-FK5323-Win8-x64"

Configure Task Sequences to Deploy Device Drivers in Selection Profiles for LTI Deployments

Modify The configuration for your task sequences to reference the selection profiles and deploy the appropriate device drivers to the target computers. Selection profiles are exposed to the LTI deployment process as:

- Selection profiles that can be configured in the Deployment Workbench, the CustomSettings.ini file, or the MDT DB
- Device driver groups that can be configured in the CustomSettings.ini file or the MDT DB

  Selection profiles and device driver groups are additive. The LTI deployment process creates the list of device drivers to deploy based on the union of both selection profiles and device driver groups, which can cause unpredictable results, because the default selection profile and default device driver group include all device drivers. For example, if you specify a selection profile that contains only 64-bit device drivers and leave the default device driver group, the result will include all device drivers.

  To change this behavior, specify:

- The**Nothing** device driver group in the CustomSettings.ini file or the MDT DB to allow the selection profile to control the device drivers deployed to the target computer
- The**Nothing** selection profile in the Inject Driver task sequence step, the CustomSettings.ini file, or the MDT DB to allow the device driver group to control the device drivers deployed to the target computer
- A specific set of device drivers for the selection profile and the device driver group so that a known set of device drivers is deployed

  The following are strategies for configuring task sequences to control the deployment of device drivers for LTI deployments:

- For a single selection profile, modify the **Inject Driver** task sequence step, which is in most of the LTI task sequence templates, to use the selection profile.
- For a limited number of selection profiles, add an **Inject Driver** task sequence step for each selection profile, and then configure each task sequence step with the corresponding selection profile.
- Configure a single **Inject Driver** task sequence step in your task sequence, and then override the selection profile specified in the **Inject Driver** task sequence step using the **DriverSelectionProfile** task sequence variable in the CustomSettings.ini file or the MDT DB.

> ⓘ Note
>
> The selection profile you specify in the **DriverSelectionProfile** task sequence variable overrides all **Inject Driver** task sequence steps in a task sequence. If you have multiple all **Inject Driver** task sequence steps in your task sequence, they all will be overridden.

- Configure a single **Inject Driver** task sequence step in the task sequence (which already exists in most of the LTI task sequence templates), but specify additional device drivers to be added using the **DriverGroup** task sequence variable. Because selection profiles and device driver groups are additive, the device drivers specified in the **DriverGroup** task sequence variable are deployed in addition to the device drivers in the selection profile.

- If you want to use the **DriverGroup** task sequence variable for backward compatibility with previous versions of MDT, configure the **Inject Driver** task sequence step to use the **Nothing** selection profile.

  > ⓘ Note
  >
  > You can also use the **DriverPaths** property to specify the UNC path to the folders containing the device drivers to deploy. However, this method is provided for backward compatibility with previous versions of MDT. Instead, use selection profiles or the **DriverGroup** task sequence variable.

  Example: Woodgrove Bank Device Driver Task Sequence Configuration for LTI Deployments

  Woodgrove Bank has decided that it wants to maintain precise control over the device drivers deployed to target computers. Its IT pros have created a device driver folder structure and selection profile strategy that allows detailed control of device drivers.

  The IT pros configured their task sequences using the **Inject Driver** task sequence step in their task sequences with the most common configuration in their organization, and then used the **DriverSelectionProfile** task sequence variable in the CustomSettings.ini file and the MDT DB to override the selection profile specified in the **Inject Driver** task sequence step as necessary.

  The IT pros added the following line in their CustomSettings.ini file for the **DriverSelectionProfile** task sequence variable:

```ini
DriverSelectionProfile =%MAKE%-%MODEL%-Win8-%ARCHITECTURE%
```

> **Note**
>
> The operating system is a static value for the **DriverSelectionProfile** task sequence variable, because the task sequence will deploy only one operating system.

Control Device Driver Deployments Using Configuration Manager for ZTI

ZTI deployments in Configuration Manager use the driver catalog in Configuration Manager as the central repository for device drivers. After you import device drivers into the driver catalog, you can organize them by:

- **Device driver packages**. Like software packages, device driver packages are distributed to distribution points so that they are accessible to the target computers. You can create multiple device driver packages to group device drivers to be deployed to the target computer, such as the make and model of target computer. You can control the device drivers deployed based on the device driver packages using the **Apply Driver Package** task sequence step.

- **Device driver categories**. Device driver categories allow you to specify a category for each device driver you import into the driver catalog. Categories allow you to group device drivers based common characteristics, such as all network adapter drivers or by processor architecture. You can control the device drivers deployed to the target computer based on the device driver categories using the **Auto Apply Drivers** task sequence step.

  Most of the ZTI task sequences created using the MDT task sequence templates include the **Auto Apply Drivers** task sequence step. By default, this step configures the task sequence to deploy all device drivers to the target computer, allowing the target operating system to select the device drivers required.

  The list that follows provides strategies for configuring task sequences to control the deployment of device drivers for ZTI deployments in Configuration Manager:

- Create multiple device driver categories that are configured in multiple Auto **Apply Drivers** task sequence steps. Control the deployment of device drivers by performing the following steps:

1. Create device driver categories based on the level of granularity you want in controlling the device drivers to be deployed.
2. Add multiple **Auto Apply Drivers** task sequence steps based on the number of combination categories you want to deploy.
3. Configure the **Auto Apply Drivers** task sequence step to use the proper combination of device driver categories using the **Limit driver matching to only consider drivers in selected categories** list box.
4. Configure the task sequence step conditions based on the contents of the device driver package.

   For example, if the device driver packages are based on the make and model of the target computer, configure the task sequence step to run when the **Make** and **Model** task sequence variables equal the make and model for the device driver package.

- Create multiple device driver packages that are configured in multiple **Apply Driver Package** task sequence steps. Control the deployment of device drivers by performing the following steps:

  1. Create device driver packages based on the criteria for limiting the device drivers to be deployed.
  2. For each device driver package, add an **Apply Driver Package** task sequence step for each category.
  3. Configure the **Apply Driver Package** task sequence step to use the corresponding device driver package.
  4. Configure the task sequence step conditions based on the contents of the device driver package.

     For example, if the device driver packages are based on the make and model of the target computer, configure the task sequence step to run when the Make and **Model** task sequence variables equal the make and model for the device driver package.

- Configure a single **Auto Apply Drivers** task sequence step in your task sequence, and then override the selection profile specified in the **Auto Apply Drivers** task sequence step using the **OSDAutoApplyDriverCategoryList** task sequence variable in the CustomSettings.ini file or the MDT DB. Control the deployment of device drivers by performing the following steps:

1. Create device driver categories based on the level of granularity you want in controlling the device drivers to be deployed.

2. Add a new or an existing **Auto Apply Drivers** task sequence step in your task sequence.

3. Configure the **Auto Apply Drivers** task sequence step to use any of the device driver categories using the **Limit driver matching to only consider drivers in selected categories** list box.

   > ⓘ **Note**
   >
   > The device driver category you select is not important, as the category will be overridden by the **OSDAutoApplyDriverCategoryList** task sequence variable.

4. Determine GUIDs for each device driver category you created by running the following script, substituting *strSiteCode* with your site code, *strServer* with you Configuration Manager site server, and *strDriverCatName* with the name of a device driver category you created:

   VB

   ```
   strSiteCode = "NYC"
   strServer = "CMSERVER"
   strDriverCatName = "Fabrikam"
   set objWMIService= GetObject("winmgmts:
   {impersonationlevel=impersonate}!\\" & strServer &
   "\root\sms\site_" & strSiteCode)

   set DriverGUIDS = objWMIService.ExecQuery("select
   CategoryInstance_UniqueID from CMDB_categoryinstance where
   LocalizedCategoryInstanceName = '" & strDriverCatName & "'")
   For each DriverGuid in DriverGuids
       wscript.echo DriverGuid.CategoryInstance_UniqueID
   Next
   ```

5. Modify the CustomSettings.ini file as follows, substituting *SECTION* with the name of a section (such as `[Default]`) and *GUID* with the GUID you retrieved in the previous step:

   ini

   ```
   [Settings]
   Properties=OSDAutoAPplyDriverCategoryList
   ```

```
[SECTION]
OSDAutoApplyDriverCategoryList=DriverCategories:GUID
```

- When performing deployments using stand-alone media, use an **Apply Driver Package** task sequence step, because the **Auto Apply Drivers** task sequence requires connectivity to a management point, and the stand-alone media will not attempt a connection to a management point.

Resolve Device Driver Signing Issues

Digital signatures tell you whether a device driver is provided by a legitimate publisher. Windows features take advantage of code-signing technologies, and requirements for security in the operating system enforce the use of digital signatures for some kinds of code.

In many instances, device drivers from vendors are already signed. However, there may be instances where you modify the files include with the device drivers and need to sign the device drivers again. For example, you might need to modify an INF file for a device driver, and then sign the device driver.

Review the following resources to help you resolve device driver signing issues:

- Driver Signing Requirements for Windows
- Device Management and Installation Step-by-Step Guide: Signing and Staging Device Drivers in Windows 7 and Windows Server 2008 R2

Running Orchestrator Runbooks

System Center 2012 Orchestrator can tie disparate tasks and procedures together by using the Runbook Designer graphical user interface to create reliable, flexible, and efficient end-to-end solutions in the IT environment.

You can carry out the following tasks using Orchestrator:

- Automate processes in your data center, regardless of hardware or platform.
- Automate your IT operations and standardize best practices to improve operational efficiency.
- Connect different systems from different vendors without having to know how to use scripting and programming languages.

An Orchestrator *runbook* is the sequence of activities that orchestrate actions on computers and networks. You can initiate Orchestrator runbooks in MDT using the **Execute Runbook** task sequence step type.

> **Note**
>
> The **Execute Runbook** task sequence step is not included any MDT task sequence templates. You must add the **Execute Runbook** task sequence step to any task sequences you create.

To configure the Execute Runbook task sequence step type to run Orchestrator runbooks

1. Edit ***task_sequence_name*** (where *task_sequence_name* is the name of the task sequence to which you want to add the task sequence step) for:
   - LTI as described in Configure the Task Sequence Properties Task Sequence Tab
   - ZTI using Configuration Manager as described in Configuring ZTI Task Sequence Steps in Configuration Manager
2. Add a new task sequence step based on the **Execute Runbook** task sequence type for:
   a. LTI on the **Task Sequence** tab (In the task sequence hierarchy, select **Add**, select **General**, and then select **Execute Runbook**.)
   b. ZTI in the task sequence hierarchy (Select **Add**, point to **MDT**, and then select **Execute Runbook**.)
3. On the **Properties** tab, configure the settings listed in Table 171 based on the requirements of your organization, and then select **OK**.

Table 171. Configuration Settings on the Properties Tab of the Execute Runbook Task Sequence Step Type

Expand table

Setting	Description
Name	Type a name for the task.

Setting	Description
Description	Type a description of the task—for example, ***runbook_name*** (where *runbook_name* is the name of the Orchestrator runbook that this task sequence step will run).

Orchestrator Server	Type the URL for the Orchestrator web service, which includes the server name. The Orchestrator web service can use either Hypertext Transfer Protocol (HTTP) or HTTP over Secure Sockets Layer (HTTPS). The Orchestrator web service defaults to port 81.
	
	The Orchestrator web service supports multiple runbook servers. By default, a runbook can run on any runbook server. A runbook can be configured to specify which runbook servers should be used to run the runbook.
	
	The Orchestrator web service supports the ability to run a runbook on a specific runbook server. This feature is not supported in MDT.
	
	Specify the URL in any of the following formats:
	
	- **servername**. When using this format, the URL defaults to:
	
	`https://<servername>:81/Orchestrator2012/Orchestrator.svc`
	
	- **servername:port**. When using this format, the URL defaults to:
	
	`https://<servername:port>/Orchestrator2012/Orchestrator.svc.`
	
	- **https://servername:port**. When using this format, the URL defaults to:
	
	`https://<servername:port>/Orchestrator2012/Orchestrator.svc.`
	
	- **https://servername:port**. When using this format, the URL defaults to:
	
	`https://<servername:port>/Orchestrator2012/Orchestrator.svc.`
	
	- **https://servername:port/Orchestrator2012/Orchestrator.svc**. When using this format, MDT assumes that you are providing the fully qualified URL, because the value ends with *.svc*.
	
	- **https://servername:port/Orchestrator2012/Orchestrator.svc**. When using this format, MDT assumes that you are providing the fully qualified URL, because the value ends with *.svc*.

Runbook	Select **Browse**, and then select the name of the Orchestrator runbook that this task sequence should run. To successfully browse for Orchestrator runbooks, install the ADO.NET

Setting	Description
	Data Services Update for .NET Framework 3.5 SP1 for Windows 7 and Windows Server 2008 R2.
Automatically provide runbook parameter values	Select this option to automatically provide the Orchestrator runbook input parameter values(which assumes that the runbook parameter values are task sequence variables). For example, if a runbook has an input parameter named **OSDComputerName**, then the **OSDComputerName** task sequence variable value is passed to the runbook. This option works only for input parameters that are valid task sequence variable names and do not contain spaces or other special characters. Although spaces and other special characters are supported as Orchestrator parameter names, they are not valid task sequence variable names. If you need to pass values to parameters with spaces or other special characters, use the **Specify explicit runbook parameters** option. The other option is **Specify explicit runbook parameters**. The values provided for the runbook input parameters to the Orchestrator web service are formatted as XML. Passing values that contain data that is or resembles XML-formatted data may cause errors.
Specify explicit runbook parameters	Select this option to explicitly provide the Orchestrator runbook input parameters. You must configure the following settings for each input parameter that the Orchestrator runbook requires: - **Name**. This is the name of the input runbook parameter. If you change the parameters for an existing Orchestrator runbook, you need to browse (reselect) for the runbook again, because MDT only retrieves the parameter list when initially adding the Orchestrator runbook. - **Value**. This can be a constant or a variable, such as a task sequence variable or an environment variable. For example, you can specify a value of **%OSDComputerName%**, which will pass the value of the **OSDComputerName** task sequence variable to the runbook input parameter.
Wait for the runbook to finish before continuing	This check box controls whether the task sequence step will wait for the runbook to finish before proceeding to the next task sequence step.If this check box is: - **Selected**, then the task sequence step will wait for the runbook to finish before proceeding on to the next task sequence step.

Setting	Description
	When this check box is selected, the task sequence step will poll the Orchestrator web service for the runbook to finish. The amount of time between polls starts at 1 second, then increases to 2, 4, 8, 16, 32, and 64 seconds between each poll. Once the amount of time reaches 64 seconds, the task sequence step continues to poll every 64 seconds. - **Cleared**, then the task sequence step will not wait for the runbook to finish before proceeding to the next task sequence step. This check box must be selected if the runbook returns output parameters.

If the Orchestrator runbook returns parameters, the values of the return parameters are set to corresponding task sequence variable names. If an Orchestrator runbook return parameter name contains spaces, the **ZTIExecuteRunbook.wsf** script will strip the spaces from the parameter name when creating the corresponding task sequence variable name.

> ⓘ **Note**
>
> If a runbook return parameter name contains other special characters, the return parameter may be ignored or generate errors.

For example, if a runbook return parameter has a name of **OSD Computer Name**, then the corresponding task sequence variable name **OSDComputerName** and the value in the return parameter will be saved in the **OSDComputerName** task sequence variable.

> ⓘ **Note**
>
> The **Wait for the runbook to finish before continuing** check box must be selected if the runbook returns output parameters.

Running Windows PowerShell Scripts During Deployment

MDT supports running Windows PowerShell scripts as a part of the deployment process. You can develop Windows PowerShell scripts to help automate the deployment process and then run those scripts within an MDT task sequence.

Run the Windows PowerShell scripts using a task sequence step created using the **Run PowerShell Script** task sequence step type. You can add a task sequence step based on

the **Run PowerShell Script** task sequence step type for task sequences in LTI, ZTI, or UDI.

> ⓘ **Note**
>
> For Configuration Manager task sequences, run the **Use Toolkit Package** task sequence step prior to running the **Run PowerShell Script** task sequence step.

To run a Windows PowerShell script in a task sequence

1. Create the Windows PowerShell script.

   For more information about creating Windows PowerShell scripts for use in MDT task sequences, see Create Windows PowerShell Scripts for Use in MDT.

2. Place the Windows PowerShell script in the Scripts subfolder in the:

   - Deployment share for LTI
   - MDT files package for ZTI and UDI

3. Add a new task sequence step to your task sequence based on the **Run PowerShell Script** task sequence step type.

4. In the newly create task sequence step, in the **PowerShell script** text box, ***script_name*** (where *script_name* is the fully qualified path to the Windows PowerShell script).

   If you specify:

   - Just the script name, then the assumption is that the script exists in the Scripts subfolder
   - A fully qualified path and script name, then ensure that the task sequence has access to the folder in which the script is stored (For example, if the script is stored on a network shared folder, ensure that there is an existing connection to that server prior to running the script.)

Applying Group Policy Object Packs

Deploying operating systems and applications so that they are compliant with security and regulatory standards is an essential part of any deployment effort. MDT allows you to apply security and compliance configuration templates to the operating system and applications after they are deployed using Group Policy object (GPO) packs.

GPO packs are created by exporting a GPO backup in the Microsoft Security Compliance Manager. These GPO packs are applied by the **Apply Local GPO Package** task sequence step for task sequences created using the MDT task sequence templates. The **Apply Local GPO Package** task sequence step runs the ZTIApplyGPOPack.wsf script, which is responsible for applying the GPO packs to the target computer.

> ⓘ **Note**
>
> GPO packs are only used to configure security and compliance configuration settings for Windows operating systems, not the applications running on the operating system. For example, the Internet Explorer or Microsoft Office security and compliance configuration settings in Security Compliance Manager cannot be used as GPO packs.

The following MDT task sequence templates include the**Apply Local GPO Package** task sequence step:

- **Standard Client Task Sequence** in LTI in ZTI
- **Standard Server Task Sequence** in LTI and ZTI
- **Deploy to VHD Client Task Sequence** in LTI
- **Deploy to VHD Server Task Sequence** in LTI

> ⓘ **Note**
>
> Applying GPO packs affects system behavior and features because of the increased security requirements that GPO packs could configure. The result is that you may lose certain functionality after a GPO pack is applied.

If the security configuration settings that the Security Compliance Manager GPO packs provide are too stringent, perform one of the following tasks:

- Modify the existing GPO templates to be less restrictive.
- Provide a custom GPO template that you have created that is less restrictive.
- Disable the **Apply Local GPO Package** task sequence step in your task sequence.

  For example, the GPO pack for Windows 7 can enforce Server Message Block (SMB) configuration settings that could prevent Windows 7 from communicating with other devices running Common Internet File System (CIFS) or SAMBA.

Apply GPO packs templates by performing the following steps:

1. Identify or create the GPO packs required by your organization as described in Identify or Create the GPO Packs.
2. Place the GPO packs in the appropriate MDT folders as described in Place the GPO Packs in the Appropriate MDT Folders.
3. Configure MDT to deploy the GPO packs as described in Configure MDT to Deploy the GPO Packs.

Identify or Create the GPO Packs

You can use GPO packs that are:

- **Generated from the Security Compliance Manager**. The Security Compliance Manager can export a GPO backup that you can use as a GPO pack. You can copy these GPO packs to the MDT files folder and apply them during the deployment process.
- **Customized by you**. You can create your own customized GPO packs based on your organization's requirements. You can use the security and compliance configuration settings in Security Compliance Manager as a beginning, and then customize those settings for your organization. Then, you can export the security and compliance configurations settings as a GPO backup and subsequently a GPO pack.

Place the GPO Packs in the Appropriate MDT Folders

After you have identified or created the GPO packs that your organization requires, place the GPO packs in subfolder in the Templates\GPOPacksfolder in the:

- Deployment share for LTI
- MDT files package for ZTI and UDI

Configure MDT to Deploy the GPO Packs

The **Apply Local GPO Package** task sequence step can be configured using the properties listed in Table 172. These properties can be configured using the CustomSettings.ini file or the MDT DB.

Table 172. Properties Used to Configure the Apply Local GPO Package Task Sequence Step

Expand table

Property	Description
ApplyGPOPack	This property is used to determine whether the **Apply Local GPO Package** task sequence step is performed. If the value is set to: - **YES**, then the task sequence step is performed - **NO**, then the task sequence step is not performed The default value is **YES**. The **ApplyGPOPack** property can be set in the CustomSettings.ini file or in the MDT DB. For more information on configuring the **ApplyGPOPack** property, see the **ApplyGPOPack** property in the MDT document *Toolkit Reference*.
GPOPackPath	This property is used to override the default path to the root folder for all the GPO packs. The path specified in this property is relative to the Templates\GPOPacks folder.The default path to the root folder for all the GPO packs is the Templates\GPOPacks folder. For more information on configuring the **GPOPackPath** property, see the **GPOPackPath** property in the MDT document *Toolkit Reference*.

> **Note**
>
> The appropriate GPO pack is selected based on the operating system being deployed. If no matching GPO pack can be found, then no GPO pack will be applied.

Enabling Participation in CEIP and WER

MDT includes a new task sequence step that automates the configuration of participation in Windows Customer Experience Improvement Program (CEIP) and Windows Error Reporting (WER). The **Opt In to CEIP and WER** task sequence step is used to automate this participation.

> **Note**

Although you can use MDT task sequences to enable CEIP and WER for Windows operating systems (only when the corresponding task sequence step is enabled), this is separate from the CEIP information gathered when you opt in to the MDT CEIP program. For more information, see the Microsoft Privacy Statement.

The **Opt In to CEIP and WER** task sequence step is included in the following MDT task sequence templates but is disabled by default:

- **Standard Client Task Sequence** in LTI and ZTI
- **Standard Server Task Sequence** in LTI and ZTI
- **Deploy to VHD Client Task Sequence** in LTI
- **Deploy to VHD Server Task Sequence** in LTI
- **User Driven Installation Task Sequence** in UDI

  The **Opt In to CEIP and WER** task sequence step runs the ZTIOptIn.wsf script as shown in the following example:

```
Windows Command Prompt

cscript.exe %SCRIPTROOT%\ZTIOptIn.wsf /CEIP:YES /WER:YES
```

The ZTIOptIn.wsf script has the following named parameters:

- **CEIP**. This parameter determines the participation in CEIP and can be one of the following:
  - **YES**. This value specifies to enable participation.
  - **NO**. This value specifies to not enable participation.

  > **Note**
  >
  > Any value other than **YES** is treated as **NO**, including not providing the parameter.

- **WER**. This parameter determines the participation in WER and can be one of the following:
  - **YES**. This value specifies to enable participation.
  - **NO**. This value specifies to not enable participation.

> ⓘ Note
>
> Any value other than **YES** is treated as **NO**, including not providing the parameter.

Configuring Roles and Features Task Sequence Steps

MDT automates the installation and uninstallation of Windows roles and features using the **Install Roles and Features** and **Uninstall Roles and Features** task sequence step types. These task sequence types allow organizations to deploy the target computers with the Windows roles and features that comply with configuration standards defined by organizational or regulatory authorities.

Configure the roles and features task sequence steps for LTI and ZTI by:

- Installing the appropriate Windows roles and features as described in Configure Install Roles and Features Task Sequence Steps
- Uninstalling the appropriate Windows roles and features as described in Configure Uninstall Roles and Features Task Sequence Steps

Configure Install Roles and Features Task Sequence Steps

MDT automates the deployment of Windows roles and features using the Install Roles and Features task sequence step type. This task sequence step must be run in the target operating system, not in Windows PE.

> ⓘ Note
>
> For ZTI task sequences that are not created using the MDT task sequence templates, ensure that you run the **Use Toolkit Package** and **Gather** task sequence step prior to running the **Install Roles and Features** task sequence step. The **Install Roles and Features** task sequence step depends on the **Use Toolkit Package** and **Gather** task sequence steps.

To configure the Install Roles and Features task sequence step type to install Windows roles and features

1. Edit ***task_sequence_name*** (where *task_sequence_name* is the name of the task sequence to which you want to add the task sequence step) for:

- LTI as described in Configure the Task Sequence Properties Task Sequence Tab
- ZTI using Configuration Manager as described in Configuring ZTI Task Sequence Steps in Configuration Manager

2. Add a new task sequence step based on the **Install Roles and Features** task sequence type for:
   - LTI on the **Task Sequence** tab (In the task sequence hierarchy, select **Add**, point to **Roles**, and then select **Install Roles and Features**.)
   - ZTI in the task sequence hierarchy (Select **Add**, point to **MDT**, and then select **Install Roles and Features**.)
3. On the **Properties** tab, configure the settings listed in Table 173 based on the requirements of your organization, and then select **OK**.

Table 173. Configuration Settings on the Properties Tab of the Install Roles and Features Task Sequence Step Type

Expand table

Setting	Description
Name	Type a name for the task.
Description	Type a description of the task.
Select the operating system for which roles are to be installed	Select the target operating system to be deployed from the following list: - Windows 7 - Windows 8 - Windows 8.1 - Windows Server 2008 R2 - Windows Server 2008 R2 Core - Windows Server 2012 - Windows Server 2012 Core

Setting	Description
	- Windows Server 2012 R2 - Windows Server 2012 R2 Core
Select the roles and features that should be installed	Select the check box next to the roles or features to be installed. You can select **Select All** to select all the roles and features, or you can select **Select None** to clear all the roles and features.

For information on how to uninstall Windows roles and features, see Configure Uninstall Roles and Features Task Sequence Steps.

Configure Uninstall Roles and Features Task Sequence Steps

MDT automates the removal (uninstallation) of operating system roles and features in Windows using the **Uninstall Roles and Features** task sequence step. This task sequence step must be run in the target operating system, not in Windows PE.

For ZTI task sequences that are not created using the MDT task sequence templates, ensure that you run the **Use Toolkit Package** and **Gather** task sequence steps prior to running the **Install Roles and Features** task sequence step. The **Install Roles and Features** task sequence step depends on the **Use Toolkit Package** and **Gather** task sequence steps.

> ⓘ **Note**
>
> Before uninstalling roles or features, remove all dependent roles or features.

To configure the Uninstall Roles and Features task sequence step type to install Windows roles and features

1. Edit ***task_sequence_name*** (where *task_sequence_name* is the name of the task sequence to which you want to add the task sequence step) for:

   - LTI as described in Configure the Task Sequence Properties Task Sequence Tab
   - ZTI using Configuration Manager as described in Configuring ZTI Task Sequence Steps in Configuration Manager

2. Add a new task sequence step based on the **Uninstall Roles and Features** task sequence type for:

   - LTI on the **Task Sequence** tab (In the task sequence hierarchy, select **Add**, point to **Roles**, and then select **Uninstall Roles and Features**.)
   - ZTI in the task sequence hierarchy (Select **Add**, point to **MDT**, and then select **Uninstall Roles and Features**.)

3. On the **Properties** tab, configure the settings listed in Table 174 based on the requirements of your organization, and then select **OK**.

Table 174. Configuration Settings on the Properties Tab of the Uninstall Roles and Features Task Sequence Step Type

Expand table

Setting	Description
Name	Type a name for the task.
Description	Type a description of the task.
Select the operating system for which roles are to be uninstalled	Select the target operating system to be deployed from the following list: - Windows 7 - Windows 8 - Windows 8.1 - Windows Server 2008 R2 - Windows Server 2008 R2 Core - Windows Server 2012 - Windows Server 2012 Core - Windows Server 2012 R2 - Windows Server 2012 R2 Core
Select the roles and features that should be uninstalled	Select the check box next to the roles or features to be uninstalled.

Setting	Description
	You can select **Select All** to select all the roles and features or select **Select None** to clear all the roles and features.

For information on how to install Windows roles and features, see Configure Install Roles and Features Task Sequence Steps.

Configuring Server Role Task Sequence Steps

MDT automates the deployment of server roles in Windows Server. Configure task sequence steps in MDT to deploy the server roles that are supported in MDT.

> ⓘ **Note**
>
> For ZTI task sequences that are not created using the MDT task sequence templates, ensure that you run the **Use Toolkit Package** and **Gather** task sequence steps prior to running any of the server role task sequence steps. The server role task sequence steps are dependent on the **Use Toolkit Package** and **Gather** task sequence steps.

Configure the Windows Server server role task sequence steps for MDT deployments by:

- Configuring the AD DS server role task sequence step as described in Configure AD DS Server Role Task Sequence Step Settings
- Configuring the DNS Server server role task sequence step as described in Configure DNS Server Role Settings
- Configuring the DHCP Server server role task sequence step as described in Configure DHCP Server Role Task Sequence Step Settings
- Configuring the Authorize DHCP task sequence step as described in Configure Authorize DHCP Task Sequence Step Settings

Configure AD DS Server Role Task Sequence Step Settings

AD DS stores directory data and manages communications between users and domains, including logon processes, authentication, and directory searches. An AD DS domain controller is a server that runs AD DS.

> **Note**
>
> For ZTI task sequences in Configuration Manager that are not created using the MDT task sequence templates, ensure that you run the **Use Toolkit Package** and **Gather** task sequence steps prior to running any of the server role task sequence steps. The server role task sequence steps are dependent on the **Use Toolkit Package** and **Gather** task sequence steps.

Configure the **AD DS Server Role** task sequence step by:

- Deploying a new domain controller in a new forest as described in Deploy a Domain Controller in a New Forest
- Deploying a new domain controller as a replica in an existing domain as described in Deploy a New Domain Controller as a Replica in an Existing Domain
- Deploying a new domain controller in a new domain tree in an existing forest as described in Deploy a New Domain Controller in a New Domain Tree in an Existing Forest
- Deploying a new domain controller in a new domain in an existing forest as described in Deploy a New Domain Controller in a New Domain in an Existing Forest
- Deploying a new read-only domain controller (RODC) in an existing domain as described in Deploy an RODC in an Existing Domain
- Configuring AD DS advanced properties for domain controllers as described in Configure AD DS Advanced Properties

Deploy a Domain Controller in a New Forest

Using this option, deploy a domain controller that contains a new forest environment. Use this option when deploying a new forest environment.

To deploy a domain controller with a new forest

1. Edit ***task_sequence_name*** (where *task_sequence_name* is the name of the task sequence to which you want to add the task sequence step) for:
    - LTI as described in Configure the Task Sequence Properties Task Sequence Tab

- ZTI using Configuration Manager as described in Configuring ZTI Task Sequence Steps in Configuration Manager

2. Add a new task sequence step based on the **Configure ADDS** task sequence type for:

   - LTI on the **Task Sequence** tab (In the task sequence hierarchy, select **Add**, select **Roles**, and then select **Configure ADDS**.)

   - ZTI in the task sequence hierarchy (Select **Add**, select **MDT**, and then select **Configure ADDS**.)

3. On the **Properties** tab, type the relevant information in the following boxes:

   - **Name**. Type a name for the task.

   - **Description**. Type a description of the task—for example, ***Server_Name Site_Name*** (where *Server_Name* is the name of the server and *Site_Name* is the name of the domain).

4. In the **Create** box, select **New Forest**.

5. In the **New forest domain DNS name** box, type a name for the new domain—for example, **woodgrove.com**.

   You must include the extension of the domain—for example, .com, .net, .int.

6. In the **NetBIOS name** box, type a name for the NetBIOS.

   This name is usually the domain name without .com or any other type of extension. For example, the forest domain name woodgrove.com might have the NetBIOS name WOODGROVE.

7. In the **Recovery (safe mode) password** box, type a password to use for safe mode recovery.

   You use this password to recover from a failed AD DS service. Make note of this password in case AD DS must be recovered.

8. In the **Advanced Properties** section, complete the task configuration as described in Configure AD DS Advanced Properties, and then select **OK**.

   For more information about **DCPROMO** command-line options, see Dcpromo.

Deploy a New Domain Controller as a Replica in an Existing Domain

Using this option, deploy an existing domain controller as a new domain controller by replicating it into an existing environment. Use this option when deploying a new domain controller into an existing environment if replication will obtain the existing domain information from AD DS.

To deploy a domain controller as a new domain controller replica

1. Edit ***task_sequence_name*** (where *task_sequence_name* is the name of the task sequence to which you want to add the task sequence step) for:

    - LTI as described in Configure the Task Sequence Properties Task Sequence Tab
    - ZTI using Configuration Manager as described in Configuring ZTI Task Sequence Steps in Configuration Manager

2. Add a new task sequence step based on the **Configure ADDS** task sequence type for:

    - LTI on the **Task Sequence** tab (In the task sequence hierarchy, select **Add**, select **Roles**, and then select **Configure ADDS**.)
    - ZTI in the task sequence hierarchy (Select **Add**, select **MDT**, and then select **Configure ADDS**.)

3. On the **Properties** tab, type the relevant information in the following boxes:

    - **Name**. Type a name for the task.
    - **Description**. Type a description of the task—for example, ***Server_Name Site_Name*** (where *Server_Name* is the name of the server, and *Site_Name* is the name of the domain).

4. In the **Create** box, select **New domain controller replica**.

5. In the **Existing domain DNS name** box, type the FQDN of an existing domain controller.

6. In the **Replication source domain controller** box, type the name of a domain controller the new server will replicate within the existing environment. The directory services database replicates this domain controller.

7. In the **Account** box, type the name of an account that has permissions to add a domain controller to the existing network (typically, a domain Administrator account), and then select **Set**.

8. In the **Recovery (safe mode) password** box, type a password to use for safe mode recovery.

   This password is used to recover from a failed AD DS service. Make note of this password in case AD DS must be recovered.

9. In the **Advanced Properties** section, complete the task configuration as described in Configure AD DS Advanced Properties, and then select **OK**.

   For more information about **DCPROMO** command-line options, see Dcpromo.

Deploy a New Domain Controller in a New Domain Tree in an Existing Forest

Using this option, deploy a domain controller that contains a new tree into an existing forest environment. Use this option when deploying a child domain into an existing forest environment.

To deploy a domain controller with a new domain tree in an existing forest

1. Edit ***task_sequence_name*** (where *task_sequence_name* is the name of the task sequence to which you want to add the task sequence step) for:

   - LTI as described in Configure the Task Sequence Properties Task Sequence Tab
   - ZTI using Configuration Manager as described in Configuring ZTI Task Sequence Steps in Configuration Manager

2. Add a new task sequence step based on the **Configure ADDS** task sequence type for:

   - LTI on the **Task Sequence** tab (In the task sequence hierarchy, select **Add**, select **Roles**, and then select **Configure ADDS**.)
   - ZTI in the task sequence hierarchy (Select **Add**, select **MDT**, and then select **Configure ADDS**.)

3. On the **Properties** tab, type the relevant information in the following boxes:

   - **Name**: Type a name for the task.
   - **Description**. Type a description of the task—for example, ***Server_Name Site_Name*** (where *Server_Name* is the name of the server and *Site_Name* is

the name of the domain).

4. In the **Create box**, select **New domain in existing forest**.
5. In the **Existing forest (parent) domain DNS name** box, type the name of an existing domain in the network.
6. In the **NetBIOS name** box, type the NetBIOS name of an existing domain in the network, usually the domain name without .com or any other type of extension—for example, the domain woodgrove.com might have the NetBIOS name WOODGROVE.
7. In the **New domain tree (child) DNS name** box, type a name for the child domain being created—for example, ***child.woodgrove.com*** (where *child* is the name of the child domain).
8. In the **Replication source domain controller** box, type the name of the domain controller to which the new child domain will replicate.
9. In the **Account** box, type the name of an account with permissions to add a domain controller to the existing network (typically, a domain Administrator account), and then select **Set**.
10. In the **Recovery (safe mode) password** box, type a password to use for safe mode recovery.

    You use this password to recover from a failed AD DS service. Make note of this password in case AD DS must be recovered.
11. In the **Advanced Properties** section, complete the task configuration as described in Configure AD DS Advanced Properties, and then select **OK**.

    For more information about **DCPROMO** command-line options, see Dcpromo.

Deploy a New Domain Controller in a New Domain in an Existing Forest

Using this option, deploy a domain controller that contains a new domain into an existing forest environment. Use this option when deploying a new child domain into an existing forest environment.

To deploy a domain controller with a new domain in an existing forest

1. Edit ***task_sequence_name*** (where *task_sequence_name* is the name of the task sequence to which you want to add the task sequence step) for:

    - LTI as described in Configure the Task Sequence Properties Task Sequence Tab
    - ZTI using Configuration Manager as described in Configuring ZTI Task Sequence Steps in Configuration Manager

2. Add a new task sequence step based on the **Configure ADDS** task sequence type for:

    - LTI on the **Task Sequence** tab (In the task sequence hierarchy, select **Add**, select **Roles**, and then select **Configure ADDS**.)
    - ZTI in the task sequence hierarchy (Select **Add**, select **MDT**, and then select **Configure ADDS**.)

3. On the **Properties** tab, type the relevant information in the following boxes:

    - **Name**. Type a name for the task.
    - **Description**. Type a description of the task—for example, ***Server_Name Site_Name*** (where *Server_Name* is the name of the server and *Site_Name* is the name of the domain).

4. In the **Create box**, select **New domain in existing forest**.

5. In the **Existing forest (parent) domain DNS name** box, type the name of an existing domain in the network.

6. In the **NetBIOS name** box, type the NetBIOS name of an existing domain in the network, usually the domain name without .com or any other type of extension—for example, the domain woodgrove.com might have the NetBIOS name WOODGROVE.

7. In the **New domain (child) DNS name** box, type a name for the child domain being created—for example, ***child.woodgrove.com*** (where *child* is the name of the child domain).

8. In the **Replication source domain controller** box, type the name of the domain controller to which the new child domain will be replicated.

9. In the **Account** box, type the name of an account with permissions to add a domain controller to the existing network (typically, a domain Administrator account), and then select **Set**.

10. In the **Recovery (safe mode) password** box, type a password to use for safe mode recovery.

    You use this password to recover from a failed AD DS service. Make note of this password in case AD DS must be recovered.

11. In the **Advanced Properties** section, complete the task configuration as described in Configure AD DS Advanced Properties, and then select **OK**.

    For more information about **DCPROMO** command-line options, see Dcpromo.

Deploy an RODC in an Existing Domain

Using this option, deploy a domain controller that contains a read-only replica of the existing domain into an existing forest environment. Use this option to deploy a domain controller that contains an un-editable replica of a domain structure into an existing forest environment.

To deploy an RODC in an existing domain

1. Edit ***task_sequence_name*** (where *task_sequence_name* is the name of the task sequence to which you want to add the task sequence step) for:

    - LTI as described in Configure the Task Sequence Properties Task Sequence Tab
    - ZTI using Configuration Manager as described in Configuring ZTI Task Sequence Steps in Configuration Manager

2. Add a new task sequence step based on the **Configure ADDS** task sequence type for:

    - LTI on the **Task Sequence** tab (In the task sequence hierarchy, select **Add**, select **Roles**, and then select **Configure ADDS**.)
    - ZTI in the task sequence hierarchy (Select **Add**, select **MDT**, and then select **Configure ADDS**.)

3. On the **Properties** tab, type the relevant information in the following boxes:

    - **Name**. Type a name for the task.
    - **Description**. Type a description of the task; for example, ***Server_Name Site_Name*** (where *Server_Name* is the name of the server and *Site_Name* is the name of the domain).

4. In the **Create** box, select **New read-only domain controller (RODC) replica**.
5. In the **Existing domain DNS name** box, type the name of an existing DNS server.
6. In the **Replication source domain controller** box, type the name of the domain controller to be replicated within the existing environment. The directory services database replicates this domain controller.
7. In the **Account** box, type the name of an account with permissions to add a domain controller to the existing network (typically, a domain Administrator account), and then select **Set**.
8. In the **Recovery (safe mode) password** box, type a password to use for safe mode recovery.

   You use this password to recover from a failed AD DS service. Make note of this password in case AD DS must be recovered.
9. In the **Advanced Properties** section, complete the task configuration as described in Configure AD DS Advanced Properties, and then select **OK**.

   For more information about **DCPROMO** command-line options, go to Dcpromo.

Configure AD DS Advanced Properties

To configure AD DS advanced properties, perform the following steps:

1. Edit ***task_sequence_name*** (where *task_sequence_name* is the name of the task sequence to which you want to add the task sequence step) for:
   - LTI as described in Configure the Task Sequence Properties Task Sequence Tab
   - ZTI using Configuration Manager as described in Configuring ZTI Task Sequence Steps in Configuration Manager
2. Modify the **Configure ADDS** task sequence step you created for:
   - LTI on the **Task Sequence** tab (In the task sequence hierarchy, select ***task_sequence_step*** [where *task_sequence_step* is the **Configure ADDS** task sequence step].)
   - ZTI in the task sequence hierarchy (Select **Add**, select **MDT**, and then select ***task_sequence_step*** [where *task_sequence_step* is the **Configure ADDS** task sequence step].)

3. On the **Properties** tab, select **Advanced**.

4. In the **AD DS Advanced Properties** dialog box, in the **Options**, **Functional Levels**, and **Folders** sections, select the following options as required for your environment and AD DS scenario:

   Options section:

   - **Install DNS if not already present**. Select this option when creating a new forest or new domain.
   - **Make this domain controller a global catalog (GC) server**. This is the default option and should be selected for new domains or forests and domains without a GC server.
   - **Wait for critical replication only**. Select this option to populate only the directory services database using replication.

   Functional levels section:

   - **Forest Functional Level**. Windows Server 2003, Windows Server 2008 (2 = Windows Server 2003; 3 = Windows Server 2008; 4 = Windows Server 2008 R2)
   - **Domain Functional Level**. Windows Server 2003, Windows Server 2008 (2 = Windows Server 2003; 3 = Windows Server 2008; 4 = Windows Server 2008 R2)

   Folders section:

   - **Database**. Contains the fully qualified path on the target computer to the location for the NTDS database. The default value is %SYSTEMROOT%NTDS. You can also configure this value using the **DatabasePath** property in the CustomSettings.ini file or the MDT DB.
   - **Log Files**. Contains the fully qualified path on the target computer to the location for the log files. The default value is %SYSTEMROOT%NTDS. You can also configure this value using the **LogPath** property in the CustomSettings.ini file or the MDT DB.
   - **SYSVOL**. Contains the fully qualified path on the target computer to the location for the SYSVOL folder. The default value is %SYSTEMROOT%SYSVOL. You can also configure this value using the **SysVolPath** property in the CustomSettings.ini file or the MDT DB.

> ⓘ **Note**
>
> If you configure values in the CustomSettings.ini or the MDT DB, use the %DESTINATIONLOGICALDRIVE% task sequence variable instead of the %SYSTEMROOT% variable. For LTI deployments, the variables are evaluated while running Windows PE, so the %SYSTEMROOT% variable will return the SystemRoot folder for Windows PE, not the target operating system. The %SYSTEMROOT% variable for Windows PE typically is set to X:\WINDOWS.

5. In the **Site name** box, type the name of the site in which to install the domain controller.

   The default name for a new forest or site is *default_first_site*; however, this value does not appear in the **Site name** box by default—you must type it. Then, select **OK** to complete the configuration of the **AD DS Advanced Properties** dialog box, and then select **OK** in the ***Task Name*** **Properties** dialog box to complete configuration of the task.

Configure DNS Server Role Settings

Using this option, configure and deploy the DNS server role to a new computer or a DNS server operating on an existing computer. By assigning the DNS server role, you can configure standard DNS primary, secondary, and stub zones as well as AD DS-integrated primary and stub zones. There is also an option to manage aging, updates, types, and multiple zones—all in an automated process. This is not a migration process from an existing DNS server; rather, it is a new installation of DNS zones of all types.

> ⓘ **Note**
>
> For ZTI task sequences using Configuration Manager that are not created using the MDT task sequence templates, ensure that you run the **Use Toolkit Package** and **Gather** task sequence steps prior to running any of the server role task sequence steps. The server role task sequence steps are dependent on the **Use Toolkit Package** and **Gather** task sequence steps.

To configure and deploy the DNS server role

1. Edit ***task_sequence_name*** (where *task_sequence_name* is the name of the task sequence to which you want to add the task sequence step) for:

- LTI as described in Configure the Task Sequence Properties Task Sequence Tab
- ZTI using Configuration Manager as described in Configuring ZTI Task Sequence Steps in Configuration Manager

2. Add a new task sequence step based on the **Configure DNS** task sequence type for:
    - LTI on the **Task Sequence** tab (In the task sequence hierarchy, select **Add**, select **Roles**, and then select **Configure DNS**.)
    - ZTI in the task sequence hierarchy (Select **Add**, select **MDT**, and then select **Configure DNS**.)
3. On the **Properties** tab, in **Name**, type **name** (where *name* is the name by which the task is identified in the task sequence).
4. In **Description**, type ***description*** (where *description* is the description of the task and its role in the task sequence).
5. In the **Zones** section, select the yellow**Add** button.
6. In the **DNS Zone Properties** dialog box, in **DNS zone name**, type a name for the zone (for example, **woodgrove.com**).
7. In **Type**, select **Change**.
8. In the **Change Zone Type** dialog box, select one of the following zone types:
    - Primary zone
    - Secondary zone
    - Stub zone
9. In the **Change Zone Type** dialog box, select the **Store the zone in Active Directory** check box if DNS will be installed on a domain controller, and then select **OK**.
10. In the**DNS Zone Properties** dialog box, in **Dynamic updates**, select one of the following:
    - **None**
    - **Nonsecure and Secure**

11. Select the **Scavenge stale resource records** check box to enable this feature, and then select **OK**.
12. On the **Properties** tab of the **Configure DNS** step, select **Server Properties**.
13. In the **Server Properties** dialog box, in **Server Options**, select the appropriate server options to enable.
14. In the **Server Properties** dialog box, in **Name checking**, select the appropriate name-checking options to enforce, and then select **OK**.
15. In the ***task_sequence*** **Properties** dialog box (where *task_sequence* is the name of the task sequence being edited), select **OK**.
16. Close the Deployment Workbench.

Configure DHCP Server Role Task Sequence Step Settings

Using this option, configure and deploy the DHCP server role using MDT. You can configure all the standard DHCP scope options similar to using the standard DHCP console in Windows Server. To implement the DHCP server role, configure the **Authorize DHCP** task sequence in conjunction with the **Configure DHCP Server** task sequence.

> ⓘ **Note**
>
> For ZTI task sequences using Configuration Manager that are not created using the MDT task sequence templates, ensure that you run the **Use Toolkit Package** and **Gather** task sequence steps prior to running any of the server role task sequence steps. The server role task sequence steps are dependent on the **Use Toolkit Package** and **Gather** task sequence steps.

Configure the **Configure DHCP Server** task sequence step settings by:

- Configuring the deployment of the DHCP server role as described in Configure Deployment of the DHCP Server Role
- Configuring the DHCP scopes for the DHCP server role as described in Configure DHCP Scopes for the DHCP Server Role
- Configuring the DHCP server options for the DHCP server role as described in Configure the DHCP Server Options for the DHCP Server Role

Configure Deployment of the DHCP Server Role

Install and configure the DHCP Server role on the target computer by modifying the **Configure DHCP Server** task sequence step type.

To configure and deploy the DHCP server role

1. Edit ***task_sequence_name*** (where *task_sequence_name* is the name of the task sequence to which you want to add the task sequence step) for:

    - LTI as described in Configure the Task Sequence Properties Task Sequence Tab
    - ZTI using Configuration Manager as described in Configuring ZTI Task Sequence Steps in Configuration Manager

2. Add a new task sequence step based on the **Configure DHCP Server** task sequence type for:

    - LTI on the **Task Sequence** tab (In the task sequence hierarchy, select **Add**, select **Roles**, and then select **Configure DHCP**.)
    - ZTI in the task sequence hierarchy (Select **Add**, select **MDT**, and then select **Configure DHCP**.)

3. On the **Properties** tab, type the relevant information in the following boxes:

    - **Name**. The name by which the task will be identified in the task sequence list.
    - **Description**. A description of the task (for example, DHCP scope information, scope names).
    - **Scope details**. Details about the IP address the DHCP scopes configured.
    - **Server options**. The options passed to DHCP clients (for example, default gateway, DNS server, and WINS server addresses).

4. Select **OK**.

Configure DHCP Scopes for the DHCP Server Role

Using this option, configure the DHCP scopes that contain the rules and active scopes used on the DHCP server. For more information about DHCP scope configuration options and for guidance on using each configuration option, see Chapter 6 - Dynamic Host Configuration Protocol in TCP/IP Fundamentals for Microsoft Windows.

To configure and deploy DHCP scopes

1. Edit ***task_sequence_name*** (where *task_sequence_name* is the name of the task sequence to which you want to add the task sequence step) for:

    - LTI as described in Configure the Task Sequence Properties Task Sequence Tab
    - ZTI using Configuration Manager as described in Configuring ZTI Task Sequence Steps in Configuration Manager

2. Modify the task sequence step you created based on the **Configure DHCP** task sequence type for:

    - LTI on the **Task Sequence** tab (In the task sequence hierarchy, select ***task_sequence_step*** [where *task_sequence_step* is the name of the task sequence step].)
    - ZTI in the task sequence hierarchy (Select ***task_sequence_step*** [where *task_sequence_step* is the name of the task sequence step].)

3. On the **Properties** tab, select the yellow **Add scope** button.

4. In the **Scope Properties** dialog box, configure the following options as required for the environment:

    - **Scope name**. The name used to refer to the scope.
    - **Start IP address**. The beginning address of the scope (for example, 192.168.0.150).
    - **End IP address**. The ending address of the scope (for example, 192.168.0.250).
    - **Subnet mask**. The mask used for the IP address scope (for example, 255.255.255.0).
    - **Scope IP address**. The address of the scope itself (for example, 192.168.0.1).
    - **Lease duration for DHCP clients**. The maximum time a client can keep the IP address that the DHCP server assigns.
    - **Description**. A description of the scope (for administrative reference).

5. On the **Advanced** tab, in the **Exclude IP Address Range** section, type the following information to exclude addresses for the scope created on the **General** tab:

- **Start IP address**. The beginning address for exclusion from a scope (for example, 192.168.0.251).
- **End IP address**. The ending address for exclusion from a scope (for example, 192.168.0.255).

6. On the **Options** tab, configure the following options for the scope created on the **General** tab:
   - **003 Router**. The default gateway given to DHCP clients.
   - **006 DNS Servers**. The DNS server address given to DHCP clients.
   - **015 DNS Domain Name**. The DNS domain name given to clients (for example, woodgove.com).
   - **044 WINS/NBNS Servers**. The WINS server IP address (for example, 192.168.0.2).
   - **046 WINS/NBT Node Type**. The WINS node type.
   - **060 PXE Client**. The address used for PXE client Bootstrap code.
7. Select **OK**.

Configure the DHCP Server Options for the DHCP Server Role

Using this option, configure the DHCP server options given to DHCP clients, including router or default gateway designation, DNS server IP information, and WINS server information.

To configure and deploy DHCP server options

1. Edit ***task_sequence_name*** (where *task_sequence_name* is the name of the task sequence to which you want to add the task sequence step) for:
   - LTI as described in Configure the Task Sequence Properties Task Sequence Tab
   - ZTI using Configuration Manager as described in Configuring ZTI Task Sequence Steps in Configuration Manager
2. Modify the task sequence step you created based on the **Configure DHCP** task sequence type for:

- LTI on the **Task Sequence** tab (In the task sequence hierarchy, select ***task_sequence_step*** [where *task_sequence_step* is the name of the task sequence step].)
- ZTI in the task sequence hierarchy (Select ***task_sequence_step*** [where *task_sequence_step* is the name of the task sequence step].)

3. On the **Properties** tab, in the **Server Options** section, select **Configure**, and then configure the following options as required for your environment.
4. Select **003 Router**, and then perform the following steps:
   a. In the **Server Name** box, type the IP address and resolve the name.
   b. Select **Add** to type an IP address.
   c. Select an IP address, and then select **Remove** to remove the highlighted IP address.
5. Select **006 DNS Servers**, and then perform the following tasks:
   a. In the **Server Name** box, type the IP address and resolve the name.
   b. Select **Add** to type an IP address.
   c. Select an IP address, and then select **Remove** to remove the highlighted IP address.
6. Select **015 DNS Domain Name**, and then, in the **String Value** box, type the domain name (for example, woodgrove.com).
7. Select **044 WINS/NBNS Servers**, and then perform the following tasks:
   a. In the **Server Name** box, type the IP address and resolve the name.
   b. Select **Add** to type an IP address.
   c. Select an IP address, and then select **Remove** to remove the highlighted IP address.
8. Select **046 WINS/NBT Node Type**, and then type one of the following codes: **44**, **46**, or **47**.

   For more information about how to determine the correct option for the environment, see Managing DHCP Options.

9. Select **060 PXE Client**; then, in the **String Value** box, type the PXE client string (typically, PXEClient).

Configure Authorize DHCP Task Sequence Step Settings

Authorizing the DHCP service within AD DS is imperative to successfully deploying and using DHCP services within a Windows-based network.

> ⓘ **Note**
>
> The ZTI task sequence templates that are provided with MDT do not include an **Authorize DHCP** task sequence step type. You must manually add this step type if you want to automatically authorize the DHCP server as a part of the ZTI deployment.

To authorize the DHCP server role in AD DS

1. Edit ***task_sequence_name*** (where *task_sequence_name* is the name of the task sequence to which you want to add the task sequence step) for:
    - LTI as described in Configure the Task Sequence Properties Task Sequence Tab
    - ZTI using Configuration Manager as described in Configuring ZTI Task Sequence Steps in Configuration Manager
2. Add a new task sequence step based on the **Authorize DHCP** task sequence type for:
    - LTI on the **Task Sequence** tab (In the task sequence hierarchy, select **Add**, select **Roles**, and then select **Authorize DHCP**.)
    - ZTI in the task sequence hierarchy (Select **Add**, select **MDT**, and then select **Authorize DHCP**.)
3. On the **Properties** tab, type the relevant information in the following boxes:
    - **Name**. The name by which the task appears in the task sequence list.
    - **Description**. A description of the task.
4. In the **Account** box, type the name of an account with permissions to authorize the DHCP service in AD DS. Select **Set**, and then type the following information in the

relevant boxes:

- **Username**. The account that can authorize DHCP, in the format domain\user
- **Password**. The password for the account
- **Confirm Password**. Retype the password

5. Select **OK**, then select **OK** again.

Copying Content to the Target Computer

To copy content to target computers, perform any combination of the following steps:

- Copy content to the target computer using a task sequence step as described in Copy Content to Target Computers Using Task Sequence Steps.
- Copy content to the target computer using OEM folders as described Copy Content to Target Computers Using OEM Folders.

Copy Content to Target Computers Using Task Sequence Steps

Create a task sequence based on the **Run Command Line** task sequence step type that runs the **xcopy.exe** command or a similar command to copy the content to the target computer. Ensure that the **Run Command Line** task sequence step type occurs prior to any task sequence steps or scripts that depend on the files being copied to the target computers. For more information about modifying task sequence steps, see Configure the Task Sequence Steps and Step Sequence.

Copy Content to Target Computers Using OEM Folders

MDT supports using legacy OEM folders to organize and copy supplemental files to the target computers. Data WIM files are preferred over OEM folders.

> ⓘ **Note**
>
> In an instance where multiple OEM folders have been defined, the first driver that LTIApply.wsf finds is deployed to the target computer.

For more information about using data WIM files or OEM folders see the *Windows Assessment and Deployment Kit User's Guide* in the Windows ADK.

MDT looks in the following locations within the deployment share, in the order specified, to find an OEM folder:

- **Control*task_sequence*** (where *task_sequence* is the name or ID of the task sequence that MDT is installing). Create OEM folders in this location to create a custom folder for each build.
- **Operating Systems*Name*** (where Name is the name of the operating system MDT is installing). Create OEM folders in this location to create a custom folder for each operating system.
- ***Platform*** (where *Platform* is either x86 or x64). Create OEM folders in this location to create a custom folder for each platform.
- **OEM**, which is at the root of the deployment share and is the default OEM folder if a folder is not found in the previous locations.

  An OEM folder contains supplemental files. The following list describes each folder that you can create within an OEM folder to organize these files:

- **$$**. Windows Setup copies the contents of this folder to %SystemRoot% on each destination computer. It replicates all the folders, subfolders, and files that this folder contains in the %SystemRoot% folder of each destination computer. For Windows Setup to copy a file to %SystemRoot%\System32 on each destination computer, for example, put the file in OEM\$$\System32.
- **$1**. Windows Setup copies the contents of this folder to %SystemDrive% on each destination computer. It replicates all the folders, subfolders, and files that this folder contains in the %SystemDrive% folder on each destination computer. This is typically drive C on most computers.
- **Drive**. Drive is a drive letter (C, D, E, and so on). Windows Setup copies the contents of this folder to the root of the corresponding drive on each destination computer. It replicates all the folders, subfolders, and files that this folder contains in the corresponding drive during the setup process. For example, Windows Setup copies any files put in OEM\D to the root of drive D on each destination computer.

  Microsoft recommends that these folders not be used. The folders rely on a very specific disk configuration on the destination computer. Use $1 to represent %SystemDrive%, instead. In most installations, OEM\$1 and OEM\C write to the same location: the root of drive C.

Creating Custom Scripts for MDT

Scripts provide automation of the image-build and overall deployment process. They scan the configuration files, query the configuration database, evaluate environment variables to determine the rules to be used when deploying the images to the target computers, and perform many other intricate deployment tasks. MDT uses both Microsoft Visual Basic® Scripting Edition (VBScript [.vbs]) and Windows Script file (.wsf) scripts. Typically, there is no need to modify one of the delivered scripts. If a modification is necessary, instead of modifying one of the delivered scripts, copy the script to a new file, update it, and thoroughly test the effect of any change.

The scripts create log files as the scripts automate the deployment process. The log files record the status of the deployment process and can be used to assist in troubleshooting this process:

- Develop new scripts for use in MDT deployments as described in Develop Custom Scripts.
- Create new scripts for use in MDT deployments from a template as described in Create New Scripts from a Template.
- Create Windows PowerShell scripts for use in MDT deployments as described in Create Windows PowerShell Scripts for Use in MDT.

Develop Custom Scripts

You can develop new scripts for use in MDT deployments. These scripts should be in the form of .vbs or .wsf files. For examples of scripts that the Deployment Workbench uses, open the installation path of the deployment share, and then open the Scripts folder.

> ⓘ **Note**
>
> Microsoft does not support customized and custom scripts.

Before describing how to create a script, it is best to review how the scripts included with MDT are constructed. The standard MDT script is a .wsf file, which allows references to be made to functions that are contained in other scripts. MDT scripts leverage this functionality by referencing the ZTIUtility.vbs script and the ZTIDataAccess.vbs script. The ZTIUtility.vbs script is used to initialize the MDT environment and setup classes. The ZTIDataAccess.vbs script includes the common routines for database access, including connecting to and querying databases, and provides a web service interface.

The scripts define several standard objects that need not be declared in the script:

- **oFSO**. File System Object

- **oShell**. WScript Shell object
- **oEnv**. Process Environment object
- **oNetwork**. WScript Network object

  The following classes are defined that perform several standard tasks:

- **Environment**. Configures environment variables gathered through WMI and MDT rule processing, allowing for direct reference from the script, and is defined in ZTIUtility.vbs as described in Environment Class.
- **Logging**. Provides the logging functionality that all MDT scripts use, creating a single log file for each script and a consolidated log file of all scripts and is defined in ZTIUtility.vbs as described in Logging Class.
- **Utility**. Provides general utility functionality and is defined in ZTIUtility.vbs as described in Utility Class.
- **Database**. Provide access to databases and is defined in ZTIDataAccess.vbs as described in Database Class. The **Database** class:
  - Is used by ZTIGather.wsf when processing database rules from the CustomSettings.ini or BootStrap.ini files
  - Can be used to access databases in scripts instead of configuring the CustomSettings.ini or BootStrap.ini files; you can specify the parameters for accessing the database in the scripts
- **WebService**. Provides access to web services and is defined in ZTIDataAccess.vbs as described in WebService Class. The **WebService** class:
  - Is used by ZTIGather.wsf when processing web service rules from the CustomSettings.ini or BootStrap.ini files
  - Can be used to access web services in scripts instead of configuring the CustomSettings.ini or BootStrap.ini files; you can specify the parameters for accessing the web services in the scripts

Environment Class

Reference this class in scripts through the **oEnvironment** object. For example, change the computer name to **Example** using the command:

```
VB
```

```
oEnvironment.Item("ComputerName") = "Example"
```

Or, to determine whether this is a 32-bit or 64-bit architecture, query the architecture using the command:

```
VB

oEnvironment.Item("Architecture")
```

Logging Class

Reference this class in scripts through the **oLogging** object. When creating an informational log entry, use the command:

```
VB

oLogging.CreateEntry "Informational message", LogTypeInfo
```

When creating an error log entry, use the command:

```
VB

oLogging.CreateEntry "An error occurred",LogTypeError
```

Utility Class

Reference this class in scripts through the **oUtility** object. To determine the name of the current script, use the command:

```
VB

oUtility.ScriptName
```

To find the location of a file, use the command:

```
VB

iRetVal = oUtility.FindFile("CustomSettings.ini", sIniFile)
```

Database Class

Reference this class in scripts through the **Database** class. You can create an instance of the object class and connect to a database using following script excerpt:

```
VB
```

```
<script language="VBScript" src="ZTIUtility.vbs"/>   <script
language="VBScript" src="ZTIDataAccess.vbs"/>
<script language="VBScript">

   Dim oDatabase
   Dim oRecordset

   Set oDatabase = new Database
   oDatabase.SQLServer = "NYC-MDT-01"
   oDatabase.Instance = "SQLExpress"
   oDatabase.Database = "MDTDB"
   oDatabase.Port = ""
   oDatabase.Netlib = "DBNMPNTW"
   oDatabase.Table = "ComputerSettings"
   oDatabase.Parameters = "UUID, AssetTag, SerialNumber, MacAddress"
   oDatabase.ParameterCondition = "OR"
   oDatabase.SQLShare = "DeploymentShare$"

   oDatabase.Connect

   Set oRecordset = oDatabase.Query
   WScript.Echo "Records retrieved: " & oRecordset.RecordCount
```

WebService Class

Reference this class in scripts through the **WebService** class. You can create an instance of the object class and connect to a database using following script excerpt:

```
VB
```

```
<script language="VBScript" src="ZTIUtility.vbs"/>   <script
language="VBScript" src="ZTIDataAccess.vbs"/>
<script language="VBScript">

   Dim oWebService
   Dim oXML

   oEnvironment.Item("USZip") = "98029"
   oEnvironment.Item("USZip") = "98029"
   Set oWebService = new WebService
   oWebService.WebService =
"https://www.webservicex.net/uszip.asmx/GetInfoByZIP"
   oWebService.Parameters = "USZip"

   Set oXML = oWebService.Query
```

```
    WScript.Echo "Web service response:"
    WScript.Echo oXML.XML
```

Create New Scripts from a Template

You can also create scripts for use in the imaging process. You call these scripts by adding them to the Task Sequence Editor and ultimately by adding them to the TS.xml file. Listing 13 shows a template for creating custom scripts.

Listing 13. Custom Script Template

VB

```
<job id="Z-Sample">
<script language="VBScript" src="ZTIUtility.vbs"/>
<script language="VBScript">

'
//***************************************************************************
*
' // ***** Script Header *****
' //
' // Solution: Solution Accelerator for Microsoft Deployment
' // File: Z-Sample.wsf
' //
' // Purpose: Template
' //
' // Usage: cscript Z-Sample.wsf [/debug:true]
' //
' // Customer Build Version: 1.0.0
' // Customer Script Version: 1.0.0
' // Customer History:
' //
' // ***** End Header *****
'
//***************************************************************************
*

'//---------------------------------------------------------------------------
---
'//
'// Global constant and variable declarations
'//
'//---------------------------------------------------------------------------
---

Option Explicit

Dim iRetVal

'//---------------------------------------------------------------------------
```

```
---
'// End declarations
'//----------------------------------------------------------------------
---

'//----------------------------------------------------------------------
---
'// Main routine
'//----------------------------------------------------------------------
---

On Error Resume Next
iRetVal = ZTIProcess
ProcessResults iRetVal
On Error Goto 0

'//----------------------------------------------------------------------
--
'//
'// Function: ZTIProcess()
'//
'// Input: None
'//
'// Return: Success - 0
'// Failure - non-zero
'//
'// Purpose: Perform main ZTI processing
'//
'//----------------------------------------------------------------------
--
Function ZTIProcess()

     iRetVal = Success

     ZTIProcess = iRetval

     '!!!!!!!!!!!   INSERT YOUR CODE HERE   !!!!!!!!!!!

End Function

</script>
</job>
```

At a high level, complete the following steps to add a custom script:

1. Create the script based on the template.
2. Place the script in the Scripts folder of the:
    - Deployment share for LTI deployments
    - Package source for the Microsoft Deployment Toolkit Files package

3. Create a task sequence step based on the **Run Command Line** task sequence step type in your task sequence.
4. Configure the task sequence step created in the previous step to run your script.

   Follow these guidelines when creating a script:

- Always declare variables.
- Only create objects where required, because MDT includes most objects that are needed.
- Verify that ZTIUtility.vbs and ZTIDataAccess.vbs do not already provide the functionality required before writing a function.

Create Windows PowerShell Scripts for Use in MDT

MDT allows you to create Windows PowerShell scripts, and then run those scripts as a part of a MDT task sequence using the **Run PowerShell Script** task sequence step type. The Windows PowerShell scripts that you create can perform any typical automation supported by the target operating system.

Create Windows PowerShell scripts for use in MDT by:

1. Including the prerequisites in your Windows PowerShell script for running in MDT as described in Include Prerequisites for Running Windows PowerShell Scripts in MDT
2. Using task sequence variables within your Windows PowerShell script as described in Use Task Sequence Variables Within Windows PowerShell Scripts
3. Updating the MDT logs with output from your Windows PowerShell scripts as described in Update MDT Logs Using Windows PowerShell Scripts
4. Interpreting Windows PowerShell return codes generated by your script as described in Interpret Windows PowerShell Script Return Codes

Include Prerequisites for Running Windows PowerShell Scripts in MDT

When a **Run PowerShell Script** task sequence step runs a Windows PowerShell script, the step automatically loads the **Microsoft.BDD.TaskSequenceModule** module prior to running the script. The **Microsoft.BDD.TaskSequenceModule** module is responsible for creating the **TSENV**: and **TSENVLIST**: Windows PowerShell drives.

In addition, it is recommended that your script load the **ZTIUtility.psm1** module at the beginning of your script by adding the following line to the beginning of your script:

Import-Module `.\ZTIUtility.psm1`

Including the **ZTIUtility.psm1** module at the beginning of your script provides the following features:

1. Creates the **TSENV**: and **TSENVLIST**: Windows PowerShell drives so that you can test your script outside a task sequence
2. Provides support for Windows PowerShell task sequence module features that may be provided in future releases

   For more information on the **TSENV**: and **TSENVLIST**: Windows PowerShell drives, see Use Task Sequence Variables Within Windows PowerShell Scripts.

> **Note**
>
> For Configuration Manager task sequences, run the **Use Toolkit Package** task sequence step prior to running the **Run PowerShell Script** task sequence step.

If your script needs to load other custom modules, places those modules in a subfolder beneath the Tools\Modules folder. By default, the module path is modified to include the Tools\Modules folder and all subfolders. For example, if you have a module in the MyCustomModule.psm1 file, create the Tools\Modules\MyCustomModule folder and the place MyCustomModule.psm1 file in that folder. To load your custom module in your script, include the following line in your script:

```PowerShell
Import-Module MyCustomModule
```

Use Task Sequence Variables Within Windows PowerShell Scripts

Your Windows PowerShell script can get or set task sequence variables that can be consumed by MDT. You get or set task sequence variables using the following Windows PowerShell drives:

- **TSENV**:. This drive is a list of all the task sequence variables and their current values. You can access this drive as you would any other Windows drive. For example, you could type the following command within Windows PowerShell to list all the task sequence variables and their current values:dir tsenv:

- **TSENVLIST**:. This drive lists all the task sequence variables and their current values as an array.

  > **Note**
  >
  > Any values in the TSENV: and TSENVLIST: drives are lost after the task sequence completes as any values are stored in memory and are not persisted after the task sequence finishes.

  The following are examples that illustrate how to access task sequence variables using these drives:

- `$tsenv:GPOPackPath = "\\WDG-MDT-01\Packages$\GPOPacks"`

  This command updates the **ApplyGPOPack** task sequence variable to "\\WDG-MDT-01\Packages$\GPOPacks"

- `Write-Host $tsenv:ScriptRoot`

  This command displays the contents of the **ScriptRoot** task sequence variable to the Windows PowerShell console.

- `$packages = $tsenvlist:PACKAGES`

  This command sets the **$packages** Windows PowerShell variable to the current value of the **PACKAGES** task sequence variable.

- `$tsenvlist:PACKAGES = @("XXX00001:Program","XXX00002:Program")`

  This command sets the value of the **PACKAGES** task sequence variable to `"XXX00001:Program","XXX00002:Program"`.

Update MDT Logs Using Windows PowerShell Scripts

By default, the output from your Windows PowerShell scripts is written to BDD.LOG. Table 175 lists the message types that appear in the BDD.LOG file and how those types of messages are generated by your script.

Table 175. BDD.LOG Message Types and Script Output That Generates Them

Expand table

Message type	Generated by the following cmdlets
Informational	Any cmdlets that write to the Windows PowerShell console, such as Write-Host or Out-Host
Warning	Write-Warning cmdlet
Error	Write-Error cmdlet

The progress for your Windows PowerShell script is shown in the task sequence progress UI and is displayed as follows:

- Progress is shown as a percentage of total completion for the current script and any messages that are being displayed.
- Progress is updated using the standard Write-Progress cmdlet.

> **Note**
>
> Unlike some of the other scripts in MDT, a separate log file is not generated for the task sequence step running the Windows PowerShell script. Logging is only performed in the BDD.LOG file.

Interpret Windows PowerShell Script Return Codes

By default, if your Windows PowerShell script reports a terminating error or non-zero return code, the task sequence step will fail and stop running (unless you have configured the step to ignore the generated return code).

Table 176 lists the predefined return codes that MDT returns and provides a brief description of each. Any other return codes not listed in Table 176 were returned by the Windows PowerShell script.

Table 176. Predefined Windows PowerShell Script Return Codes

Expand table

Return code	Description
10901	Unable to find the Windows PowerShell script in the location specified.

Return code	Description
10902	No Windows PowerShell script was specified in the task sequence step.
10903	A terminating error was reported by the Windows PowerShell script.

Performing Deployments Using the MDT DB

MDT includes a database—the MDT DB—that you can use to provide configuration settings for LTI or ZTI deployments using Configuration Manager. Configure the MDT DB through the Deployment Workbench in MDT or any other data-management tool that you can use to modify information stored in SQL Server databases.

Conceptually, the MDT DB is a centralized version of the CustomSettings.ini file. The advantage to using the MDT DB is that for larger deployments, you have a centralized repository for managing deployment configuration settings.

You can store the MDT DB on the same SQL Server computer that is used for Configuration Manager, on a SQL Server computer on which MDT is installed, or on any other SQL Server computer in your organization.

Perform deployments using the MDT DB by:

- Prepare the MDT DB to provide configuration settings for MDT deployments as described in Preparing the MDT DB.
- Select the methods for applying configuration settings to targeted computers using the MDT DB as described in Selecting the Methods for Applying Configuration Settings.
- Manage the configuration settings stored in the MDT DB as described in Managing Configuration Settings Stored in the MDT DB
- Configuring MDT deployment process to retrieve configuration settings from the MDT DB by changing CustomSettings.ini as described in Configuring the MDT Deployment Process to Retreive Configuration Settings from the MDT DB
- Extending the schema of the MDT DB as described in Extending the MDT DB Schema

Preparing the MDT DB

Before you can use the MDT DB to provide configuration settings to MDT deployments, you must prepare the MDT DB for use by MDT. Prepare the MDT DB for storing

Configuration settings by:

- Creating a new MDT DB or connecting to an existing MDT DB as described in Creating a New MDT DB or Connecting to an Existing MDT DB
- Upgrading the MDT DB from a previous version of MDT as described in Upgrading an Existing MDT DB
- Assigning appropriate permissions to the MDT DB as described in Assigning the Appropriate Permissions to the MDT DB

Creating a New MDT DB or Connecting to an Existing MDT DB

Before you can manage configuration settings in the MDT DB, create a new MDT DB or connect to an existing MDT DB in the Deployment Workbench. The MDT DB contains the database objects that the MDT deployment process uses, such as views, tables, and stored procedures. You create a new MDT DB or connect to an existing the MDT DB database objects using the Advanced Configuration/Database node in the Deployment Workbench.

> **Note**
>
> To create and manage the MDT DB for LTI or ZTI deployments using Configuration Manager, create a deployment share in the Deployment Workbench. However, for ZTI deployments, you do not need to add any content to the deployment share, as ZTI deployments do not require it.

Create a new MDT DB or connect to an existing MDT DB by performing one of the following tasks:

- Create a new MDT DB as described in Create a New MDT DB.
- Connect to an existing MDT DB as described in Connect to an Existing MDT DB.

Create a New MDT DB

Create a new MDT DB, and then create the MDT DB database objects in the database. You can create the MDT DB in:

- An existing database as described in Create the MDT DB in an Existing Database
- A new database as described in Create the MDT DB in a New Database

Create the MDT DB in an Existing Database

If you want to store the MDT DB in a SQL Server database that database administrator (DBA) manages, the DBA must create the MDT DB, and then grant you the appropriate permissions to create the MDT DB objects in the new database. When the database has been created, create the MDT DB in the new database using the Deployment Workbench.

> **Note**
>
> To create the MDT DB, the db_owner database role must exist in the database that the DBA created.

To create the MDT DB in an existing database

1. The DBA creates the database and grants you DBA permission on the database.
2. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
3. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/Advanced Configuration/Database (where *deployment_share* is the name of the deployment share in which you will add the application).
4. In the Actions pane, select **New Database**.

   The New DB Wizard starts.
5. Complete the New DB Wizard using the information in Table 177.

Table 177. Information for Completing the New DB Wizard

Expand table

On this wizard page	Do this
SQL Server Details	a. In **SQL Server name**, type ***computer_name*** (where *computer_name* is the name of the computer running SQL Server). b. In **Instance**, type ***instance_name*** (where *instance_name* is the name of SQL Server instance on the computer running SQL Server).

On this wizard page	Do this
	If you are using SQL Server installed as the default instance, you can leave **Instance** blank. If you are using Microsoft SQL Server Express, type **SQLExpress** for the instance name. c. In **Port**, type ***port_number*** (where *port_number* is the TCP port number for the SQL Server instances if different than the default value). If you are using the default value, you can leave **Port** blank. d. In **Network Library**, select ***network_library*** (where *network_library* is the network library used to communicate with SQL Server and can be either **Named Pipes** or **TCP/IP Sockets**).
Database	a. Select **Create (or recreate) the tables and views in an existing database**. b. In **Database**, select ***database_name*** (where *database_name* is the name of the database in which you want to store the MDT DB). If you select a database that has an existing MDT DB, the information in the existing MDT DB tables and views will be lost. c. Select **Next**.
SQL Share	a. In **SQL Share**, type ***share_name*** (where *share_name* is the name of a network shared folder on the computer specified on the **SQL Server Details** wizard page). This text box is required for Windows PE to connect to the database using the Windows Integrated Security with Named Pipes protocol. Windows PE connects to the network shared folder, establishing a network connection to the server so that Windows Integrated Security in SQL Server authenticates correctly. b. If Windows PE is not connecting to the database using this protocol, you can leave **SQL Share** blank. c. Select **Next**.
Summary	Review the information in the **Details** box, and then select **Next**.
Confirmation	You can select **Save Output** to save the output of the wizard to a file. You can also select **View Script** to view the Windows PowerShell scripts used to perform the wizard tasks. Select **Finish**.

The New DB Wizard finishes. The database configuration information is listed in the details pane of the Deployment Workbench.

Create the MDT DB in a New Database

In instances where you are the SQL Server DBA or have the necessary permissions, you can create the new MDT DB, and then create the MDT DB database objects in the new database using the Deployment Workbench.

> **Note**
>
> To create the MDT DB, you need the sysadmin or dbcreator server roles.

To create the MDT DB in a new database

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/Advanced Configuration/Database (where *deployment_share* is the name of the deployment share in which you will add the application).
3. In the Actions pane, select **New Database**.

   The New DB Wizard starts.
4. Complete the New DB Wizard using the information in Table 178.

Table 178. Information for Completing the New DB Wizard

Expand table

On this wizard page	Do this
SQL Server Details	a. In **SQL Server name**, type ***computer_name*** (where *computer_name* is the name of the computer running SQL Server). b. In **Instance**, type ***instance_name*** (where *instance_name* is the name of the SQL Server instance on the computer running SQL Server).

On this wizard page	Do this
	If you are using SQL Server installed as the default instance, you can leave **Instance** blank. If you are using SQL Server Express, type **SQLExpress** for the instance name. c. In **Port**, type ***port_number*** (where *port_number* is the TCP port number for the SQL Server instance if different than the default value). d. In **Network Library**, select ***network_library*** (where *network_library* is the network library used to communicate with SQL Server and can be either **Named Pipes** or **TCP/IP Sockets**). e. Select **Next**.
Database	a. Select **Create a new database**. b. In **Database**, type ***database_name*** (where *database_name* is the name of the database in which you want to store the MDT DB). c. Select **Next**.
SQL Share	a. In **SQL Share**, type ***share_name*** (where *share_name* is the name of a network shared folder on the computer specified on the **SQL Server Details** wizard page). This text box is required for Windows PE to connect to the database using the Windows Integrated Security with Named Pipes protocol. Windows PE connects to the network shared folder, establishing a network connection to the server so that Windows Integrated Security in SQL Server authenticates correctly. If Windows PE is not connecting to the database using this protocol, you can leave **SQL Share** blank. b. Select **Next**.
Summary	Review the information in the Details box, and then select **Next**.
Confirmation	You can select **Save Output** to save the output of the wizard to a file. You can also select **View Script** to view the Windows PowerShell scripts used to perform the wizard tasks. Select Finish.

The New DB Wizard finishes. The database configuration information is shown in the details pane of the Deployment Workbench.

Connect to an Existing MDT DB

You can connect the Deployment Workbench to an existing MDT DB, which allows you to copy, or replicate, the MDT DB to another SQL Server instance, and then configure CustomSettings.ini to access the MDT DB.

> **Note**
>
> To connect to an existing MDT DB, you must be a user in the MDT DB and be granted the db_datareader database role.

To connect to an existing MDT DB

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/Advanced Configuration/Database (where *deployment_share* is the name of the deployment share in which you will add the application).
3. In the Actions pane, select **New Database**.

   The New DB Wizard starts.
4. Complete the New DB Wizard using the information in Table 179.

Table 179. Information for Completing the New DB Wizard

Expand table

On this wizard page	Do this
SQL Server Details	a. In **SQL Server name**, type ***computer_name*** (where *computer_name* is the name of the computer running SQL Server). b. In **Instance**, type ***instance_name*** (where *instance_name* is the name of SQL Server instance on the computer running SQL Server). If you are using SQL Server installed as the default instance, you can leave **Instance** blank. If you are using SQL Server Express, type **SQL Express** for the instance name.

On this wizard page	Do this
	c. In **Port**, type ***port_number*** (where *port_number* is the TCP port number for the SQL Server if different than the default value). If you are using the default value, you can leave **Port** blank. d. In **Network Library**, select ***network_library*** (where *network_library* is the network library used to communicate with SQL Server and can be either Named Pipes or TCP/IP Sockets). e. Select **Next**.
Database	a. Select **Use an existing database that already contains the required tables and views**. b. In **Database**, select ***database_name*** (where *database_name* is the name of the existing MDT DB). c. Select**Next**.
SQL Share	a. In **SQL Share**, type ***share_name*** (where *share_name* is the name of a network shared folder on the computer specified on the **SQL Server Details** wizard page). This text box is required for Windows PE to connect to the database using the Windows Integrated Security with Named Pipes protocol. Windows PE connects to the network shared folder, establishing a network connection to the server so that Windows Integrated Security in SQL Server authenticates correctly. If Windows PE is not connecting to the database using this protocol, you can leave **SQL Share** blank. b. Select **Next**.
Summary	Review the information in the **Details** box, and then select **Next**.
Confirmation	You can select **Save Output** to save the output of the wizard to a file. You can also select **View Script** to view the Windows PowerShell scripts used to perform the wizard tasks. Select **Finish**.

The New DB Wizard finishes. The database configuration information is shown in the details pane of the Deployment Workbench.

Upgrading an Existing MDT DB

The MDT DB is automatically upgraded from a previous version of MDT when you upgrade the deployment share. The upgrade process preserves any schema modification in the existing MDT DB when you upgrade to the new MDT DB.

If for some reason the existing MDT DB is not upgraded when you upgrade the deployment share, you can manually upgrade the MDT DB using the **Upgrade-MDTDatabaseSchema** Windows PowerShell cmdlet.

Assigning the Appropriate Permissions to the MDT DB

Depending on the tasks to be performed, you may need different permissions on the MDT DB. Table 180 lists the types of tasks to be performed and the corresponding SQL Server server roles and database roles required to perform them.

Table 180. Roles and Required Roles

Expand table

To perform this task	You need these roles
Create a new MDT DB from an existing database	db_owner database role in the MDT DB
Create a new MDT DB and database at the same time	sysadmin or dbcreator server role in the SQL Server instance in which the MDT DB resides
Connect to an existing MDT DB	db_datareader database role in the MDT DB or rights to the individual tables and views in the MDT DB
Modify the schema of the MDT DB	db_ddladmin database role in the MDT DB
Modify the configuration information stored in the MDT DB	db_datawrite database role in the MDT DB or rights to the individual tables and view in the MDT DB
View the configuration information stored in the MDT DB	db_datareader database role in the MDT DB or rights to the individual tables and views in the MDT DB

For more information on how to assign these permissions see Identity and Access Control (Database Engine).

Selecting the Methods for Applying Configuration Settings

After preparing the MDT DB, select the method for applying MDT configuration settings using the MDT DB. You can manage the configuration settings stored in the MDT DB using the nodes beneath the Database node in a Deployment Workbench deployment share.

> **Note**
>
> Managing configuration settings stored in the MDT DB using the Deployment Workbench is good when adding a limited number of entries or managing individual entries. If you want to make bulk additions or updates to the MDT DB, use standard SQL Server database tools as described in Manage Configuration Settings Stored in the MDT DB Using SQL Server Management Tools.

Each node beneath the Database node corresponds to a method for applying configuration settings. Table 181 lists the nodes beneath the Database node and describes when to make configuration settings in each node.

Table 181. Nodes Beneath the Database Node in a Deployment Share

Expand table

Node	Use this node to define
Computers	Specific target computers based on one of these properties: **AssetTag**, **UUID**, **SerialNumber**, or **MACAddress** properties. Associate property settings, applications, packages, roles, and Administrator-level accounts with a computer. Select this method when you want to provide configuration settings to a specific target computer. For example, you could use the MAC address of the primary network adapter of each computer to uniquely identify each computer.
Roles	A group of computers based on the tasks performed by the users of the target computers (using the **Role** property). Associate property settings, applications, packages, and Administrator-level accounts with a role. Select this method when you want to provide configuration settings to a group of computers that perform a specific job function. The roles can be based on users or actual target computers. For example, you could provide configuration settings for highly secured computers used to process confidential data.
Locations	A group of computers using the **DefaultGateway** property of the target computers to identify a geographic location. Associate property settings, applications, packages, roles, and Administrator-level accounts with a location.

Node	Use this node to define
	Select this method when you want to provide configuration settings to a group of computers that reside with in a specific geographic site. For example, you could provide configuration settings for services that are provided within the local site.
Make and Model	A group of computers using the **Make** and **Model** properties of the target computers. Associate property settings, applications, packages, roles, and Administrator-level accounts with target computers that are of the same make and model. Select this method when you want to provide configuration settings to a group of computers that have the same make and model. For example, you could deploy device drivers to the computers that all have the same make and model.

You can use any combination of the methods listed in Table 181 to provide configuration settings for target computers. For example, one set of configuration settings could be provided based on the location of the target computer, while another set of configuration settings could be provided based on the role of the computer, and finally, a set of configuration settings could be provided based on the MAC address of the target computer.

> ⓘ **Note**
>
> Create the items in the Roles node before creating items beneath the other nodes (Computers, Locations, and Make and Model), because items in the other nodes can be associated with roles.

Managing Configuration Settings Stored in the MDT DB

The MDT DB contains tables and views that the MDT deployment process can query. The tables contain configuration settings that are accessible through the views.

You can manage the configuration settings stored in the MDT DB using any combination of the following methods:

1. Manage configuration settings stored in the MDT DB using the Deployment Workbench as described in Manage Configuration Settings Stored in the MDT DB Using the Deployment Workbench.
2. Manage configuration settings stored in the MDT DB using SQL Server database tools as described in Manage Configuration Settings Stored in the MDT DB Using SQL Server Management Tools.

Manage Configuration Settings Stored in the MDT DB Using the Deployment Workbench

You can manage the configuration settings stored in the MDT DB using the nodes beneath the Database node in a Deployment Workbench deployment share.

> ⓘ **Note**
>
> Managing configuration settings stored in the MDT DB using the Deployment Workbench is good when adding a limited number of entries or managing individual entries. If you want to make bulk additions or updates to the MDT DB, use standard SQL Server database tools as described in Manage Configuration Settings Stored in the MDT DB Using SQL Server Management Tools.

To manage the configuration settings stored in the MDT DB using the Deployment Workbench, perform the following steps:

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/Advanced Configuration/Database/*config_method* (where *deployment_share* is the name of the deployment share to configure and config_method is the configuration method to be used and can be Computers, Roles, Locations, or Make and Models).
3. In the Actions pane, select **New**.
4. In the **Properties** dialog box, complete the **Identity** tab, based on the method being used to apply configuration settings listed in Table 182.

Table 182. Information for Completing the Identity Tab

Expand table

For this method	Do this
Computers	a. In **Description**, type ***description_name*** (where *description_name* is a descriptive name of the computer). b. Provide any combination of the following information (you must configure at least one of the following):

For this method	Do this
	- In **Asset tag**, type ***asset_tag*** (where *asset_tag* is the asset tag or inventory control tag assigned to the computer). - In **UUID**, type ***uuid*** (where *uuid* is the UUID assigned to the target computer). - In **Serial number**, type ***serial_number*** (where *serial_number* is the serial number assigned to the target computer). - In **MAC address**, type ***mac_address*** (where *mac_address* is the MAC address of the primary network adapter for the target computer).
Roles	In **Role name**, type ***role_name*** (where *role_name* is a descriptive name of the role).
Locations	a. In **Location**, type ***location_name*** (where *location_name* is a descriptive name of the location). b. In **Default Gateways**, add the IP addresses for all default gateways that exist within the location.
Make and Models	a. In **Make**, type ***make_name*** (where *make_name* is the exact make [manufacturer] name returned by the BIOS of the target computer using WMI). b. In **Model**, type ***model_name*** (where *model_name* is the exact model name returned by the BIOS of the target computer using WMI).

5. Complete the **Details** tab by performing the following steps:

   a. In the **Property** column, find ***property*** (where *property* is the name of the property you wish to configure).

   b. In the **Value** column, type ***property_value*** (where *property_value* is the value you wish to assign to the property.

      For more information on the properties that you can configure for MDT, see the corresponding sections under the section, "Property Definition", in the MDT document *Toolkit Reference*.

   c. Repeat steps a and c for each property that you want to configure.

6. Complete the **Applications** tab by performing any combination of the actions in Table 183.

Table 183. Information for Completing the Application Tab

Expand table

To	Do this
Add an application from the deployment share (which is managed in the Applications node in the Deployment Workbench)	a. Select **Add**, and then select **Lite TouchApplication**. The **Select an item** dialog box appears. b. In the **Select an item** dialog box, select the application to add, and then select **OK**. The application must exist in the same deployment share as the deployment share associated with the MDT DB you are configuring.
Add a Configuration Manager application	a. Select **Add**, and then select **ConfigMgr 2012 Application**. The **New ConfigMgr Application** dialog box appears. b. In the **New ConfigMgr Application** dialog box, in **Please specify the new ConfigMgr application to add**, type ***configmgr_application***, and then select **OK** (where *configmgr_application* is the name of the Configuration Manager application to be added). The name entered in **Please specify the new ConfigMgr application to add** must match the name of the Configuration Manager application exactly. For more information on specifying the application name, see the section, "Install Application," at Task Sequence Steps in Configuration Manager. The **Allow this application to be installed from the Install Application task sequence action without being deployed**check box must be selected for the Configuration Manager application to be deployed properly.
Remove an application from the list of applications to be deployed	Select ***application*** (where *application* is the name of the application that you want to remove), and then select **Remove**.

To	Do this
Install an application prior to other applications	Select ***application*** (where *application* is the name of the application that you want to install before other applications), and then select **Up**.
Install an application after other applications	Select ***application*** (where *application* is the name of the application that you want to install after other applications), and then select **Down**.

7. Complete the **ConfigMgr Packages** tab by performing any combination of the actions in Table 184.

Table 184. Information for Completing the ConfigMgr Packages Tab

Expand table

To	Do this
Add a Configuration Manager package and program combination	a. Select **Add**. The **New ConfigMgr Package** dialog box appears. b. In the **New ConfigMgr Package** dialog box, in **Please specify the new ConfigMgr package and program to add**, type ***configmgr_pkg_prog***, and then select **OK** (where *configmgr_pkg_prog* is the name of the Configuration Manager package and corresponding program to be added). The format for entering the package and program combination is as follows: ***package_id:program_name*** In this format: - *package_id* is the Configuration Manager package identifier assigned by Configuration Manager You can determine the package identifier for a package by viewing the package in the Configuration Manager console. - *program_name* is the name of the program to be run for the package specified in ***package_id*** You can determine the program name by viewing the package and corresponding programs in the Configuration Manager console.

To	Do this
	The value entered for **program_name** must match the name of the Configuration Manager program exactly.
Remove a package and corresponding program from the list of packages and programs to be deployed	Select ***pkg_prog*** (where *pkg_prog* is the name of the package and corresponding program that you want to remove), and then select **Remove**. Removing an application from this list does not remove the application from the deployment share or from Configuration Manager.
Install a package and corresponding program prior to other packages and programs	Select ***pkg_prog*** (where *pkg_prog* is the name of the package and corresponding program that you want to install before other packages and programs), and then select **Up**.
Install a package and corresponding program after other packages and programs	Select ***pkg_prog*** (where *pkg_prog* is the name of the package and corresponding program that you want to install after other packages and programs), and then select **Down**.

8. Complete the **Roles** tab by performing any combination of the actions in Table 185.

> **Note**
>
> The **Roles** tab is not available when configuring configuration settings in the MDT using the roles method. The **Roles** tab is available for all other methods.

Table 185. Information for Completing the Roles Tab

Expand table

To	Do this
Add a role	Select **Add**, and then select ***role_name*** (where *role_name* is the name of the role to be added). The role must exist prior adding the role. As a recommended best practice, create configuration settings for roles first and then configure the remaining methods.
Remove a role	Select ***role_name*** (where *role_name* is the name of the role that you want to remove), and then select **Remove**.

To	Do this
	Removing a role from this list does not remove the role from the Roles node in the Deployment Workbench.

9. Complete the **Administrators** tab by performing any combination of the actions in Table 186, and then select **OK**.

Table 186. Information for Completing the Administrators Tab

Expand table

To	Do this
Add an administrator	a. Select **Add**. b. The **New Administrator** dialog box appears.
Remove an administrator	Select ***admin_name*** (where *admin_name* is the name of the user or group account that you want to remove), and then select **Remove**. Removing an account from this list does not remove the account from AD DS or from the local computer.

Manage Configuration Settings Stored in the MDT DB Using SQL Server Management Tools

Managing the configuration settings stored in the MDT DB using the Deployment Workbench is good for management of a limited number of database entries. However, if you need to add hundreds of entries (such as adding the configuration settings for the target computers) or if you want to perform a bulk update to the configuration settings, use SQL Server database management tools.

Perform bulk import and update of configuration settings in the MDT DB using:

- **Microsoft Office system applications**. Microsoft Office Access® or Office Excel have data-manipulation features that allow you to directly modify tables in the MDT DB using tools with which you are already familiar.
- **bcp utility (Bcp.exe)**. This bulk import/export tool allows you to import configuration settings to and export configuration settings from the tables in the MDT DB.

- **Transact-SQL language in SQL Server**. The Transact-SQL language allows you to add, update, or delete entries in tables in the MDT DB.

  You can review the tables, views, and other database objects in the MDT DB to determine the tables that you want to update. For more information on the tables and views in the MDT DB, see the section, "Tables and Views in the MDT DB", in the MDT document*Toolkit Reference*.

Configuring the MDT Deployment Process to Retreive Configuration Settings from the MDT DB

Configure the MDT deployment process to access the MDT DB and retrieve the configuration settings stored in it by modifying CustomSettings.ini. You can modify CustomSettings.ini to allow the MDT deployment process to access the MDT DB using any combination of the following methods:

- Use the Configure DB Wizard in the Deployment Workbench as described in Configure the Retreival of Configuration Settings from the MDT DB Using the Configure DB Wizard.
- Directly modify the CustomSettings.ini file as described in Configure the Retreival of Configuration Settings from the MDT DB by Directly Modifying the CustomSettings.ini File.

Configure the Retrieval of Configuration Settings from the MDT DB Using the Configure DB Wizard

The Configure DB Wizard in Deployment Workbench provides a graphical interface for configuring the MDT deployment process to access the MDT DB. The advantages of using this wizard is that it helps reduce the:

1. Possibility of configuration errors in the CustomSettings.ini file
2. Effort required to modify the CustomSettings.ini file to enable retrieval of configuration settings from the MDT DB

   The disadvantage of using the Configure DB Wizard is that it does not allow you to access any additional database objects that you have created, such as tables, views, or stored procedures. However, the wizard recognizes any columns that you have added to the existing tables and views.

   In most instances, use the Configure DB Wizard to modify the CustomSettings.ini file. Only make direct modifications to the CustomSettings.ini file if you have

added database objects, such as tables, views, or stored procedures.

Run the Configure DB Wizard for each deployment share defined in the Deployment Workbench that will be updated.

To configure CustomSettings.ini for database queries using the Configure DB Wizard in the Deployment Workbench

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/Advanced Configuration/Database (where *deployment_share* is the name of the deployment share to which you will add the application).
3. In the Actions pane, select **Configure Database Rules**.

   The Configure DB Wizard starts.
4. Complete the **Computer Options** wizard page using the information in Table 187, and then select **Next**.

Table 187. Information to Complete the Computer Options Wizard Page

Expand table

Select this check box	To do this
Query for computer-specific settings	Query the MDT DB for settings configured on the **Details** tab of the computer item's **Properties** dialog box.
Query for roles assigned to this computer	Query the MDT DB for roles associated with the computer on the **Roles** tab of the computer item's **Properties** dialog box.
Query for applications to be installed on this computer	Query the MDT DB for Lite Touch and Configuration Manager applications to be installed on the computer as configured on the **Applications** tab of the computer item's **Properties** dialog box.
Query for SMS packages to be installed on this computer	Query the MDT DB for Configuration Manager packages and programs to be installed on the computer as configured on the **ConfigMgr Packages** tab of the computer item's **Properties** dialog box.

Select this check box	To do this
Query for administrators to be assigned to this computer	Query the MDT DB for user or group accounts that are to be made members of the local Administrators group on the target computer as configured on the **Administrators** tab of the computer item's **Properties** dialog box.

> **Tip**
>
> You can select **Select All** or **Deselect All** to correspondingly select or deselect all the check boxes on the **Computer Options** wizard page.

5. Complete the **Location Options** wizard page using the information in Table 188, and then select **Next**.

Table 188. Information to Complete the Location Options Wizard Page

Expand table

Select this check box	To do this
Query for location names based on default gateways	Query the MDT DB for location names based on the IP addresses of the default gateways configured on the **Identity** tab of the location item's **Properties** dialog box.
Query for location-specific settings	Query the MDT DB for settings configured on the **Details** tab of the location item's **Properties** dialog box.
Query for roles assigned for this location	Query the MDT DB for roles associated with the location as configured on the **Roles** tab of the location item's **Properties** dialog box.
Query for applications to be installed for this location	Query the MDT DB for Lite Touch and Configuration Manager applications to be installed on the target computers within the location as configured on the **Applications** tab of the location item's **Properties** dialog box.
Query for SMS packages to be installed for this location	Query the MDT DB for Configuration Manager packages and programs to be installed on the target computers within the location as configured on the **ConfigMgr Packages** tab of the location item's **Properties** dialog box.
Query for administrators to be	Query the MDT DB for user and group accounts that are to be made members of the local Administrators group on the target

Select this check box	To do this
assigned for this location	computers within the location as configured on the **Administrators** tab of the location item's **Properties** dialog box.

> **Tip**
>
> You can select **Select All** or **Deselect All** to correspondingly select or deselect all the check boxes on the **Location Options** wizard page.

6. Complete the **Make/Model Options** wizard page using the information in Table 189, and then select **Finish**.

Table 189. Information to Complete the Make/Model Options Wizard Page

Expand table

Select this check box	To do this
Query for model-specific settings	Query the MDT DB for settings configured on the **Details** tab of the make and model item's **Properties** dialog box.
Query for roles assigned to computers with this make and model	Query the MDT DB for roles associated with the make and model on the **Roles** tab of the make and model item's **Properties** dialog box.
Query for applications to be installed on computers with this make and model	Query the MDT DB for Lite Touch and Configuration Manager applications to be installed on the target computers with the make and model as configured on the **Applications** tab of the make and model item's **Properties** dialog box.
Query for SMS packages to be installed for this make and model	Query the MDT DB for Configuration Manager packages and programs to be installed on the target computers with the make and model as configured on the **ConfigMgr Packages** tab of the make and model item's **Properties** dialog box.
Query for administrators to be assigned to machines with this make and model	Query the MDT DB for user and group accounts that are to be made members of the local Administrators group on the target computer with the make and model as configured on the **Administrators** tab of the make and model item's **Properties** dialog box.

> **Tip**

You can select **Select All** or **Deselect All** to correspondingly select or deselect all the check boxes on the **Make/Model Options** wizard page.

7. Complete the **Role Options** wizard page using the information in Table 190, and then select **Next**.

Table 190. Information to Complete the Role Options Wizard Page

Expand table

Select this check box	To do this
Query for role-specific settings	Query the MDT DB for settings configured on the **Details** tab of the role item's **Properties** dialog box.
Query for applications to be installed for this role	Query the MDT DB for Lite Touch and Configuration Manager applications to be installed on computers that perform this role as configured on the **Applications** tab of the role item's **Properties** dialog box.
Query for SMS packages to be installed for this role	Query the MDT DB for Configuration Manager packages and programs to be installed on the target computers that perform this role as configured on the **ConfigMgr Packages** tab of the role item's **Properties** dialog box.
Query for administrators to be assigned for this role	Query the MDT DB for user and group accounts that are to be made members of the local Administrators group on computers that perform this role as configured on the **Administrators** tab of the role item's **Properties** dialog box.

Tip

You can select **Select All** or **Deselect All** to correspondingly select or deselect all the check boxes on the **Role Options** wizard page.

8. On the **Summary** wizard page, review the information in the **Details** box, and then select **Next**.

9. On the **Confirmation** wizard page, review the information, and then select **Finish**.

   When the Configure DB Wizard has finished, the CustomSettings.ini file is configured to perform the selected queries. For more information about how the

CustomSettings.ini file is configured using the Configure DB Wizard, see Review the CustomSettings.ini File After Running the Configure DB Wizard.

For more information about customizing each property, see:

- The corresponding section for each property in the section, "Properties", in the MDT document Toolkit Reference

For more information about customizing tables and views in the configuration database, see:

1. The section, "Tables and Views in the MDT DB", in the MDT document *Toolkit Reference*

Configure the Retreival of Configuration Settings from the MDT DB by Directly Modifying the CustomSettings.ini File

The Configure DB Wizard in the Deployment Workbench provides the method that requires the least effort, requires minimal knowledge about how the CustomSettings.ini file is modified, and minimizes the risk of configuration errors. However, if you need to create custom queries for information, you can modify CustomSettings.ini to return the proper query results.

> ⓘ **Note**
>
> Although it is possible to manually customize the CustomSettings.ini file to retrieve configuration settings from the MDT DB without running the Configure DB Wizard, it is recommend that you run the Configure DB Wizard, and then modify the CustomSettings.ini file to meet your needs.

Directly modify the CustomSettings.ini file to configure the retrieval of configuration settings from the MDT DB by performing the following steps:

- Review the CustomSettings.ini file after running the Configure DB Wizard as a starting point for customization as described in Review the CustomSettings.ini File After Running the Configure DB Wizard.
- Configure the properties in the CustomSettings.ini file that are used to create a SQL query as described in Configure Properties That Are Used to Create an SQL Query.
- Configure the properties in the CustomSettings.ini file that are used for deployment to a specific computer as described in Configure Properties That Are

Used for Deployment to a Specific Computer.

- Configure the properties in the CustomSettings.ini file that are used for deployment by location as described in Configure Properties That Are Used for Deployment by Location.
- Configure the properties in the CustomSettings.ini file that are used for deployment by computer make and model as described in Configure Properties That Are Used for Deployment by Computer Make and Model.
- Configure the properties in the CustomSettings.ini file that are used for deployment by role as described in Configure Properties That Are Used for Deployment by Role.

Review the CustomSettings.ini File After Running the Configure DB Wizard

When the Configure DB Wizard has finished, the CustomSettings.ini file is configured to perform the selected queries. **Error! Reference source not found**. provides an example of the CustomSettings.ini file after the Configure DB Wizard has finished.

Listing 14. The CustomSettings.ini File After the Configure DB Wizard Has Finished

ini

```
[Settings]
Priority=CSettings, CPackages, CApps, CAdmins, CRoles, Locations, LSettings,
LPackages, LApps, LAdmins, LRoles, MMSettings, MMPackages, MMApps, MMAdmins,
MMRoles, RSettings, RPackages, RApps, RAdmins, Default
Properties=MyCustomProperty

[Default]
OSInstall=Y

SkipCapture=NO
SkipAdminPassword=YES
SkipProductKey=YES

[CSettings]
SQLServer=R33-dep-01
Instance=R33instance
Database=AdminDB
Netlib=DBNMPNTW
SQLShare=Client
Table=ComputerSettings
Parameters=UUID, AssetTag, SerialNumber, MacAddress
ParameterCondition=OR

[CPackages]
```

```
SQLServer=R33-dep-01
Instance=R33instance
Database=AdminDB
Netlib=DBNMPNTW
SQLShare=Client
Table=ComputerPackages
Parameters=UUID, AssetTag, SerialNumber, MacAddress
ParameterCondition=OR
Order=Sequence

[CApps]
SQLServer=R33-dep-01
Instance=R33instance
Database=AdminDB
Netlib=DBNMPNTW
SQLShare=Client
Table=ComputerApplications
Parameters=UUID, AssetTag, SerialNumber, MacAddress
ParameterCondition=OR
Order=Sequence

[CAdmins]
SQLServer=R33-dep-01
Instance=R33instance
Database=AdminDB
Netlib=DBNMPNTW
SQLShare=Client
Table=ComputerAdministrators
Parameters=UUID, AssetTag, SerialNumber, MacAddress
ParameterCondition=OR

[CRoles]
SQLServer=R33-dep-01
Instance=R33instance
Database=AdminDB
Netlib=DBNMPNTW
SQLShare=Client
Table=ComputerRoles
Parameters=UUID, AssetTag, SerialNumber, MacAddress
ParameterCondition=OR

[Locations]
SQLServer=R33-dep-01
Instance=R33instance
Database=AdminDB
Netlib=DBNMPNTW
SQLShare=Client
Table=Locations
Parameters=DefaultGateway

[LSettings]
SQLServer=R33-dep-01
Instance=R33instance
Database=AdminDB
Netlib=DBNMPNTW
```

```
SQLShare=Client
Table=LocationSettings
Parameters=DefaultGateway

[LPackages]
SQLServer=R33-dep-01
Instance=R33instance
Database=AdminDB
Netlib=DBNMPNTW
SQLShare=Client
Table=LocationPackages
Parameters=DefaultGateway
Order=Sequence

[LApps]
SQLServer=R33-dep-01
Instance=R33instance
Database=AdminDB
Netlib=DBNMPNTW
SQLShare=Client
Table=LocationApplications
Parameters=DefaultGateway
Order=Sequence

[LAdmins]
SQLServer=R33-dep-01
Instance=R33instance
Database=AdminDB
Netlib=DBNMPNTW
SQLShare=Client
Table=LocationAdministrators
Parameters=DefaultGateway

[LRoles]
SQLServer=R33-dep-01
Instance=R33instance
Database=AdminDB
Netlib=DBNMPNTW
SQLShare=Client
Table=LocationRoles
Parameters=DefaultGateway

[MMSettings]
SQLServer=R33-dep-01
Instance=R33instance
Database=AdminDB
Netlib=DBNMPNTW
SQLShare=Client
Table=MakeModelSettings
Parameters=Make, Model

[MMPackages]
SQLServer=R33-dep-01
Instance=R33instance
Database=AdminDB
```

```
Netlib=DBNMPNTW
SQLShare=Client
Table=MakeModelPackages
Parameters=Make, Model
Order=Sequence

[MMApps]
SQLServer=R33-dep-01
Instance=R33instance
Database=AdminDB
Netlib=DBNMPNTW
SQLShare=Client
Table=MakeModelApplications
Parameters=Make, Model
Order=Sequence

[MMAdmins]
SQLServer=R33-dep-01
Instance=R33instance
Database=AdminDB
Netlib=DBNMPNTW
SQLShare=Client
Table=MakeModelAdministrators
Parameters=Make, Model

[MMRoles]
SQLServer=R33-dep-01
Instance=R33instance
Database=AdminDB
Netlib=DBNMPNTW
SQLShare=Client
Table=MakeModelRoles
Parameters=Make, Model

[RSettings]
SQLServer=R33-dep-01
Instance=R33instance
Database=AdminDB
Netlib=DBNMPNTW
SQLShare=Client
Table=RoleSettings
Parameters=Role

[RPackages]
SQLServer=R33-dep-01
Instance=R33instance
Database=AdminDB
Netlib=DBNMPNTW
SQLShare=Client
Table=RolePackages
Parameters=Role
Order=Sequence

[RApps]
SQLServer=R33-dep-01
```

```
Instance=R33instance
Database=AdminDB
Netlib=DBNMPNTW
SQLShare=Client
Table=RoleApplications
Parameters=Role
Order=Sequence

[RAdmins]
SQLServer=R33-dep-01
Instance=R33instance
Database=AdminDB
Netlib=DBNMPNTW
SQLShare=Client
Table=RoleAdministrators
Parameters=Role
```

Configure Properties That Are Used to Create an SQL Query

The Configure DB Wizard creates entries in the CustomSettings.ini file that are used to build an SQL query. Table 191 lists the MDT properties that are used to create the query. For more information on the properties in Table 191, see the corresponding section for each property in the section, "Properties", in the MDT document *Toolkit Reference*.

Table 191. Properties That Are Used to Create an SQL Query

Expand table

Property	Description
SQLServer	Specifies the name of the computer running SQL Server to use in the query
Instance	Specifies the name of the SQL Server instance on the computer specified in the **SQLServer** property. If you are using: - SQL Server installed as the default instances, you can leave **Instance** blank - SQL Server Express, type **SQLExpress** for the instance name
Database	Specifies the name of the MDT DB in the SQL Server instance specified in the **Instance** property.
Netlib	Specifies the network library to be used in connecting to the MDT DB specified in the **Database** property and can be either Named Pipes

Property	Description
	(DBNDPNTW) or TCP/IP Sockets (DBMSSOCN).
SQLShare	Specifies a network shared folder on the computer specified in the **SQLServer** property, which is used when performing user authentication using Windows Integrated Authentication with Named Pipes protocol.
Table	Specifies the table or view to be used in the query. If you specify the **Table** property, you cannot specify the **StoredProcedure** property, too: You must specify one or the other.
StoredProcedure	Specifies the stored procedure to be used in the query. If you specify the **StoredProcedure** property, you cannot specify the **Table** property, too: You must specify one or the other.
Parameters	Specifies the selection criteria for returning rows from the query. Conceptually, this property is the **WHERE** clause in an SQL **SELECT** statement.
ParameterCondition	Specifies the Boolean operation to be performed when you specify more than one selection criterion in the **Parameters** property. You can select either Boolean **AND** (the default) or **OR** operations. If you want to use a Boolean **OR** operation, you must specify **ParameterCondition=OR**.
Order	Specifies the sort order of the rows returned from the query. Conceptually, this property is the **ORDER BY** clause in an SQL **SELECT** statement.

> ⓘ Note
>
> The SQL query that these properties create returns all columns from the table or view specified in the **Table** property.

Most of these properties in Table 191 were configured when you created the MDT DB. Listing 15 provides an example of CustomSettings.ini file that creates an SQL query that returns all the configuration information in the Computers view for a computer that matches the **SerialNumber** or **AssetTag** of the computer being deployed.

Listing 15. The CustomSettings.ini File After the Configure DB Wizard Has Finished

ini

```
[Settings]
Priority=CSettings, Default

[Default]
OSInstall=YES

[CSettings]
```

```
SQLServer=NYC-SQL-01
Instance=SQLEXPRESS
Database=MDTDB
Netlib=DBNMPNTW
SQLShare=SQL$
Table=ComputerSettings
Parameters=UUID, AssetTag, SerialNumber, MacAddress
ParameterCondition=OR
```

The Priority line in the CustomSettings.ini file contains the `Computers` value, which directs MDT to process the `[CSettings]` section later in the CustomSettings.ini file. The properties in the `[CSettings]` section performs the query and returns the result set from the query to be used during the MDT deployment process, such as the computer name or IP address of the target computer.

You can review the tables and views in the MDT DB to determine the columns you want to reference in the **Parameters** property. For more information on the tables and views in the MDT DB, see the section, "Tables and Views in the MDT DB", in the MDT document *Toolkit Reference*.

Configure Properties That Are Used for Deployment to a Specific Computer

The Configure DB Wizard configures the **Priority** property and creates the corresponding section in the CustomSettings.ini file for computer specific deployments using the MDT DB. Table 192 lists the sections created and a provides brief description of the purpose of each section. Refer to the example CustomSettings.ini file in Table 192.

Table 192. Sections in the CustomSettings.ini File for Deployment to a Specific Computer and Their Purpose

Expand table

Section	Purpose
`CSettings`	Queries the **ComputerSettings** view to identify a computer based on any combination of the **UUID, AssetTag**, **SerialNumber**, and **MACAddress** properties
`CPackages`	Queries the **ComputerPackages** view to return all the packages associated with a computer
`Capps`	Queries the **ComputerApplications** view to return all the applications associated with a computer

Section	Purpose
`CAdmins`	Queries the **ComputerAdministrators** view to return all the user accounts to be made members of the local Administrator group on the target computer
`CRoles`	Queries the **ComputerRoles** view to return all the roles associated with a computer

For more information about the database views and tables that the Deployment Workbench manages, see the section, "Tables and Views in the MDT DB", in the MDT document *Toolkit Reference*.

Configure Properties That Are Used for Deployment by Location

The Configure DB Wizard configures the **Priority** property and creates the corresponding section in the CustomSettings.ini file for deployment by location using the MDT DB. Table 193 lists the sections created and a brief description of purpose for each section. Refer to the example CustomSettings.ini file in Table 193.

Table 193. Sections in the CustomSettings.ini File for Deployment by Location and Their Purpose

Expand table

Section	Purpose
`Locations`	Queries the Locations view to identify the locations that are associated with the IP address of a default gateway
`LSettings`	Queries the **LocationSettings** view to return all the property values associated with a location
`LPackages`	Queries the **LocationPackages** view to return all the packages associated with a location
`LApps`	Queries the **LocationApplications** view to return all the applications associated with a location
`LAdmins`	Queries the **LocationAdministrators** view to return all the user accounts to be made members of the local Administrator group on the target computers within a location
`LRoles`	Queries the **LocationRoles** view to return all the roles associated with a location
`CSettings`	Queries the **ComputerSettings** view to return all the property values associated with a specific computer

For more information about the database views and tables that the Deployment Workbench manages, see the section, "Tables and Views in the MDT DB", in the MDT document *Toolkit Reference*.

Configure Properties That Are Used for Deployment by Computer Make and Model

The Configure DB Wizard configures the **Priority** property and creates the corresponding section in the CustomSettings.ini file for deployment by computer make and model using the MDT DB. Table 194 lists the sections created and provides brief description of the purpose of each. Refer to the example CustomSettings.ini file in Table 194.

Table 194. Sections in the CustomSettings.ini File for Deployment by Computer Make and Model and Their Purpose

Expand table

Section	Purpose
`MMSettings`	Queries the **MakeModelSettings** view to return all the property values associated with a make and model combination
`MMPackages`	Queries the **MakeModelPackages** view to return all the packages associated with a make and model combination
`MMApps`	Queries the **MakeModelApplications** view to return all the applications associated with a make and model combination
`MMAdmins`	Queries the **MakeModelAdministrators** view to return all the user accounts to be made members of the local Administrator group on the target computers that match a make and model combination
`MMRoles`	Queries the **MakeModelRoles** view to return all the roles associated with a make and model combination

For more information about the database views and tables that the Deployment Workbench manages, see the section, "Tables and Views in the MDT DB", in the MDT document *Toolkit Reference*.

Configure Properties That Are Used for Deployment by Role

The Configure DB Wizard configures the **Priority** property and creates the corresponding section in the CustomSettings.ini file for deployments by role using the MDT DB. Table 195 lists the sections created and provides a brief description of purpose for each. Refer to the example CustomSettings.ini file in Table 195.

Table 195. Sections in the CustomSettings.ini File for Deployment by Roles and Their Purpose

Expand table

Section	Purpose
`CRoles`	Queries the **ComputerRoles** view to identify the roles that are associated with a computer based on any combination of the **UUID**, **AssetTag**, **SerialNumber**, and **MACAddress** properties
`RSettings`	Queries the **RoleSettings** view to return all the property values associated with a role
`RPackages`	Queries the **RolePackages** view to return all the packages associated with a role
`RApps`	Queries the **RoleApplications** view to return all the applications associated with a role
`RAdmins`	Queries the **RoleAdministrators** view to return all the user accounts to be made members of the local Administrator group on the target computers associated with a role
`LRoles`	Queries the **LocationRoles** view to return all the roles associated with a location
`MMRoles` `MMRoles`	Queries the **MakeModelRoles** view to return all the roles associated with a specific make and model

For more information about the database view and tables that the Deployment Workbench manages, see the section, "Tables and Views in the MDT DB", in the MDT document *Toolkit Reference*.

Extending the MDT DB Schema

Although the MDT DB contains most of the common configuration settings that you would need in your deployment, there may be instances in which you need to extend the information associated with one or more of the entities in the MDT DB, such as computers, roles, locations, or the makes and models.

If so, you must extend the schema for one or more of the tables and views in the MDT DB. If you extend the schema for the tables, you might also need to modify the views that depend on those tables. You extend the schema for the:

- Tables using the Microsoft SQL Server Management Studio or the **ALTER TABLE** Transact-SQL statement
- Views using the SQL Server Management Studio or the **ALTER VIEW** Transact-SQL statement

  To extend the schema of the MDT DB, review the tables and views in the MDT DB to determine which you need to modify. For more information on the tables and views in the MDT DB, see the section, "Tables and Views in the MDT DB", in the MDT document *Toolkit Reference*.

To extend the schema of the MDT DB

1. Determine the tables and views to be modified by reviewing the schema of the MDT DB as described in Determine the Tables and Views to Be Modified.
2. Create a backup of the existing MDT DB as described in Create a Backup of the MDT DB.
3. Modify the tables using SQL Server Management Studio or the **ALTER** Transact-SQL statement as described in Modify the Table.
4. Modify and refresh the views that are dependent on the table and in which you want to return the new column in the result set as described in Modify and Refresh the Dependent Views.
5. Reference the new column in the CustomSettings.ini file as described in Reference the New Column in the CustomSettings.ini File.
6. Create a task sequence step that references the new column as described in Reference the New Column in a Task Sequence Step.

Determine the Tables and Views to Be Modified

Determine the tables and views to be modified by reviewing the schema of the tables and views in the MDT DB. The table most commonly modified is **Settings**, which is common to all views, such as the **ComputerSettings** or **LocationSettings**.

> **Tip**

the views provided with MDT do not require modification, because they already return all columns.

For more information on the tables and views in the MDT DB, see the section, "Tables and Views in the MDT DB", in the MDT document Toolkit *Reference*.

> **Tip**
>
> To determine which views are dependent on a table, look at the Transact-SQL statements used to create the view by right-clicking the view, and then selecting **Edit**.

Example: How Woodgrove Bank Determined the Tables and Views to Be Modified

Woodgrove Bank has decided that it needs to configure the antivirus server based on the location of a target computer. To do this, the bank's IT pros need to add an antivirus server configuration setting for each location that they define in the MDT DB.

To make the antivirus server information available, they must:

- Modify the **Settings** table
- Retrieve the information through the **LocationSettings** view

Create a Backup of the MDT DB

As a best practice, create a backup of the MDT DB before modifying the tables and views in it. Create a backup of the MDT DB using:

- **SQL Server Management Studio**. For more information on how to perform database backups using this method, see "How to: Back Up a Database (SQL Server Management Studio)" in SQL Server Books Online, included with SQL Server.
- **Existing SQL Server backup solution in your organization**. For more information on how to perform database backups using this method, see the documentation for your backup solution, or contact your DBA.

Modify the Table

You modify the table by adding new columns to it. You add new columns to tables in the MDT DB using:

- **SQL Server Management Studio**. For more information on how to add a column using SQL Server Management Studio, see "How to: Insert Columns into Tables (Visual Database Tools)" in SQL Server Books Online, included with SQL Server.
- **ALTER TABLE Transact-SQL statement**. For more information on how to add a column using the **ALTER TABLE** Transact-SQL statement, see "ALTER TABLE (Transact-SQL)" in SQL Server Books Online, included with SQL Server.
- **Other SQL Server management tools**. For more information on how to perform this task using other SQL Server management tools, see the documentation for the corresponding tool.

  To add a new column to a table, you need to know:

- The columns that you will add to the table
- The SQL Server data type for each column you will add to the table
- If the new column allows Null values
- The default value for the new column (optional)
- Check constraints for the new column (optional)

  For more information on how to determine what to use for each item in the previous list, see the section, "Table Column Properties (SQL Server Management Studio)," in SQL Server Books Online, included with SQL Server.

  Example: How Woodgrove Bank Modified Tables

  Woodgrove Bank will add a new column to the **Settings** table called **AVServer**. The **AVServer** column will contain the computer name of the antivirus server for each location specified in the **LocationIdentity** table. Woodgrove Bank has decided to create the **AVServer** column using the information in Table 196.

Table 196. Information for the AVServer Column

Expand table

Property	Value
Name	AVServer
Allow Nulls	Yes
Data Type	char(20)

Modify and Refresh the Dependent Views

After the column has been added to the table, modify any views you want to display for the newly created column. Typically, you would add the new column to custom views that you have created.

> ⓘ **Note**
>
> The views provided with MDT do not require modifications, because all these views already return all columns. If you have not created custom views, you can skip this step.

Modify the dependent views in the MDT DB using:

- **SQL Server Management Studio**. For more information on how to add a column using SQL Server Management Studio, see the section, "Working with Views (Visual Database Tools)," in SQL Server Books Online, included with SQL Server.
- **ALTER VIEW Transact-SQL statement**. For more information on how to drop a view using the **ALTER VIEW** Transact-SQL statement, see the section, "ALTER View (Transact-SQL)," in SQL Server Books Online, included with SQL Server.
- **Other SQL Server management tools**. For more information on how to perform this task using other SQL Server management tools, see the documentation for the corresponding tool.

  To modify the dependent view, you need to know:

- The columns that you will add to the view
- The SQL **SELECT** statement used to create the view

  For more information on determining what to use of each of the items in the previous list, see the section, "Modifying Views," in SQL Server Books Online, included with SQL Server.

  Even if the views do not require modification, refresh all views that reference modified tables by running the **sp_refreshview** stored procedure. The**sp_refreshview** stored procedure is required to update views when the columns in dependent tables or views have changed. For more information about the **sp_refreshview** stored procedure, see the section, "sp_refreshview (Transact-SQL)," in SQL Server Books Online, included with SQL Server.

  Example: How Woodgrove Bank Modified the Dependent Views

Woodgrove Bank does not need to modify any views, as the **LocationSettings** view already returns all columns from the **Settings** table. However, Woodgrove Bank ran the **sp_refreshview** stored procedure to refresh the **ComputerSettings**, **LocationSettings**, **MakeModelSettings**, or **RoleSettings** views, which reference the Settings table. This allows all the views to return the computer name of the antivirus server, if required.

Reference the New Column in the CustomSettings.ini File

After you have added the column to the table and modified the appropriate views, configure the CustomSettings.ini file to reference the new column. To reference the new column in the CustomSettings.ini file, perform the following steps:

1. Add a reference to the query section on the **Priority** line in the CustomSettings.ini file, if required.

   This reference initiates the query defined in the query section that you will create in a later step. This step may not be necessary if CustomSettings.ini already has an existing query for the view or table referenced in the query.

2. Add the new column name to the **Properties** line in the CustomSettings.ini file.

   This reference informs MDT to resolve the value for **AVServer** and create a task sequence variable of the same name. The task sequence variable can then be referenced in a task sequence step.

3. Create a query section that was referenced in step 1, which queries the appropriate table or view.

   Example: How Woodgrove Bank Referenced the New Column in the CustomSettings.ini File

   Woodgrove Bank queries the **LocationSettings** view to return the antivirus server for a particular location. The **LSettings** query already exists in the CustomSettings.ini file after running the Configure DB Wizard. As shown in Listing 16, all Woodgrove Bank needs to do is add **AVServer** to the `Priority` line so that MDT creates a task sequence variable of the same name.

   Listing 16. CustomSettings.ini File to Retrieve Antivirus Server for Woodgrove Bank

```ini
```

```
[Settings]
Priority=LSettings, Default
Properties=AVServer

[Default]
OSInstall=YES

[LSettings]
SQLServer=NYC-SQL-01
Instance=SQLExpress
Database=MDTDB
Netlib=DBNMPNTW
SQLShare=SQL$
Table=LocationSettings
Parameters=DefaultGateway
```

Reference the New Column in a Task Sequence Step

Now that the CustomSettings.ini file is modified to return the configuration settings from the new column, you are ready to reference the new column in a task sequence step. You reference the new column as a task sequence variable in the task sequence step. The variable will have the same name as the column. For example, if you create a column named **Zip_Code**, the task sequence variable will be named **Zip_Code**.

Example: How Woodgrove Bank Referenced the New Column in a Task Sequence Step

Woodgrove Bank creates a custom task sequence step to run the antivirus setup program. As a part of the antivirus setup program, the antivirus server name can be provided by using the **-server** parameter and referencing the new column name as a task sequence variable (%AVSERVER%). The command line used to run the antivirus setup program is:

```
avsetup.exe -server %AVSERVER%
```

Monitoring MDT Deployments

You can monitor MDT deployments using the monitoring feature supported by the MDT scripts and the Deployment Workbench. The MDT deployment monitoring feature allows you to view the MDT deployment process for LTI, ZTI, and UDI deployments. You can view the deployment process in the Deployment Workbench or by using the **Get-MDTMonitorData** cmdlet.

Monitor MDT deployments using the MDT monitoring features by performing the following steps:

1. Enable monitoring of MDT deployments as described in Enable MDT Deployment Monitoring.
2. View MDT deployment process as described in View MDT Deployment Progress.

Enable MDT Deployment Monitoring

Before you can monitor MDT deployments, you must enable MDT deployment monitoring. The process for enabling MDT deployment monitoring is different for LTI deployments and Configuration Manager-based deployments (including ZTI and UDI deployments).

Enable MDT deployment monitoring by:

- Enabling MDT monitoring for LTI deployments as described in Enabling LTI Deployment Monitoring
- Enabling MDT monitoring for ZTI or UDI deployments as described in Enabling ZTI or UDI Deployment Monitoring

Enabling LTI Deployment Monitoring

Prior to deploying computers using LTI, enable monitoring of the LTI deployment process. You enable monitoring on the **Monitoring** tab in the deployment share properties dialog box.

To enable monitoring of the LTI deployment process

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares.
3. In the details pane, select ***deployment_share*** (where *deployment_share* is the name of the deployment share where you want to enable monitoring).
4. In the Actions pane, select **Properties**.

   The ***deployment_share*** **Properties** dialog box opens (where *deployment_share* is the name of the deployment share for which you want to enable monitoring).
5. In the ***deployment_share*** **Properties** dialog box (where *deployment_share* is the name of the deployment share for which you want to enable monitoring), on the

Monitoring tab, select the **Enable monitoring for this deployment share** check box, and then select **Apply**.

6. In the ***deployment_share*** **Properties** dialog box (where *deployment_share* is the name of the deployment share for which you want to enable monitoring), on the Rules tab, notice that the **EventService** property has been added to the CustomSettings.ini file, and then select **OK**.

7. Close all open windows and dialog boxes.

Enabling ZTI or UDI Deployment Monitoring

Prior to deploying computers using ZTI or UDI, enable monitoring of theses deployment process in the Deployment Workbench. You enable monitoring on the **Monitoring** tab in the deployment share **Properties** dialog box just as you do for LTI deployments.

Then, copy the **EventService** property line on the **Rules** tab in the deployment share **Properties** dialog box to the CustomSettings.ini file in the MDT files package in Configuration Manager. Update the MDT files package on all distribution points.

To enable monitoring of the ZTI or UDI deployment processes

1. Enable MDT monitoring for a deployment share using the Deployment Workbench as described in Enabling LTI Deployment Monitoring.

2. Copy the **EventService** property line on the **Rules** tab in the deployment share **Properties** dialog box to the CustomSettings.ini file in the MDT files package in Configuration Manager.

   The following is an example of the **EventService** property line:

   ```
   EventService=https://WDG-MDT-01:9800
   ```

   For more information on customizing the MDT configuration files to include the **EventService** property line, see "Step 3-4: Customize the MDT Configuration Files for the Reference Computer" in*Quick Start Guide for Microsoft System Center 2012 R2 Configuration Manager* for Configuration Manager environments.

3. Update the MDT files package for distribution points so that the customized CustomSettings.ini file is available.

   For more information on updating the MDT files package for distribution points, see "Step 3-5: Update the Distribution Points for the Custom Settings Files

Package" in *Quick Start Guide for Microsoft System Center 2012 R2 Configuration Manager* for Configuration Manager environments.

View MDT Deployment Progress

You can view the MDT deployment progress using the Deployment Workbench or the **Get-MDTMonitorData** cmdlet.

> **Note**
>
> To view the MDT deployment progress, monitoring must be enabled as described in Enable MDT Deployment Monitoring.

To view the MDT deployment process, complete either of the following tasks:

1. View the MDT deployment progress using the Deployment Workbench as described in Viewing the MDT Deployment Progress in the Deployment Workbench.
2. View the MDT deployment progress using the **Get-MDTMonitorData** cmdlet as described in Viewing the MDT Deployment Progress Using the Get-MDTMonitorData Cmdlet.

Viewing the MDT Deployment Progress in the Deployment Workbench

You view the MDT deployment process in the **Monitoring** node in the deployment share. The progress of the LTI deployment process is displayed as a percentage of completion.

> **Note**
>
> The percentage of completion displayed in the Monitoring node is based on the percentage completion of the steps in the task sequence, not in overall time. For example, if a task sequence has completed 20 steps in task sequence that has a total of 50 steps, then the process will show 40% complete.

To view the LTI deployment process

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/deployment_share/Monitoring (where *deployment_share* is the name of the deployment share that you want to monitor)
3. In the details pane, view the deployment process for each computer being deployed.
4. In the Actions pane, periodically select **Refresh**.

   The status of the deployment process is updated in the details pane. Continue to monitor the deployment process till the process completes.
5. In the details pane, select ***target_computer*** (where *target_computer* is the name of the computer being monitored).
6. In the Actions pane, select **Properties**.

   The ***target_computer*** **Properties** dialog box is displayed (where *target_computer* is the name of the computer being monitored).
7. In the ***target_computer*** **Properties** dialog box (where *target_computer* is the name of the computer being monitored), on the **Identity** tab, view the monitoring information provided about the deployment process as described in Table 197.

Table 197. Monitoring Information About the Deployment Process

Expand table

Information	Description
ID	Unique identifier for the computer being deployed.
Computer Name	The name of the computer being deployed.
Deployment status	The current status of the computer being deployed; can be one of the following: - **Running**. Indicates that the task sequence is healthy and running. - **Failed**. Indicates that the task sequence failed and the deployment process was unsuccessful.

Information	Description
	- **Completed**. Indicates that the task sequence has finished. - **Unresponsive**. The task sequence has not updated its status in the past four hours and is assumed to be nonresponsive.
Step	The current task sequence step being run.
Progress	The overall progress of the task sequence. The progress bar indicates how many task sequence steps have been run out of the total number of task sequence steps.
Start	Time the deployment process started.
End	Time the deployment process ended.
Elapsed	The length of time the deployment process has been running or took to run if the deployment process has finished.
Error	The number of errors encountered during the deployment process.
Warnings	The number of warnings encountered during the deployment process.
Remote Desktop	This button allows you to establish a remote desktop connection with the computer being deployed using the Windows Remote Desktop feature. This method assumes that: - The target operating system is running and has remote desktop support enabled - **mstsc.exe** is in the path This button is always visible but may not be able to establish a remote desktop session if the monitored computer is running Windows PE, has not completed installation of the target operating system, or does not have the Remote Desktop feature enabled.
VM Connection	This button allows you to establish a remote desktop connection to virtual machine running in Hyper-V. This method assumes that: - The deployment is being performed to a VM running on Hyper-V - **vmconnect.exe** is located in the %ProgramFiles%\Hyper-V folder This button appears when ZTIGather.wsf detects that Hyper-V integration components are running on the monitored computer. Otherwise, this button will not be visible.

Information	Description
DaRT Remote Control	This button allows you to establish a remote control session using the remote viewer feature in the Diagnostics and Recovery Toolkit (DaRT). This method assumes that: - DaRT has been deployed to the target computer and is currently running - **DartRemoteViewer.exe** is located in the %ProgramFiles%\Microsoft DaRT 7\v7 folder This button appears when ZTIGather.wsf detects that DaRT is running on the monitored computer. Otherwise, this button will not be visible.
Automatically refresh this information every 10 seconds	Check box that controls whether the information in the dialog box is automatically refreshed. If the check box is: a. Selected, the information is refreshed every 10 seconds b. Cleared, the information is not automatically refreshed and must be manually refreshed using the **Refresh Now** button
Refresh Now	This button immediately refreshes the information displayed in the dialog box.

8. In the ***target_computer*** **Properties** dialog box (where *target_computer* is the name of the computer being monitored), select **OK**.

9. Close the Deployment Workbench

Viewing the MDT Deployment Progress Using the Get-MDTMonitorData Cmdlet

You can view the MDT deployment process using the **Get-MDTMonitorData** cmdlet. This cmdlet is included in the MDT PowerShell **microsoft.bdd.pssnapin** snap-in, which is included with MDT. To use this cmdlet, monitoring must be enabled as described in Enable MDT Deployment Monitoring.

To view MDT deployment progress using the Get-MDTMonitorData cmdlet

1. Open a Windows PowerShell console.

2. Add the MDT PowerShell snap-in by running the **Add-PSSnapIn** cmdlet as shown in the following example:

   Add-PSSnapIn Microsoft.BDD.PSSnapIn

3. Create a PowerShell drive that uses the MDT PowerShell provider by running the **New-PSDrive** cmdlet as shown in the following example:

   ```
   New-PSDrive -Name DS001 -PSProvider mdtprovider -Root d:\DeploymentShare$
   ```

4. View the MDT monitoring process by running the following command:

   ```
   Get-MDTMonitorData -Path DS001:
   ```

   This command returns the monitoring data collected by the MDT monitoring service running on the same computer that hosts the deployment share, as shown in the following example output:

   ```
   Name : WDG-REF-01

   PercentComplete : 100

   Settings :

   Warnings : 0

   Errors : 0

   DeploymentStatus : 3

   StartTime : 5/23/2012 6:45:39 PM

   EndTime : 5/23/2012 8:46:32 PM

   ID : 1

   UniqueID : 94a0830e-f2bb-421c-b1e0-6f86f9eb9fa1

   CurrentStep : 88

   TotalSteps : 88

   StepName :

   LastTime : 5/23/2012 8:46:32 PM

   DartIP :
   ```

`DartPort :`

`DartTicket :`

`VMHost : XYL-DC-02`

`VMName : WDG-REF-01`

`ComputerIdentities : {}`

For more information about the monitoring data that the cmdlet returns, see Table 197 in Viewing the MDT Deployment Progress in the Deployment Workbench.

5. Close the Windows PowerShell console.

Supporting Windows RE and DaRT

MDT integrates with Windows RE and DaRT to provide enhanced support and troubleshooting features. MDT support for Windows RE and DaRT is as follows:

- LTI supports Windows RE and DaRT in LTI boot images and on the recovery partition on the target computer as described in Supporting Windows RE and DaRT in LTI.
- ZTI and UDI support DaRT in boot images as described in Supporting DaRT in ZTI and UDI Boot Images.

Supporting Windows RE and DaRT in LTI

MDT supports the ability to deploy Windows RE partitions to computers. In addition, if your organization is licensed for the Microsoft Desktop Optimization Package (MDOP), you can include DaRT in the Windows RE partitions.

> ⓘ **Note**
>
> MDT does not support Windows RE in Windows 7 when using the Windows ADK.

Provide support for Windows RE and DaRT by performing the following steps:

1. Enable Windows RE support on target computers by installing the MDT-enabled boot image to the target computer as described in Enable Windows RE Support in

LTI.

2. Enable DaRT support on target computers by installing DaRT along with Windows RE as described in Enable DaRT Support in LTI.

3. Customize the DaRT configuration when DaRT is integrated with LTI as described in Customize DaRT When Integrated with LTI.

Enable Windows RE Support in LTI

Windows RE helps users troubleshoot and recover from startup-related problems on their computers. When a deployment share is updated, Deployment Workbench automatically generates .iso and .wim files that contain Windows RE support.

> **Note**
>
> You must import the entire Windows 7 installation files to a deployment share to support Windows RE. Otherwise, Windows 7 is installed without Windows RE support.

While running the LTI task sequence, the **Add Windows Recovery (WinRE)** task sequence step is responsible for:

- Installing the appropriate .wim image to the active partition
- Modifying the BCD file so that a user can choose to start Windows RE by pressing F8 as Windows is starting.

  The **Add Windows Recovery (WinRE)** task sequence step runs when the **PrepareWinRE** property is set to a value of **YES**. For more information about the **PrepareWinRE** property, see the "PrepareWinRE" property in the MDT document *Toolkit Reference*.

Enable DaRT Support in LTI

DaRT is included as a part of the Microsoft Desktop Optimization Package, which is provided as a part of Microsoft Software Assurance. You can include DaRT in the Windows RE partitions.

The following is a summary of the DaRT features:

- Includes 14 administrative, system, and network tools.

- Provides many options for recovery, even when Windows Safe mode or normal startup will not function
- Provides an easy-to-use, offline boot environment that helps IT teams quickly restart computers
- Helps recover deleted files and reset lost or forgotten local passwords, even the administrator's

  Enable DaRT support in LTI for:

- DaRT version 7 (used with Windows 7) as described in Enable DaRT 7 Support in LTI
- DaRT version 8 (used with Windows 8) as described in Enable DaRT 8 Support in LTI

Enable DaRT 7 Support in LTI

DaRT version 7 is for use with Windows 7. For information on how to enable DaRT version 8 for use with Windows 8, see Enable DaRT 8 Support in LTI.

To enable DaRT 7 support in LTI

1. Perform an administrative installation of DaRT on the computer running the Deployment Workbench.

   By default, if you do a traditional installation of DaRT, the processor architecture of DaRT is the same as the processor architecture of the operating system where you installed DaRT. For example, if you install DaRT on a 64-bit operating system, you will have a 64-bit version of DaRT.

   Performing an administrative installation allows you to install both 32-bit and 64-bit versions of DaRT. With an administrative installation the folder structures for both processor architectures are created, but none of the shortcuts are created.

   Perform an administrative installation of DaRT by running the following command from a command prompt:

   ```
   msiexec.exe /a MSDart70.msi
   ```

2. Copy the 32-bit version of Tools.cab file from the DaRT administrative installation to the Tools\x86 folder in a deployment share.
3. Copy the 64-bit version of Tools.cab file from the DaRT administrative installation to the Tools\x64 folder in a deployment share.

4. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.

5. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares

6. In the details pane, select ***deployment_share*** (where *deployment_share* is the name of the deployment share for which you want to enable DaRT support).

7. In the Actions pane, select **Properties**.

   The ***deployment_share*** **Properties** dialog box appears (where *deployment_share* is the name of the deployment share for which you want to enable DaRT support).

8. In the ***deployment_share*** **Properties** dialog box, on the **Windows PE** tab, select **platform** (where *deployment_share* is the name of the deployment share for which you want to enable DaRT support and *platform* is the processor architecture platform for which you want to enable DaRT support), select the **Microsoft Diagnostics and Recovery Toolkit (DaRT)** check box, and then select **OK**.

9. Update the deployment share.

   As a part of updating the deployment share, the DaRT files are integrated with the Lite Touch Windows PE .wim files, which automatically include Windows RE. When the .wim files are installed on the target computer, DaRT support will automatically be included.

   > ⓘ **Note**
   >
   > For more information about updating a deployment share see Update a Deployment Share in the Deployment Workbench.

10. Close all open windows and dialog boxes.

Enable DaRT 8 Support in LTI

DaRT version 8 is for use with Windows 8. For information on how to enable DaRT 7 for use with Windows 7, see Enable DaRT 7 Support in LTI.

To enable DaRT 8 support in LTI

1. Perform an administrative installation of DaRT on the computer running the Deployment Workbench.

By default, if you perform a traditional installation of DaRT, the processor architecture of DaRT is the same as the processor architecture of the operating system on which you installed DaRT. For example, if you install DaRT on a 64-bit operating system, you will have a 64-bit version of DaRT.

Performing an administrative installation allows you to install both 32-bit and 64-bit versions of DaRT. With an administrative installation, the folder structures for both processor architectures are created, but none of the shortcuts are created.

Perform an administrative installation of DaRT by running the following command from a command prompt:

```
msiexec.exe /a MSDart80.msi
```

2. If the computer on which you installed MDT is running Windows 8, you can proceed to step 5.

   MDT automatically performs the following two steps if you install DaRT 8 on the computer running Windows 8 and MDT.

3. Copy the Toolsx86.cab file from the DaRT administrative installation to the Tools\x86 folder in a deployment share.

   > **Tip**
   >
   > By default, the administrative installation of DaRT installs the Toolsx86.cab file in C:\Program Files\Microsoft DaRT 8\v8.

4. Copy the Toolsx64.cab file from the DaRT administrative installation to the Tools\x64 folder in a deployment share.

   > **Tip**
   >
   > By default, the administrative installation of DaRT installs the Toolsx64.cab file in C:\Program Files\Microsoft DaRT 8\v8.

5. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.

6. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares.

7. In the details pane, select ***deployment_share*** (where *deployment_share* is the name of the deployment share for which you want to enable DaRT support).

8. In the Actions pane, select **Properties**.

   The ***deployment_share*** **Properties** dialog box appears (where *deployment_share* is the name of the deployment share for which you want to enable DaRT support).

9. In the ***deployment_share*** **Properties** dialog box, on the **Windows PE** tab, select ***platform*** (where *deployment_share* is the name of the deployment share for which you want to enable DaRT support and *platform* is the processor architecture platform for which you want to enable DaRT support). Select the **Microsoft Diagnostics and Recovery Toolkit 8 (DaRT 8)** check box, and then select **OK**.

10. Update the deployment share.

    As a part of updating the deployment share, the DaRT files are integrated with the Lite Touch Windows PE .wim files, which automatically include Windows RE. When the .wim files are installed on the target computer, DaRT support will automatically be included.

    > ⓘ **Note**
    >
    > For more information about updating a deployment share see Update a Deployment Share in the Deployment Workbench.

11. Close all open windows and dialog boxes.

Customize DaRT When Integrated with LTI

You can customize DaRT, and then save the customizations in LTI so that the deployed partitions that include DaRT are configured consistently. You can do this by creating a new DaRT recovery image that includes the configuration settings you desire, then copying the DartConfig.dat file from the newly configured DaRT recovery image to an LTI deployment share.

To customize DaRT when integrated with LTI

1. Install DaRT on the computer where you installed MDT.

2. Create a new DaRT recovery image using the DaRT Recovery Image Wizard.

While running the DaRT Recovery Image Wizard, make the configuration settings that you wish to have applied to your environment. Select to save the DaRT recovery image as an .iso file.

For more information about how to create a new DaRT recovery image for:

- DaRT 7, see How to create & test Diagnostics & Recovery Toolkit (DaRT 7) Recovery Image-Part I.
- DaRT 8, see the section, "Create the DaRT 8 Recovery Image," in the *Microsoft Diagnostics and Recovery Toolset Administrator's Guide*, which is included with DaRT 8.

3. Extract the DartConfig.dat file from the .iso file created in the previous step.
4. Copy the DartConfig.dat file extracted in the previous step into the *installation_folder*\Templates folder (where *installation_folder* is the folder where you installed MDT) on the computer where you installed MDT.
5. Update the deployment share in the Deployment Workbench to create a LTI boot image that includes the customized DartConfig.dat file.

   For more information about updating a deployment share, see Update a Deployment Share in the Deployment Workbench.

Supporting DaRT in ZTI and UDI Boot Images

MDT supports DaRT version 7 (for Windows 7) and DaRT version 8 (for Windows 8) in ZTI and UDI boot images. This support makes DaRT features available when the ZTI or UDI boot images are running on the target computers.

After a ZTI or UDI boot image is enabled for DaRT, the **Use Toolkit Package** task sequence step will recognize that the DaRT remote control files are present and will automatically start the DaRT remote control agent. The DaRT remote control agent provides remote control of the target computer during the deployment process, which helps troubleshoot deployment problems.

Enable DaRT support in ZTI and UDI boot images for:

- DaRT version 7 (used with Windows 7) as described in Enable DaRT 7 Support in ZTI and UDI Boot Images.
- DaRT version 8 (used with Windows 8) as described in Enable DaRT 8 Support in ZTI and UDI Boot Images.

Enable DaRT 7 Support in ZTI and UDI Boot Images

DaRT version 7 is for use with Windows 7. For information on how to enable DaRT version 8 for use with Windows 8, see Enable DaRT 8 Support in ZTI and UDI Boot Images.

To enable DaRT 7 support in ZTI and UDI Boot Images

1. Perform an administrative installation of DaRT on the computer running the Deployment Workbench.

   By default, if you do a traditional installation of DaRT, the processor architecture of DaRT is the same as the processor architecture of the operating system where you installed DaRT. For example, if you install DaRT on a 64-bit operating system, you will have a 64-bit version of DaRT.

   Performing an administrative installation allows you to install both 32-bit and 64-bit versions of DaRT. With an administrative installation the folder structures for both processor architectures are created, but none of the shortcuts are created.

   Perform an administrative installation of DaRT by running the following command from a command prompt:

   ```
   msiexec.exe /a MSDart70.msi
   ```

2. Copy the 32-bit version of Tools.cab file from the DaRT administrative installation to the *installation_folder*\Templates\Distribution\Tools\x86 folder (where *installation_folder* is the folder where you installed MDT).

3. Copy the 64-bit version of Tools.cab file from the DaRT administrative installation to the *installation_folder*\Templates\Distribution\Tools\x64 folder (where *installation_folder* is the folder where you installed MDT).

4. Run the Create Boot Image using MDT wizard to generate the boot image.

   While running the Create Boot Image using MDT wizard, on the **General Settings: Components** wizard page, select the **Microsoft Diagnostics and Recovery Toolkit (DaRT)** check box.

   For information about how to run the Create Boot Image using MDT wizard for Configuration Manager, see Creating ZTI Boot Images in Configuration Manager

5. Close all open windows and dialog boxes.

Enable DaRT 8 Support in ZTI and UDI Boot Images

DaRT version 8 is for use with Windows 8. For information on how to enable DaRT 7 for use with Windows 7, see Enable DaRT 7 Support in ZTI and UDI Boot Images.

You enable DaRT support based on the operating system running on the computer on which you installed MDT. DaRT 8 can only be installed on Windows 8. If you install MDT on a computer running:

- Windows 8, then install DaRT on the same computer, and MDT will automatically copy the necessary files to support DaRT 8 as described in Enable DaRT 8 Support in ZTI and UDI Boot Images for Windows 8 Operating Systems.
- Operating systems prior to Windows 8, then perform an administrative installation on a computer running Windows 8, and then copy the Toolsx86.cab and Toolsx64.cab files to the computer running MDT as described in Enable DaRT 8 Support in ZTI and UDI Boot Images for Operating Systems Prior to Windows 8.

Enable DaRT 8 Support in ZTI and UDI Boot Images for Windows 8 Operating Systems

Enabling DaRT 8 support in ZTI and UDI boot images for Windows 8 Operating Systems requires the installation of DaRT 8 on the computer on which you installed MDT. After DaRT 8 is installed, the Deployment Workbench in MDT automatically copies the necessary DaRT 8 files to the appropriate locations.

To enable DaRT 8 support in ZTI and UDI boot images for Windows 8 operating systems

1. Install DaRT on the computer running MDT.

   Install DaRT by running the following command from a command prompt:

   ```
   msiexec.exe /i MSDart80.msi
   ```

2. Start the Deployment Workbench.

   When the Deployment Workbench starts, MDT copies the necessary DaRT files to the MDT installation. The Deployment workbench copies the:

   - Toolsx86.cab file from the DaRT installation to the *installation_folder*\Templates\Distribution\Tools\x86 folder (where *installation_folder* is the folder where you installed MDT).

- Toolsx64.cab file from the DaRT installation to the *installation_folder*\Templates\Distribution\Tools\x64 folder (where *installation_folder* is the folder where you installed MDT).

3. Run the Create Boot Image Using the MDT Wizard to generate the boot image.

   While running the Create Boot Image Using MDT Wizard, on the General Settings: Components wizard page, select the **Microsoft Diagnostics and Recovery Toolkit 8 (DaRT 8)** check box.

   For information about how to run the Create Boot Image Using MDT Wizard for Configuration Manager, see Creating ZTI Boot Images in Configuration Manager

4. Close all open windows and dialog boxes.

Enable DaRT 8 Support in ZTI and UDI Boot Images for Operating Systems Prior to Windows 8

Enabling DaRT 8 support in ZTI and UDI boot images for operating systems prior to Windows 8 requires an administrative installation of DaRT 8 on a computer running an operating system prior to Windows 8. After DaRT 8 is installed, you will need to manually copy the DaRT 8 files to the appropriate locations on the computer running MDT.

To enable DaRT 8 support in ZTI and UDI boot images for operating systems prior to Windows 8

1. Perform an administrative installation of DaRT on the computer running the Deployment Workbench.

   By default, if you perform a traditional installation of DaRT, the processor architecture of DaRT is the same as the processor architecture of the operating system on which you installed DaRT. For example, if you install DaRT on a 64-bit operating system, you will have a 64-bit version of DaRT.

   Performing an administrative installation allows you to install both 32-bit and 64-bit versions of DaRT. With an administrative installation the folder structures for both processor architectures are created, but none of the shortcuts are created.

   Perform an administrative installation of DaRT by running the following command from a command prompt:

   ```
   msiexec.exe /a MSDart80.msi
   ```

2. Copy the Toolsx86.cab file from the DaRT administrative installation to the *installation_folder*\Templates\Distribution\Tools\x86 folder (where *installation_folder* is the folder where you installed MDT).

   > **Tip**
   >
   > By default the administrative installation of DaRT installs the Toolsx86.cab file in C:\Program Files\Microsoft DaRT 8\v8.

3. Copy the Toolsx64.cab file from the DaRT administrative installation to the *installation_folder*\Templates\Distribution\Tools\x64 folder (where *installation_folder* is the folder where you installed MDT).

   > **Tip**
   >
   > By default the administrative installation of DaRT installs the Toolsx64.cab file in C:\Program Files\Microsoft DaRT 8\v8.

4. Run the Create Boot Image using MDT wizard to generate the boot image.

   While running the Create Boot Image using MDT wizard, on the **General Settings: Components** wizard page, select the **Microsoft Diagnostics and Recovery Toolkit 8 (DaRT 8)** check box.

   For information about how to run the Create Boot Image using MDT wizard for Configuration Manager, see Creating ZTI Boot Images in Configuration Manager

5. Close all open windows and dialog boxes.

Preparing the MDT Migration Resources

During deployment to the target computers, deployment scripts connect to the deployment shares and shared folders. Create accounts for the scripts to use when accessing these resources.

Prepare the MDT migration resources by:

- Creating additional shared folders as described in Creating Additional Shared Folders
- Configuring shared folder permissions as described in Configuring Shared Folder Permissions

- Configuring access to other resources as described in Configuring Access to Other Resources

Creating Additional Shared Folders

Before starting the deployment, create additional shared folders in which to store the user state migration data and the deployment logs. Table 198 lists the shared folders that must be created and describes the purpose of each.

Table 198. Shared Folders and Their Descriptions

Expand table

Shared folder	Description
MigData	Stores the user state migration data during the LTI deployment process.
Logs	Stores the deployment logs during the LTI or ZTI deployment process. This folder is optional for either deployment.

> **Note**
>
> The files in Table 198 are recommended shared folder names. Use any name for these shared folders. However, the remainder of the deployment process refers to these shared folders by these names.

Configuring Shared Folder Permissions

After creating additional shared folders listed in Table 198, configure the appropriate shared folder permissions. Ensure that unauthorized users are unable to access user state migration information and the deployment logs. Only the target computer creating the user state migration information and the deployment logs should have access to these folders.

To configure the shared folder permissions for the folders listed in Table 198

1. In Windows Explorer, right-click ***shared_folder*** (where *shared_folder* is one of the shared folders listed in Table 198), and then select **Properties**.

2. On the **Security** tab, select **Advanced**.
3. On the **Permissions** tab, clear the **Allow inheritable permissions from the parent to propagate to this object and all child objects** check box.
4. In the **Security** dialog box, select **Remove**.
5. On the **Permissions** tab, select **Add**.
6. In the **Enter the object name to select** box, type **Authenticated Users**, and then select **OK**.
7. In the **Permission Entry for *shared_folder*** dialog box (where *shared_folder* is one of the shared folders listed in Table 198), in the **Apply onto** list, select **This folder only**.
8. In the **Permission Entry for *shared_folder*** dialog box (where *shared_folder* is one of the shared folders listed in Table 198), in the **Permissions** list, select **Allow for the Create Folders/Append Data permission**, and then select **OK**.
9. On the **Permissions** tab, select **Add**.
10. In the **Enter the object name to select** box, type **CREATOR OWNER**, and then select **OK**.

    This action allows domain computers and domain users to access the subfolders they create.
11. In the **Permission Entry for *shared_folder*** dialog box (where *shared_folder* is one of the shared folders listed in Table 198), in the **Apply onto** list, select **Subfolders and files only**.
12. In the **Permission Entry for *shared_folder*** dialog box (where *shared_folder* is one of the shared folders listed in Table 198), in the **Permissions** list, select **Allow for the Full Control permission**, and then select **OK**.
13. Repeat steps 10-13 for each group that will receive Administrator privileges.

    The permissions set in these steps work for both LTI and ZTI deployments. In some instances, you may want to further restrict the user accounts that can access the shared folder. You can restrict user accounts for:

- LTI deployments by substituting **Authenticated Users** in the steps above with each account you want to have access

- ZTI deployments by substituting **Authenticated Users** in the steps above with a network access account in Configuration Manager

Configuring Access to Other Resources

In addition to the shared folders just created, the MDT scripts might require access to other resources. The resources include application or database servers (such as Microsoft SQL Server or Microsoft Exchange Server).

Access is granted to the credentials specified in the:

- **UserID**, **UserPassword**, and **UserDomain** properties for LTI deployments
- Deployment Wizard for LTI deployments
- Network access account used by the Configuration Manager client for ZTI deployments

  Grant access to the following resources:

- **MDT deployment share for LTI deployments**. Configure access to the deployment share created in the Deployment Workbench.
- **Any resources accessed using the ZTIConnect.wsf script for LTI or ZTI deployments**. Configure access to resources that are referenced using the ZTIConnect.wsf script.

  > ⓘ **Note**
  >
  > For further guidance on using the ZTIConnect.wsf script, see the MDT document *Microsoft Deployment Toolkit Samples Guide*.

- **Any resources on application or database servers for LTI or ZTI deployments**. Configure access to applications or databases that are accessed through the **SQLServer**, **SQLShare**, and **Database** properties.

  > ⓘ **Note**
  >
  > Other connections to the same servers, such as Named Pipes and RPC, use the same credentials listed above. Use the ZTIConnect.wsf script to establish these connections.

Preparing Windows Deployment Services

You can use Windows Deployment Services in conjunction with MDT to automatically initiate boot images on target computers. These boot images can be Windows PE images or custom images that can deploy operating systems directly to the target computers.

Prepare Windows Deployment Services for use with MDT by:

- Preparing Windows Deployment Services for use in LTI deployments as described in Preparing Windows Deployment Services for LTI Deployments
- Preparing Windows Deployment Services for use in ZTI deployments using Configuration Manager as described in Preparing Windows Deployment Services for ZTI Deployments Using Configuration Manager
- Preparing Windows Deployment Services for use in UDI deployments as described in Preparing Windows Deployment Services for UDI Deployments

Preparing Windows Deployment Services for LTI Deployments

You can use Windows Deployment Services in LTI deployments in the following ways:

- **Start Windows PE on the target computers**. The beginning of the New Computer deployment scenario and the second half of the Replace Computer deployment scenario both start the target computer in Windows PE. For these scenarios, you can automate starting Windows PE using Windows Deployment Services.
- **Install images created in the Deployment Workbench on the target computers**. You can create custom WIM images in the Deployment Workbench that you can deploy directly to target computers using Windows Deployment Services.

  For more information about setting up and configuring Windows Deployment Services, see:

- Windows Deployment Services
- The Windows Deployment Services Help file, included in Windows Deployment Services

  Prepare Windows Deployment Services for LTI deployments by:

- Adding boot images created in the Deployment Workbench for LTI deployments to Windows Deployment Services as described in Add LTI Boot Images to Windows Deployment Services
- Pre-staging target computers in Windows Deployment Services for LTI deployments as described in Pre-stage Target Computers for Windows Deployment Services for LTI Deployments
- Enabling Windows Deployment Services multicast deployment of images in Windows Deployment Services for LTI deployments as described in Enable Windows Deployment Services Multicast Deployment for LTI Deployments

Add LTI Boot Images to Windows Deployment Services

You can add the LTI boot image WIM files in the Boot folder of a deployment share to Windows Deployment Services. Doing so allows Windows Deployment Services to automatically initiate LTI deployment by starting LTI boot images.

> **Note**
>
> Add the LTI boot images only to Windows Deployment Services. You do not need to add operating system images from the Deployment Workbench.

You can add LTI boot images to Windows Deployment Services using the Windows Deployment Services management console or the WDSUTIL.exe tool.

For more information about adding an LTI boot image to Windows Deployment Services, see:

- "Add an Image," in Windows Deployment Services Help, which is included in Windows Deployment Services
- Windows Deployment Services Getting Started Guide

Pre-stage Target Computers for Windows Deployment Services for LTI Deployments

You can pre-stage PXE client computers in AD DS domains. When target computers are pre-staged, the computer accounts exist in AD DS domains (also called *known computers*). Target computers that are not pre-staged do not have computer accounts in AD DS domains (also called unknown computers).

> ⓘ **Note**
>
> Responding to unknown computers is the preferred method for LTI deployments, because it is the simplest method. If you pre-stage the target computers, LTI is unable to use the pre-staged computer account. Only Windows Deployment Services can use the pre-staged computer accounts.

You can configure Windows Deployment Services to respond to computers that are known or unknown. Depending on the Windows Deployment Services configuration, you may need to pre-stage the target computers. Doing so authorizes Windows Deployment Services to deploy operating system images to the target computer.

> ⓘ **Note**
>
> If Windows Deployment Services is configured to respond to any computer (known or unknown), pre-staging the target computers is not necessary. LTI will not use a pre-staged computer account when joining the domain. Instead, LTI uses the computer name and credentials configured in the task sequence or through the rules process.

To pre-stage the target computers for Windows Deployment Services

1. Select **Start**, point to **Administrative Tools**, and then select **Active Directory Users and Computers**.
2. In the console tree, right-click ***organizational_unit*** (where *organizational_unit* is the name of the OU that will contain the target computer), point to **New**, and then select **Computer**.
3. In the **New Object - Computer** dialog box, in the **Computer name** box, type ***computer_name*** (where *computer_name* is the name of the target computer), and then select **Next**.
4. In the **Managed** dialog box, select the **This is a managed computer** check box. In the **Computer's unique ID (GUID/UUID)** box, type ***guid_uuid*** (where *guid_uuid* is the GUID/UUID of the computer), and then select **Next**.
5. In the **Host server** dialog box, select one of the following options, and then select **Next**:

- **Any available remote installation server**. This option specifies that this computer can be serviced by any Windows Deployment Services server.
- **The following remote installation server**. This option designates a specific server to service the computer. Specify the FQDN of the server running Windows Deployment Services.

6. In the **New Object - Computer** dialog box, review the information displayed, and then select **Finish**.
7. Close all open windows.

Enable Windows Deployment Services Multicast Deployment for LTI Deployments

Multicast deployment of LTI operating systems using Windows Deployment Services allows multiple computers to receive a single copy of an image, which reduces the amount of network traffic required when multiple computers need to receive the same image. By default, multicasting support is disabled in MDT.

For LTI deployments, the Deployment Workbench creates a multicast namespace for the deployment share. The images are transferred to the target computers using multicast from the deployment share, not from a Windows Deployment Services share.

> ⓘ **Note**
>
> MDT supports only the multicast transfer of images stored in the LTI$ distribution share. Images stored in Windows Deployment Services cannot be deployed using multicast transfer.

The multicast types available for use in LTI deployments include:

- **Auto-Cast**. In this option, as soon as an applicable client computer requests an installation image, a multicast transmission of the selected image begins. Then, as other client computers request the same image, they are joined to the transmission that has already started.
- **Scheduled-Cast**. This option sets the start criteria for the transmission based on the number of client computers requesting an image or a specific day and time.

  MDT supports multicast-based deployments when MDT and the deployment share are installed on:

- **The computer running Windows Deployment Services**. In this scenario, MDT is installed on a computer running Windows Server with the Windows Deployment Services role as described in Enable Multicast Deployments with MDT Installed on the Same Computer as Windows Deployment Services.
- **A computer other than the computer Windows Deployment Services is running on**. In this scenario, MDT and the deployment share are installed on a separate computer from the one running Windows Server with the Windows Deployment Services role as described in Enable Multicast Deployments with MDT Installed on a Different Computer from Windows Deployment Services.

  > ⓘ **Note**
  >
  > If MDT is installed on a separate computer, you must install the Remote Server Administration Tools feature so that the WDSUTIL command-line utility is available.

> ⓘ **Note**
>
> You cannot use these scenarios to allow multicast for boot images, as the multicast client is not loaded until after Windows PE is running. LTI only uses multicast to transfer operating system WIM files.

Enable Multicast Deployments with MDT Installed on the Same Computer as Windows Deployment Services

In this scenario, MDT is installed on a computer running Windows Server with the Windows Deployment Services server role. In this scenario, MDT can automatically configure Windows Deployment Services to support multicast deployments.

To enable multicast deployments with MDT installed on the same computer as Windows Deployment Services

1. Install Windows Server on the computer that is to be the deployment server.
2. Install the Windows Deployment Services server role on the computer that is to be the deployment server.
3. Install Windows ADK for Windows 8.1 on the computer that is to be the deployment server.

4. Install MDT on the computer that is to be the deployment server.
5. Start the Deployment Workbench.
6. In the Deployment Workbench console tree, select **Deployment Shares**.
7. In the details pane, right-click ***deployment_share*** (where *deployment_share* is the name of an existing deployment share for which multicast deployments will be enabled), and then select **Properties**.
8. In the ***deployment_share*** **Properties** dialog box (where *deployment_share* is the name of an existing deployment share), on the **General** tab, select the **Enable multicast for this deployment share** check box, and then select **OK**.

   > ⓘ **Note**
   >
   > The **Network (UNC) path** and **Local Path** text boxes on the **General** tab must contain valid paths for multicasting to function properly.

9. In the Actions pane, select **Update Deployment Share**.

   When completed, the Deployment Workbench creates an Auto-Cast Windows Deployment Services multicast transmission from the deployment share.

Enable Multicast Deployments with MDT Installed on a Different Computer from Windows Deployment Services

In this scenario, Windows Deployment Services and Windows Server are all installed on the computer acting as the deployment server, but MDT is installed on another computer. In this configuration, remotely run the **WDSUTIL** command on the computer running Windows Deployment Services and Windows Server.

To enable multicast deployments with MDT Installed on a different computer than Windows Deployment Services

1. Install Windows Server on the computer hosting the deployment share.
2. Install the Windows Deployment Services server role on the computer hosting the deployment share.
3. Install Windows ADK for Windows 8.1 on a computer other than the computer hosting the deployment share.
4. Install MDT on the same computer as in step 3.

5. On the computer hosting the deployment share, in a Command Prompt window, type the following command, and then press ENTER (where *remote_server* is the name of the computer running Windows Server with the Windows Deployment Services server role and *deploy_share_path* is the fully qualified path to the root of the deployment share).

   ```
   wdsutil.exe /new-namespace /friendlyname:"BDD Share Deploy$"
   /server:remote_server /namespace:"Deploy$" /contentprovider:WDS
   /configstring:"deploy_share_path" /namespacetype:AutoCast
   ```

   When completed, the WDSUTIL tool creates an Auto-Cast Windows Deployment Services multicast transmission from the deployment share.

Preparing Windows Deployment Services for ZTI Deployments Using Configuration Manager

For ZTI deployments using Configuration Manager, configure a Configuration Manager PXE service point on the computer on which Windows Deployment Services is installed. Doing so allows Configuration Manager to directly service PXE boot requests received by Windows Deployment Services as a PXE service point, which in turn allows target computers to boot images that Configuration Manager manages using PXE. The PXE service point is a feature of the distribution point site system role, which means that you will configure the computer running Windows Deployment Services as a distribution point site system role.

For more information about preparing Windows Deployment Services for ZTI deployments using Configuration Manager, see:

- "How to Deploy Operating Systems by Using PXE in Configuration Manager" in the Configuration Manager Documentation Library, included with Configuration Manager
- "Configuring Distribution Points to Accept PXE Requests" in the Configuration Manager Documentation Library, included with Configuration Manager

> ⓘ **Note**
>
> In addition to the methods described here, you can use traditional Windows Deployment Services methods for responding to PXE boot requests. For more information, see the Windows Deployment Services Help file included with Windows Deployment Services.

Preparing Windows Deployment Services for UDI Deployments

Prepare Windows Deployment Services for UDI deployments using the same process for ZTI deployments as described in Preparing Windows Deployment Services for ZTI Deployments Using Configuration Manager.

Feedback

Was this page helpful? Yes No

Provide product feedback

Microsoft Deployment Toolkit Samples Guide

Article • 02/12/2024

This guide is part of Microsoft Deployment Toolkit (MDT) and guides a specialist team through deploying Windows operating systems and Microsoft Office. Specifically, this guide is designed to provide sample configuration settings for specific deployment scenarios.

> **Note**
>
> In this article, *Windows* applies to the Windows 8.1, Windows 8, Windows 7, Windows Server 2012 R2, Windows Server 2012, and Windows Server 2008 R2 operating systems unless otherwise noted. MDT does not support ARM processor-based versions of Windows. Similarly, *MDT* refers to the current version of MDT unless otherwise stated.

To use this guide

Review the list of scenario topics in the Table of Contents.

1. Select the scenario that most closely represents your organization's deployment goals.
2. Review the sample configuration settings for the selected scenario.
3. Use the sample configuration settings as the foundation for the configuration settings in your environment.
4. Customize the sample configuration settings for your environment.

   In many instances, more than one scenario might be necessary to complete the configuration settings for the environment.

   Because this guide contains only sample configuration settings, reviewing the guides listed in the following table can further assist in customizing the configuration settings for the environment.

Expand table

Guide	This guide offers assistance to help
Quick Start Guide for Microsoft System Center 2012 R2 Configuration Manager	Use System Center 2012 R2 Configuration Manager to install the Windows 8.1 operating system in a New Computer deployment scenario.
Quick Start Guide for Lite Touch Installation	Install the Windows 8.1 operating system through Lite Touch Installation (LTI) using bootable media in a New Computer deployment scenario.
Quick Start Guide for User-Driven Installation	Install the Windows 8.1 operating system with User-Driven Installation and System Center 2012 R2 Configuration Manager in a New Computer deployment scenario.
Using the Microsoft Deployment Toolkit	Further customize the configuration files used in Zero Touch Installation (ZTI) and LTI deployments. This guide also provides generic configuration guidance and a technical reference for configuration settings.

Deploying Windows 8 Applications Using MDT

MDT can deploy Windows 8 application packages, which have an .appx file extension. These application packages are new to Windows 8. For more information on these applications, see Windows Store App development.

Deploy Windows 8 applications using MDT by performing the following steps:

- Deploy Windows 8 applications using LTI as described in Deploying Windows 8 Applications Using LTI.
- Deploy Windows 8 applications using User-Driven Installation (UDI) as described in Deploying Windows 8 Applications Using UDI.

Deploying Windows 8 Applications Using LTI

You can deploy Windows 8 applications using LTI like any other application that initiates the installation process from a command line. You can add Windows 8 applications to LTI deployments in the Applications node in the Deployment Workbench.

To deploy a Windows 8 application using LTI

1. Create a network shared folder in which to store the application.
2. Copy the Windows 8 application into the network shared folder that you created in the previous step.

Ensure that you copy the Windows 8 application .appx file and any other required files, such as a .cer file that contains the application certificate.

3. Create an LTI application item for the Windows 8 application in the Applications node in the Deployment Workbench using the New Application Wizard.

   While completing the New Application Wizard, on the **Command Details** wizard page, in **Command line**, type **app_file_name** (where *app_file_name* is the name of the Windows 8 application).

   For more information about how to complete the New Application Wizard in the Deployment Workbench, see the following sections in the MDT document, *Using the Microsoft Deployment Toolkit*:

   - "Create a New Application That Is Deployed from the Deployment Share"
   - "Create a New Application That Is Deployed from Another Network Shared Folder"

4. Select the LTI application item created in the previous step in an LTI task sequence.

Deploying Windows 8 Applications Using UDI

You can deploy Windows 8 applications using UDI like any other application that initiates the installation process from a command line. You can add Windows 8 applications to UDI deployments on the **ApplicationPage** wizard page in the UDI Wizard Designer.

> ⓘ **Note**
>
> Deployment of Windows 8 and Windows 8 applications using UDI requires System Center 2012 R2 Configuration Manager.

To deploy a Windows 8 application using UDI

1. Create a network shared folder in which to store the application.

   This folder will be the source folder for the Configuration Manager application that you will create later in the process.

2. Copy the Windows 8 application into the network shared folder that you created in the previous step.

Ensure that you copy the Windows 8 application .appx file and any other required files, such as a .cer file that contains the application certificate.

3. Add the Windows 8 application as a Configuration Manager application

4. Create a Configuration Manager application item for the Windows 8 application using the Create Application Wizard in the Configuration Manager console.

   While completing the Create Application Wizard, create a deployment type to deploy the Windows 8 application using the Create Deployment Type Wizard. In the Create Deployment Type Wizard, on the **Content** page, in **Installation program**, type **app_file_name** (where *app_file_name* is the name of the Windows 8 application).

   For more information about how to complete the Create Application Wizard in the Configuration Manager console, see the following sections in the Documentation Library for System Center 2012 Configuration Manager, which is included with Configuration Manager:

   - How to Create Applications in Configuration Manager
   - How to Create Deployment Types in Configuration Manager
   - How to Manage Applications and Deployment Types in Configuration Manager

5. Ensure that the user device affinity (UDA) feature in Configuration Manager is configured properly to support affinity between users and devices for Configuration Manager application deployment.

   For more information about how to configure UDA to support Configuration Manager application deployment, see How to Manage User Device Affinity in Configuration Manager.

6. Deploy the application created in step 4 to the targeted users.

   For more information about how to deploy an application to user, see How to Deploy Applications in Configuration Manager.

7. Configure the **ApplicationPage** wizard page to include the Configuration Manager application created in step 4 using the UDI Wizard Designer.

   For more information about how to configure the **ApplicationPage** wizard page using the UDI Wizard Designer, see the section, "Step 5-11: Customize the UDI

Wizard Configuration File for the Target Computer", in the MDT document *Quick start Guide for User-Driven Installation*.

8. Select the UDI application item created in the previous step in a UDI task sequence.

   > **Note**
   >
   > The Windows 8 application is not installed by the task sequence but rather will be installed the first time the user logs on to the targeted computer (as defined by the UDA setting configured in step 5) using the User-Centric App Installer feature (AppInstall.exe) in UDI.

   For more information on the User-Centric App Installer feature in UDI, see the section, "User-Centric App Installer Reference", in the MDT document *Toolkit Reference*.

Managing MDT Using Windows PowerShell

You can manage MDT deployment shares using the Deployment Workbench and Windows PowerShell. MDT includes a Windows PowerShell™ snap-in—Microsoft.BDD.SnapIn—that must be loaded prior to using the MDT-specific features in Windows PowerShell. The MDT Windows PowerShell snap-in includes:

- A Windows PowerShell provider—MDTProvider—that provides access to the contents of a deployment share
- Cmdlets that provide the ability to administer MDT deployment shares

  Manage MDT deployment shares using Windows PowerShell by performing the following steps:

- Load the MDT Windows PowerShell snap-in as described in Loading the MDT Windows PowerShell Snap-In.
- Create a deployment share using Windows PowerShell as described in Creating a Deployment Share Using Windows PowerShell.
- View deployment share properties using Windows PowerShell as described in Viewing Deployment Share Properties Using Windows PowerShell.
- View the list of deployment shares using Windows PowerShell as described in Viewing the List of Deployment Shares Using Windows PowerShell.

- Update a deployment share, which generates new Windows Preinstallation Environment (Windows PE) boot images, as described in Updating a Deployment Share Using Windows PowerShell.
- Update a linked deployment share, which replicates content from a deployment share to the linked deployment share, as described in Updating a Linked Deployment Share Using Windows PowerShell.
- Update deployment media, which replicates content from a deployment share to the deployment media, and then generates new bootable images as described in Updating Deployment Media Using Windows PowerShell.
- Manage items in a deployment share (such as operating systems, operating system packages, applications, and device drivers) as described in Managing Items in a Deployment Share Using Windows PowerShell.
- Automate the population of items in a deployment share (such as operating systems, operating system packages, applications, and device drivers) as described in Automating Population of a Deployment Share.
- Manage the folders in a deployment share using Windows PowerShell as described in Managing Deployment Share Folders Using Windows PowerShell.

Loading the MDT Windows PowerShell Snap-In

The MDT cmdlets are provided in a Windows PowerShell snap-in **Microsoft.BDD.SnapIn** that must be loaded prior to using the MDT cmdlets. Load the MDT Windows PowerShell snap-in using the **Add-PSSnapIn** cmdlet as described in Load the MDT Windows PowerShell Snap-In Using the Add-PSSnapIn Cmdlet.

Load the MDT Windows PowerShell Snap-In Using the Add-PSSnapIn Cmdlet

You can load the MDT Windows PowerShell snap-in **Microsoft.BDD.PSSnapIn** from any Windows PowerShell environment using the Add-PSSnapIn cmdlet, as show in the following example:

PowerShell

```
Add-PSSnapin -Name Microsoft.BDD.PSSnapIn
```

Creating a Deployment Share Using Windows PowerShell

You can create deployment shares using the MDT Windows PowerShell cmdlets. The root folder for the deployment share is created and shared using standard Windows PowerShell cmdlets and calls to Windows Management Instrumentation (WMI) class commands. The deployment share is populated using the MDTProvider Windows PowerShell provider and the NewPSDrive cmdlet. The MDTProvider Windows PowerShell drive is persisted using the **Add-MDTPersistentDrive** cmdlet.

To prepare a deployment share using the MDT Windows PowerShell cmdlets

1. Load the MDT Windows PowerShell snap-in as described in Loading the MDT Windows PowerShell Snap-In.

2. Create the folder that will be the root of the new deployment share using the **New-Item** cmdlet, as shown in the following example and described in Using the New-Item Cmdlet:

   ```
   PowerShell

   New-Item "C:\MDTDeploymentShare$" -Type directory
   ```

   The cmdlet displays the successful creation of the folder.

3. Share the folder created in the previous step using the **WMI win32_share** class as sown in the following example:

   ```
   PowerShell

   ([wmiclass]"win32_share").Create("C:\MDTDeploymentShare$",
   "MDTDeploymentShare$",0)
   ```

   The call to the **win32_share** class returns the results of the call. If the value of **ReturnValue** is zero (0), then the call was successful.

4. Specify the new shared folder as a deployment share using the NewPSDrive cmdlet, as shown in the following example:

   ```
   PowerShell

   New-PSDrive -Name "DS002" -PSProvider "MDTProvider" -Root
   "C:\MDTDeploymentShare$" -Description "MDT Deployment Share Created
   with Cmdlets" -NetworkPath "\\WDG-MDT-01\MDTDeploymentShare$" -Verbose
   ```

   The cmdlet automatically starts creating the deployment share and copying the template information into the new deployment share. Upon completion of the

copy process, the cmdlet displays the information for the new deployment share.

> **Note**
>
> The value provided in the *Name* parameter (DS002) must be unique and cannot be the same as an existing deployment share Windows PowerShell drive.

5. Verify that the appropriate deployment share folders have been created using the **dir** command, as show in the following example:

```
PowerShell

Get-ChildItem ds002:
```

The list of default folders in the root of the deployment share is displayed.

6. Add the new deployment share to the list of persisted MDT deployment shares using the **Add-MDTPersistentDrive** cmdlet, as shown in the following example:

```
PowerShell

$NewDS=Get-PSDrive "DS002"
Add-MDTPersistentDrive  -Name "DS002" -InputObject $NewDS Verbose
```

In this example, the *$NewDS* variable is used to pass the Windows PowerShell drive object for the new deployment share to the cmdlet.

Alternatively, you could have combined the NewPSDrive and **Add-MDTPersistentDrive** cmdlets, as shown in the following example:

```
PowerShell

New-PSDrive -Name "DS002" -PSProvider "MDTProvider" -Root
"C:\MDTDeploymentShare$" -Description "MDT Deployment Share Created
with Cmdlets" -NetworkPath "\\WDG-MDT-01\MDTDeploymentShare$" -Verbose
| Add-MDTPersistentDrive -Verbose
```

In the previous example, the Windows PowerShell pipeline provides both the *Name* and *InputObject* parameters.

Viewing Deployment Share Properties Using Windows PowerShell

You can view the properties of MDT deployment shares using the Get-ItemProperty cmdlet and the MDTProvider Windows PowerShell provider. These same properties can also be seen in the Deployment Workbench.

To view deployment share properties using the MDT Windows PowerShell cmdlets

1. Load the MDT Windows PowerShell snap-in as described in Loading the MDT Windows PowerShell Snap-In.

2. Ensure the MDT deployments share Windows PowerShell drives are restored using the **Restore-MDTPersistentDrive** cmdlet, as shown in the following example:

   ```
   PowerShell
   Restore-MDTPersistentDrive -Verbose
   ```

   > **Note**
   >
   > If the MDT deployments that share Windows PowerShell drives are already restored, you will receive a warning message indicating that the cmdlet is unable to restore the drive.

3. Verify that the MDT deployments that share Windows PowerShell drives are restored properly using the Get-PSDrive cmdlet, as follows:

   ```
   PowerShell
   Get-PSDrive -PSProvider Microsoft.BDD.PSSnapIn\MDTProvider
   ```

   The list of Windows PowerShell drives that are provided using the MDTProvider are listed.

4. View the properties of the deployment share using the Get-ItemProperty cmdlet, as shown in the following example:

   ```
   PowerShell
   Get-ItemProperty "DS002:"
   ```

   In this example, *DS002:* is the name of a Windows PowerShell drive returned in step 3. The cmdlet returns the properties for the deployment share.

Viewing the List of Deployment Shares Using Windows PowerShell

You can view the list of MDT deployment shares using the Get-PSDrive cmdlet and the MDTProvider Windows PowerShell provider. The same list of deployment shares can also be viewed in the Deployment Workbench.

To view a list of deployment shares using the MDT Windows PowerShell cmdlets

1. Load the MDT Windows PowerShell snap-in as described in Loading the MDT Windows PowerShell Snap-In.
2. Ensure that the MDT deployments share Windows PowerShell drives are restored using the **Restore-MDTPersistentDrive** cmdlet, as shown in the following example:

   ```
   PowerShell
   Restore-MDTPersistentDrive -Verbose
   ```

   > **Note**
   >
   > If the MDT deployments that share Windows PowerShell drives are already restored, you will receive a warning message indicating that the cmdlet is unable to restore the drive.

3. View the list of MDT deployments that share Windows PowerShell drives, one for each deployment share, using the Get-PSDrive cmdlet, as follows:

   ```
   PowerShell
   Get-PSDrive -PSProvider Microsoft.BDD.PSSnapIn\MDTProvider
   ```

   The list of Windows PowerShell drives provided using the MDTProvider are listed, one for each deployment share.

Updating a Deployment Share Using Windows PowerShell

You can update deployment shares using the **Update-MDTDeploymentShare** cmdlet and the MDTProvider Windows PowerShell provider. Updating a deployment share creates the Windows PE boot images (WIM and International Organization for

Standardization [ISO] files) necessary to start LTI deployment. You can perform the same process using the Deployment Workbench, as described in "Update a Deployment Share in the Deployment Workbench".

To update a deployment share using Windows PowerShell

1. Load the MDT Windows PowerShell snap-in as described in Loading the MDT Windows PowerShell Snap-In.

2. Ensure that the MDT deployments that share Windows PowerShell drives are restored using the **Restore-MDTPersistentDrive** cmdlet, as shown in the following example:

   ```
   PowerShell
   Restore-MDTPersistentDrive -Verbose
   ```

   > **Note**
   >
   > If the MDT deployments that share Windows PowerShell drives are already restored, you will receive a warning message indicating that the cmdlet is unable to restore the drive.

3. Verify that the MDT deployments that share Windows PowerShell drives are restored properly using the Get-PSDrive cmdlet, as follows:

   ```
   PowerShell
   Get-PSDrive -PSProvider Microsoft.BDD.PSSnapIn\MDTProvider
   ```

   The list of Windows PowerShell drives provided using the MDTProvider are listed.

4. Update the deployment share using the **Update-MDTDeploymentShare** cmdlet, as shown in the following example:

   ```
   PowerShell
   Update-MDTDeploymentShare -Path "DS002:" -Force
   ```

   In this example, *DS002:* is the name of a Windows PowerShell drive returned in step 3.

> **Note**
>
> Updating the deployment share can take a long time. The progress of the cmdlet is shown at the top of the Windows PowerShell console.

The cmdlet returns with no output if the update is successful.

Updating a Linked Deployment Share Using Windows PowerShell

You can update (replicate) linked deployment shares using the **Update-MDTLinkedDS** cmdlet and the MDTProvider Windows PowerShell provider. Updating a linked deployment share replicates the content from the original deployment share to the linked deployment share. You can perform the same process using the Deployment Workbench, as described in "Replicate Linked Deployment Shares in the Deployment Workbench".

To update a linked deployment share using Windows PowerShell

1. Load the MDT Windows PowerShell snap-in as described in Loading the MDT Windows PowerShell Snap-In.
2. Ensure that the MDT deployments that share Windows PowerShell drives are restored using the **Restore-MDTPersistentDrive** cmdlet, as shown in the following example:

```
PowerShell
```

```
Restore-MDTPersistentDrive -Verbose
```

> **Note**
>
> If the MDT deployments that share Windows PowerShell drives are already restored, you will receive a warning message indicating that the cmdlet is unable to restore the drive.

3. Verify that the MDT deployments that share Windows PowerShell drives are restored properly using the Get-PSDrive cmdlet, as follows:

```
PowerShell
```

```
Get-PSDrive -PSProvider Microsoft.BDD.PSSnapIn\MDTProvider
```

The list of Windows PowerShell drives provided using the MDTProvider are listed.

4. Update the deployment share using the **Update-MDTDeploymentShare** cmdlet, as shown in the following example:

```
PowerShell
Update-MDTLinkedDS -Path "DS002:\Linked Deployment Shares\LINKED002"
```

In this example, *DS002:* is the name of a Windows PowerShell drive returned in step 3.

> **Note**
>
> Updating the linked deployment share can take a long time. The progress of the cmdlet is shown at the top of the Windows PowerShell console.

The cmdlet returns with no output if the update is successful.

Updating Deployment Media Using Windows PowerShell

You can update (generate) deployment media using the **Update-MDTMedia** cmdlet and the MDTProvider Windows PowerShell provider. Updating deployment media replicates the content from the original deployment share to the linked deployment share, and then generates .iso and .wim files. You can perform the same process using the Deployment Workbench, as described in "Generate Media Images in the Deployment Workbench".

When the **Update-MDTMedia** cmdlet finishes, the following files are created:

- An .iso file in the *media_folder* folder (where *media_folder* is the name of the folder that you specified for the media)

  Generating the .iso file is an option that you configure by:

  - Selecting the **Generate a Lite Touch bootable ISO image** check box on the **General** tab of the **media Properties** dialog box (Clear this check box to reduce the time needed to generate the media unless you need to create bootable DVDs or start virtual machines [VMs] from the .iso file.)

- Setting the same property using the Set-ItemProperty cmdlet

- WIM files in the *media_folder*\Content\Deploy\Boot folder (where *media_folder* is the name of the folder that you specified for the media)

To update a linked deployment share using Windows PowerShell

1. Load the MDT Windows PowerShell snap-in as described in Loading the MDT Windows PowerShell Snap-In.

2. Ensure that the MDT deployments share Windows PowerShell drives are restored using the **Restore-MDTPersistentDrive** cmdlet, as shown in the following example:

```
PowerShell

Restore-MDTPersistentDrive -Verbose
```

> **Note**
>
> If the MDT deployments that share Windows PowerShell drives are already restored, you will receive a warning message indicating that the cmdlet is unable to restore the drive.

3. Verify that the MDT deployments that share Windows PowerShell drives are restored properly using the Get-PSDrive cmdlet, as follows:

```
PowerShell

Get-PSDrive -PSProvider Microsoft.BDD.PSSnapIn\MDTProvider
```

The list of Windows PowerShell drives provided using the MDTProvider are listed.

4. Update the deployment share using the **Update-MDTDeploymentShare** cmdlet, as shown in the following example:

```
PowerShell

Update-MDTLinkedDS -Path "DS002:\Linked Deployment Shares\LINKED002"
```

In this example, *DS002:* is the name of a Windows PowerShell drive returned in step 3.

> **Note**

> Updating the linked deployment share can take a long time. The progress of the cmdlet is shown at the top of the Windows PowerShell console.

The cmdlet returns with no output if the update is successful.

Managing Items in a Deployment Share Using Windows PowerShell

A deployment share contains items that are used to perform deployments, such as operating systems, applications, device drivers, operating system packages, and task sequences. These items can managed using cmdlets from Windows PowerShell and those provided with MDT.

For more information about manipulating items directly using Windows PowerShell cmdlets, see Manipulating Items Directly. The folder structure for a deployment share can also be managed using Windows PowerShell. For more information, see Managing Deployment Share Folders Using Windows PowerShell.

Import an Item into a Deployment Share

You can import each type of item, such as operating systems, applications, or device drivers, using MDT cmdlets. For each type of item, there is a specific MDT cmdlet. If you want to import multiple items into a deployment share using Windows PowerShell, see Automating Population of a Deployment Share.

The following table lists the MDT Windows PowerShell cmdlets used to import items into a deployment share and provides a brief description of each cmdlet. Examples of how to use each cmdlet is provided in the section that corresponds to each cmdlet.

Expand table

Cmdlet	Description
Import-MDTApplication	Imports an application into a deployment share
Import-MDTDriver	Imports one or more device drivers into a deployment share
Import-MDTOperatingSystem	Imports one or more operating systems into a deployment share
Import-MDTPackage	Imports one or more operating system packages into a deployment share
Import-MDTTaskSequence	Imports a task sequence into a deployment share

View the Properties of an Item in a Deployment Share

Each item in a deployment share has different set of properties. You can view the properties of an item in a deployment share using the Get-ItemProperty cmdlet. The Get-ItemProperty cmdlet uses the MDTProvider to display the properties for a specific item, just as you can see the properties in the Deployment Workbench.

If want wish to view the properties of multiple items in a deployment share using Windows PowerShell, see Automating Population of a Deployment Share.

To view the properties of an item in a deployment share using Windows PowerShell

1. Load the MDT Windows PowerShell snap-in as described in Loading the MDT Windows PowerShell Snap-In.

2. Ensure that the MDT deployments that share Windows PowerShell drives are restored using the **Restore-MDTPersistentDrive** cmdlet, as shown in the following example:

   ```
   PowerShell

   Restore-MDTPersistentDrive -Verbose
   ```

   > **Note**
   >
   > If the MDT deployments that share Windows PowerShell drives are already restored, you will receive a warning message indicating that the cmdlet is unable to restore the drive.

3. Verify that the MDT deployments that share Windows PowerShell drives are restored properly using the Get-PSDrive cmdlet, as shown in the following example:

   ```
   PowerShell

   Get-PSDrive -PSProvider Microsoft.BDD.PSSnapIn\MDTProvider
   ```

   The list of Windows PowerShell drives provided using the MDTProvider are listed.

4. Return a list of the items for the type of item for which you are wanting to view the properties using the Get-Item cmdlet, as shown in the following example:

   ```
   PowerShell
   ```

```
Get-Item "DS001:\Operating Systems\*" | Format-List
```

In the previous example, a list of all the operating systems in the deployment share is displayed. The output is piped to the **Format-List** cmdlet so that the long names of the operating systems can be seen. For more information on how to use the **Format-List** cmdlet, see Using the Format-List Cmdlet. The same process could be used to return the list of other types of items, such as device drivers or applications.

> **Tip**
>
> You could have also used the **dir** command to view the list of operating systems instead of the **Get-Item** cmdlet.

5. View the properties of one of the items listed in the previous step using the Get-ItemProperty cmdlet, as shown in the following example:

```
PowerShell

Get-ItemProperty -Path "DS002:\Operating Systems\Windows 8 in Windows 8 x64 install.wim"
```

In this example, the value of the *Path* parameter is the fully qualified Windows PowerShell path to the item, including the file name that was returned in the previous step. You could use the same process to view the properties of other types of items, such as device drivers or applications.

Remove an Item from a Deployment Share

You can remove an item from a deployment share using the Remove-Item cmdlet. The Remove-Item cmdlet uses the MDTProvider to remove a specific item, just as you can remove an item in the Deployment Workbench. If you want to remove multiple items in a deployment share using Windows PowerShell, see Automating Population of a Deployment Share.

> **Note**
>
> Removing an item that a task sequence uses causes the task sequence to fail. Ensure that an item is not referenced by other items in the deployment share prior to removing the item. Once an item is removed, it cannot be recovered.

To remove an item from a deployment share using Windows PowerShell

1. Load the MDT Windows PowerShell snap-in as described in Loading the MDT Windows PowerShell Snap-In.

2. Ensure that the MDT deployments that share Windows PowerShell drives are restored using the **Restore-MDTPersistentDrive** cmdlet, as shown in the following example.

   ```
   PowerShell

   Restore-MDTPersistentDrive -Verbose
   ```

   > **Note**
   >
   > If the MDT deployments that share Windows PowerShell drives are already restored, you will receive a warning message indicating that the cmdlet is unable to restore the drive.

3. Verify that the MDT deployments that share Windows PowerShell drives are restored properly using the Get-PSDrive cmdlet, as shown in the following example:

   ```
   PowerShell

   Get-PSDrive -PSProvider Microsoft.BDD.PSSnapIn\MDTProvider
   ```

   The list of Windows PowerShell drives provided using the MDTProvider are listed.

4. Return a list of the items for the type of item for which you are wanting to view the properties using the Get-Item cmdlet, as shown in the following example:

   ```
   PowerShell

   Get-Item "DS001:\Operating Systems\*" | Format-List
   ```

   In the previous example, a list of all the operating systems in the deployment share is displayed. The output is piped to the **Format-List** cmdlet so that the long names of the operating systems can be seen. For more information on how to use the **Format-List** cmdlet, see Using the Format-List Cmdlet. You could use the same process to return the list of other types of items, such as device drivers or applications.

> **Tip**
>
> You could have also used the **dir** command to view the list of operating systems instead of the **Get-Item** cmdlet.

5. Remove one of the items listed in the previous step using the Remove-Item cmdlet, as shown in the following example:

```
PowerShell

Remove-Item -Path "DS002:\Operating Systems\Windows 8 in Windows 8 x64 install.wim"
```

   In this example, the value of the *Path* parameter is the fully qualified Windows PowerShell path to the item, including the file name that was returned in the previous step.

   You could use the same process to remove other types of items, such as device drivers or applications.

> **Note**
>
> Removing an item that a task sequence uses causes the task sequence to fail. Ensure that an item is not referenced by other items in the deployment share prior to removing the item.

Automating Population of a Deployment Share

The MDT Windows PowerShell cmdlets allow you to manage individual items. However, by using some of the scripting features in Windows PowerShell, the cmdlets can be used to automate the population of a deployment share.

For example, an organization may need to deploy multiple deployment shares for different business units, or an organization may provide operating system deployment services for other organizations. In both of these examples, the organizations need the ability to create and populate deployment shares that are configured consistently.

One method for managing multiple items would be to use a comma-separated values (CSV) file that contains a list of all the items you want to manage in a deployment share using the Import-CSV cmdlet.

The following is an excerpt of a Windows PowerShell script to import a list of applications based on information in a .csv file using the Import-CSV, ForEach-Object, and **Import-MDTApplication** cmdlets:

PowerShell

```
$List=Import-CSV "C:\MDT\Import-MDT-Apps.csv"
ForEach-Object ($App in $List) {
     Import-MDTApplication -path $App.ApplicationFolder -enable "True" -Name
$App.DescriptiveName -ShortName $App.Shortname -Version $App.Version -
Publisher $App.Publisher -Language $App.Language -CommandLine
$App.CommandLine -WorkingDirectory $App.WorkingDirectory -
ApplicationSourcePath $App.SourceFolder -DestinationFolder
$App.DestinationFolder -Verbose
}
```

In this example, the C:\MDT\Import-MDT-Apps.csv file contains a field for each variable necessary to import an application. For more information about how to create a .csv file for use with the Import-CSV cmdlet, see Using the Import-Csv Cmdlet.

You can use this same method to import operating systems, device drivers, and other items in a deployment share by performing the following steps:

1. Create a .csv file for each type of deployment share item that you want to populate.

2. For more information about how to create a .csv file for use with the Import-CSV cmdlet, see Using the Import-Csv Cmdlet.

3. Create a Windows PowerShell script file that will be used to automate the population of the deployment share.

   For more information about how to create a Windows PowerShell script, see Scripting with Windows PowerShell.

4. Create any prerequisite folder structure required in the deployment share prior to importing the deployment share items.

   For more information, see Managing Deployment Share Folders Using Windows PowerShell.

5. Add the Import-CSV cmdlet line for one of the .csv files created in step 1.

   For more information on the Import-CSV cmdlet, see Using the Import-Csv Cmdlet.

6. Create a ForEach-Object cmdlet loop that processes each item from the .csv file referenced in the Import-CSV cmdlet in the previous step.

   For more information on the ForEach-Object cmdlet, see Using the ForEach-Object Cmdlet.

7. Add the corresponding MDT cmdlet for importing the deployment share items inside the ForEach-Object cmdlet loop created in the previous step.

   For more information on the MDT cmdlets used for importing items into a deployment share, see Import an Item into a Deployment Share.

Managing Deployment Share Folders Using Windows PowerShell

You can manage folders in a deployment share using command-line tools, such as the **mkdir** command, or using Windows PowerShell cmdlets, such as the New-Item cmdlet and the MDTProvider Windows PowerShell provider. The same folder structure of deployment shares can also be seen and managed in the Deployment Workbench. For more information about manipulating items directly using Windows PowerShell cmdlets, see Manipulating Items Directly.

Create a Folder in a Deployment Share Using Windows PowerShell

To create a folder in a deployment share using Windows PowerShell

1. Load the MDT Windows PowerShell snap-in as described in Loading the MDT Windows PowerShell Snap-In.

2. Ensure that the MDT deployments that share Windows PowerShell drives are restored using the **Restore-MDTPersistentDrive** cmdlet, as shown in the following example:

   ```
   PowerShell
   ```

   ```
   Restore-MDTPersistentDrive -Verbose
   ```

   > **Note**

If the MDT deployments that share Windows PowerShell drives are already restored, you will receive a warning message indicating that the cmdlet is unable to restore the drive.

3. View the list of MDT deployments that share Windows PowerShell drives, one for each deployment share, using the Get-PSDrive cmdlet as follows:

```
PowerShell

Get-PSDrive -PSProvider Microsoft.BDD.PSSnapIn\MDTProvider
```

   The list of Windows PowerShell drives provided using the MDTProvider are listed, one for each deployment share

4. Create a folder named *Windows_8* in the Operating Systems folder in a deployment share using the **New-Item** command, as shown in the following example:

```
PowerShell

New-Item "DS002:\Operating Systems\Windows_8"
```

   In this example, *DS002:* is the name of a Windows PowerShell drive returned in step 3.

5. Verify that the folder is created correctly by typing the following command:

```
PowerShell

Get-ChildItem "DS002:\Operating Systems"
```

   The Windows_8 folder and any other existing folders in the Operating Systems folder is displayed.

6. Create a folder named *Windows_7* folder in the Operating Systems folder in a deployment share using the New-Item cmdlet, as shown in the following example and described in Using the New-Item Cmdlet:

```
PowerShell

New-Item "DS002:\Operating Systems\Windows_7" -Type directory
```

   The cmdlet displays the successful creation of the folder.

7. Verify that the folder is created correctly by typing the following command:

```
PowerShell

Get-ChildItem "DS002:\Operating Systems"
```

   The Windows_7 folder and any other existing folders in the Operating Systems folder is displayed.

Delete a Folder in a Deployment Share Using Windows PowerShell

To delete a folder in a deployment share using Windows PowerShell

1. Load the MDT Windows PowerShell snap-in as described in Loading the MDT Windows PowerShell Snap-In.

2. Ensure that the MDT deployments that share Windows PowerShell drives are restored using the **Restore-MDTPersistentDrive** cmdlet, as shown in the following example:

```
PowerShell

Restore-MDTPersistentDrive -Verbose
```

   > **Note**
   >
   > If the MDT deployments that share Windows PowerShell drives are already restored, you will receive a warning message indicating that the cmdlet is unable to restore the drive.

3. View the list of MDT deployments that share Windows PowerShell drives, one for each deployment share, using the Get-PSDrive cmdlet as follows:

```
PowerShell

Get-PSDrive -PSProvider Microsoft.BDD.PSSnapIn\MDTProvider
```

   The list of Windows PowerShell drives provided using the MDTProvider are listed, one for each deployment share.

4. Delete (remove) a folder named *Windows_8* in the Operating Systems folder in a deployment share using the **New-Item** command, as shown in the following

example:

```
PowerShell
Remove-Item "DS002:\Operating Systems\Windows_8"
```

In this example, *DS002:* is the name of a Windows PowerShell drive returned in step 3.

5. Verify that the folder is removed correctly by typing the following command:

```
PowerShell
Get-ChildItem "DS002:\Operating Systems"
```

The Windows_8 folder is no longer displayed in the list of folders in the Operating Systems folder

6. Delete (remove) a folder named *Windows_7* folder in the Operating Systems folder in a deployment share using the Remove-Item cmdlet, as shown in the following example:

```
PowerShell
Remove-Item "DS002:\Operating Systems\Windows_7"
```

The cmdlet displays the successful removal of the folder.

7. Verify that the folder is created correctly by typing the following command:

```
PowerShell
Get-ChildItem "DS002:\Operating Systems"
```

The Windows_7 folder is no longer displayed in the list of folders in the Operating Systems folder.

Rename a Folder in a Deployment Share Using Windows PowerShell

To rename a folder in a deployment share using Windows PowerShell

1. Load the MDT Windows PowerShell snap-in as described in Loading the MDT Windows PowerShell Snap-In.

2. Ensure that the MDT deployments share Windows PowerShell drives are restored using the **Restore-MDTPersistentDrive** cmdlet, as shown in the following example:

```
PowerShell

Restore-MDTPersistentDrive -Verbose
```

> **Note**
>
> If the MDT deployments that share Windows PowerShell drives are already restored, you will receive a warning message indicating that the cmdlet is unable to restore the drive.

3. View the list of MDT deployments share Windows PowerShell drives, one for each deployment share, using the Get-PSDrive cmdlet as follows:

```
PowerShell

Get-PSDrive -PSProvider Microsoft.BDD.PSSnapIn\MDTProvider
```

   The list of Windows PowerShell drives provided using the MDTProvider are listed, one for each deployment share.

4. Rename a folder named *Windows_8* to *Win_8* in the Operating Systems folder in a deployment share using the **ren** command, as shown in the following example:

```
PowerShell

ren "DS002:\Operating Systems\Windows_8" "Win_8"
```

   In this example, *DS002:* is the name of a Windows PowerShell drive returned in step 3.

5. Verify that the folder is removed correctly by typing the following command:

```
PowerShell

Get-ChildItem "DS002:\Operating Systems"
```

   The Windows_8 folder is renamed to *Win_8*.

6. Rename a folder named *Windows_7* to *Win-7* in the Operating Systems folder in a deployment share using the Rename-Item cmdlet, as shown in the following example:

```
PowerShell

Rename-Item "DS002:\Operating Systems\Windows_7" "Win_7"
```

   The cmdlet displays the successful rename of the folder.

7. Verify that the folder is created correctly by typing the following command:

```
PowerShell

Get-ChildItem "DS002:\Operating Systems"
```

   The Windows_7 folder is renamed to *Win_7*.

Automating the Application of Operating System Service Packs in Deployment Shares

Operating system service packs are a normal part of the software life cycle. The existing operating systems in deployment shares need to be updated with these service packs to help ensure that newly deployed or refreshed computers are current with the latest security recommendations and configuration settings.

In instances where an organization has many deployment shares with multiple operating systems in each deployment share, the process for manually updating the operating systems in each deployment share with the service packs can be time consuming. The methods for automating the application of operating system service packs in deployment shares include:

- Copying updated source content that already contains the service pack (for example, Windows 7 with SP1 media) to the folder in the deployment share in which the existing operating system resides, as described in Automating the Application of Operating System Service Packs from Updated Source Media
- Applying the service pack to a reference computer, and then capturing an updated image from a reference computer, as described in Automating the Application of Operating System Service Packs Using a Reference Computer and Windows PowerShell

Automating the Application of Operating System Service Packs from Updated Source Media

You can automate the process of updating operating system service packs using Windows PowerShell when you have source media that include the service pack, such as having a DVD that has Windows 7 with SP1 already integrated.

For this method, the operating system source media with the service pack is copied over the existing operating system files without the service pack in the deployment share using Windows PowerShell.

To automate the application of operating system service packs from updates source media using Windows PowerShell

1. Load the MDT Windows PowerShell snap-in as described in Loading the MDT Windows PowerShell Snap-In.

2. Ensure that the MDT deployments that share Windows PowerShell drives are restored using the **Restore-MDTPersistentDrive** cmdlet, as shown in the following example:

   ```
   PowerShell
   ```

   ```
   Restore-MDTPersistentDrive -Verbose
   ```

   > **Note**
   >
   > If the MDT deployments that share Windows PowerShell drives are already restored, you will receive a warning message indicating that the cmdlet is unable to restore the drive.

3. View the list of MDT deployments share Windows PowerShell drives, one for each deployment share, using the Get-PSDrive cmdlet, as shown in the following example:

   ```
   PowerShell
   ```

   ```
   Get-PSDrive -PSProvider Microsoft.BDD.PSSnapIn\MDTProvider
   ```

   The list of Windows PowerShell drives provided using the MDTProvider are listed, one for each deployment share.

4. Remove the folder for the existing operating system from the deployment share using the Get-ChildItem and Remove-Item cmdlets, as shown in the following example:

   ```
   PowerShell

   Get-ChildItem "DS002:\Operating Systems\Windows 7" -recurse | Remove-
   Item -recurse -force
   ```

   In this example, *DS002:* is the name of a Windows PowerShell drive returned in step 3.

5. Copy the contents of the operating system source files that have the service pack integrated using the Copy-Item cmdlet, as shown in the following example:

   ```
   PowerShell

   Copy-Item "E:\*" -Destination "DS002:\Operating Systems\Windows 7"-
   Recurse -Force
   ```

   In this example, the operating system source files are on drive E, and *DS002:* is the name of a Windows PowerShell drive returned in step 3.

6. Update any MDT deployment media based on deployment share using **Update-MDTMedia** cmdlet.

   For more information about how to update MDT deployment media based on deployment share using **Update-MDTMedia** cmdlet, see Updating Deployment Media Using Windows PowerShell.

Automating the Application of Operating System Service Packs Using a Reference Computer and Windows PowerShell

You can automate the process of updating operating system service packs using Windows PowerShell when you have only the service pack that is not yet integrated with the operating system, such as having SP1 for Windows 7 not yet integrated with a Windows 7 image.

For this method, deploy the operating system without the service pack to a reference computer. Then, apply the service pack to the reference computer. Next, capture an operating system image of the reference computer. Finally, copy the captured .wim file

over the Install.wim file in the operating system in the deployment share using Windows PowerShell.

To automate the application of operating system service packs from updates source media using Windows PowerShell

1. Deploy the target operating system to a reference computer.

   For more information on how to deploy a reference computer, see the following resources in the MDT document, *Using the Microsoft Deployment Toolkit*:

   - "Preparing for LTI Deployment to the Reference Computer"
   - "Deploying To and Capturing an Image of the Reference Computer in LTI"

2. Install the desired service pack to the reference computer.

   For more information on how to install the service pack, see the documentation accompanying the service pack.

3. Capture an image of the reference computer by creating and deploying a task sequence based on the **Sysprep and Capture** task sequence template.

   For more information about creating a task sequence based on the **Sysprep and Capture** task sequence template, see "Create a New Task Sequence in the Deployment Workbench".

4. Load the MDT Windows PowerShell snap-in as described in Loading the MDT Windows PowerShell Snap-In.

5. Ensure the MDT deployments that share Windows PowerShell drives are restored using the **Restore-MDTPersistentDrive** cmdlet, as shown in the following example:

   ```
   PowerShell

   Restore-MDTPersistentDrive -Verbose
   ```

   > **Note**
   >
   > If the MDT deployments that share Windows PowerShell drives are already restored, you will receive a warning message indicating that the cmdlet is unable to restore the drive.

6. View the list of MDT deployments share Windows PowerShell drives, one for each deployment share, using the Get-PSDrive cmdlet, as shown in the following example:

   ```
   PowerShell

   Get-PSDrive -PSProvider Microsoft.BDD.PSSnapIn\MDTProvider
   ```

   The list of Windows PowerShell drives provided using the MDTProvider are listed, one for each deployment share.

7. Copy the .wim file captured in step 3 over the Install.wim file in the operating system in the deployment share using the Copy-Item cmdlet, as shown in the following example:

   ```
   PowerShell

   Copy-Item "DS002:\Captures\Win7SP1.wim" -Destination "DS002:\Operating
   Systems\Windows 7\sources\Install.wim" Force
   ```

   In this example, the captured operating system image file (Win7SP1.wim) in the Captures folder in the share DS002: is the name of a Windows PowerShell drive returned in step 6, and the existing Windows 7 operating system is stored in folder named *Windows 7*.

8. Update any MDT deployment media based on deployment share using **Update-MDTMedia** cmdlet.

   For more information about how to update MDT deployment media based on deployment share using **Update-MDTMedia** cmdlet, see Updating Deployment Media Using Windows PowerShell.

Customizing Deployment Based on Chassis Type

You can customize the deployment based on the chassis type of the computer. The scripts create local variables that can be processed in the CustomSettings.ini file. The local variables `IsLaptop`, `IsDesktop`, and `IsServer` indicate whether the computer is a portable computer, desktop computer, or server, respectively.

> **Note**

In earlier versions of the Deployment Workbench, the `IsServer` flag indicated that the existing operating system is a server operating system (such as Windows Server 2003 Enterprise Edition). This flag has been renamed to `IsServerOS`.

To implement local variables in the CustomSettings.ini file

1. In the `[Settings]` section, on the `Priority` line, add a custom section to customize deployment based on the chassis type (`ByChassisType` in the following example, where *Chassis* represents the type of computer).
2. Create the custom section that corresponds to the custom section defined in step 1 (`ByChassisType` in the example in following example, where *Chassis* represents the type of computer).
3. Define a subsection for each chassis type to detect (`Subsection=Laptop-%IsLaptop%, Subsection=Desktop-%IsDesktop%, Subsection=Server-%IsServer%` in the following example).
4. Create a subsection for each `True` and `False` state of each subsection defined in step 3 (such as `[Laptop-True], [Laptop-False], [Desktop-True], [Desktop-False]` in the following example).
5. Under each `True` and `False` subsection, add the appropriate settings based on the chassis type.

   Listing 1. Example of Customizing Deployment Based on Chassis Type in the CustomSettings.ini File

```ini
[Settings]

Priority=...,ByLaptopType,ByDesktopType,ByServerType

[ByLaptopType]
Subsection=Laptop-%IsLaptop%

[ByDesktopType]
Subsection=Desktop-%IsDesktop%

[ByServerType]
Subsection=Server-%IsServer%
.
.
.

[Laptop-True]
```

```
.
.
.

[Laptop-False]
.
.
.

[Desktop-True]
.
.
.

[Desktop-False]
.
.
.

[Server-True]
.
.
.

[Server-False]
.
.
.
```

Deploying Applications Based on Earlier Application Versions

Often, when installing an operating system on an existing computer, you will install the same applications you previously installed on the computer. Do this using MDT scripts (in particular, ZTIGather.wsf) to query two separate sources of information:

- **Configuration Manager software inventory feature.** Contains one record for each application package—in this case, listings in Program and Features in Windows 8.1, Windows 8, Windows 7, Windows Server 2012 R2, Windows Server 2012, Windows Server 2008 R2—installed the last time Configuration Manager inventoried the computer.

- **A mapping table.** Describes which package and program need to be installed for each record (because the Program and Features or Add or Remove Programs records do not specify exactly which package installed the application, making it impossible to automatically select the package based on inventory alone).

To perform a dynamic computer-specific application installation

1. Use the table in the MDT DB to connect specific packages with applications listed in the target operating system.

2. Populate the table with data that associates the appropriate package with the application listed in Program and Features or Add or Remove Programs.

   SQL Query to Populate the Table

   ```
   SQL

   use [MDTDB]
   go
   INSERT INTO [PackageMapping] (ARPName, Packages) VALUES('Office12.0',
   'XXX0000F:Install Office 2010 Professional Plus')
   go
   ```

   The inserted row connects any computer that has the entry `Office12.0` with the Microsoft Office 2010 Professional Plus package.

   This means that Microsoft Office 2010 Professional Plus will be installed on any computer currently running the 2007 Microsoft Office system (Office 12.0). Add similar entries for any other packages. Any item for which there is no entry is ignored (no package will be installed).

3. Create a stored procedure to simplify joining the information in the new table with the inventory data.

   ```
   SQL

   use [MDTDB]
   go

   if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].
   [RetrievePackages]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)
   drop procedure [dbo].[RetrievePackages]
   go

   CREATE PROCEDURE [dbo].[RetrievePackages]
   @MacAddress CHAR(17)
   AS

   SET NOCOUNT ON

   /* Select and return all the appropriate records based on current
   inventory */
   SELECT * FROM PackageMapping
   WHERE ARPName IN
   ```

```
(
  SELECT ProdID0 FROM CM_DB.dbo.v_GS_ADD_REMOVE_PROGRAMS a,
CM_DB.dbo.v_GS_NETWORK_ADAPTER n
  WHERE a.ResourceID = n.ResourceID AND
  MACAddress0 = @MacAddress
)
go
```

The stored procedure in the preceding example assumes that the Configuration Manager central primary site database resides on the computer on which SQL Server is running as the MDT DB. If the central primary site database resides on a different computer, the appropriate modifications need to be made to the stored procedure. In addition, the name of the database (`CM_DB`) must be updated. Also consider granting additional accounts Read access to the **v_GS_ADD_REMOVE_PROGRAMS** view in the Configuration Manager database.

4. Configure the CustomSettings.ini file to query this database table by specifying the name of a section (`[DynamicPackages]` in the **Priority** list) that points to the database information.

```ini
[Settings]
...
Priority=MacAddress, DefaultGateway, DynamicPackages, Default
...
```

5. Create a `[DynamicPackages]` section to specify the name of a database section.

```ini
[DynamicPackages]
SQLDefault=DB_DynamicPackages
```

6. Create a database section to specify the database information and query details.

```ini
[DB_DynamicPackages]
SQLServer=SERVER1
Database=MDTDB
StoredProcedure=RetrievePackages
Parameters=MacAddress
SQLShare=Logs
Instance=SQLEnterprise2005
```

```
Port=1433
Netlib=DBNMPNTW
```

In the preceding example, the MDT DB named *MDTDB* on the computer running the SQL Server instanced named *SERVER1* will be queried. The database contains a stored procedure named `RetrievePackages` (created in step 3).

When ZTIGather.wsf runs, a Structured Query Language (SQL) `SELECT` statement is automatically generated, and the value of the **MakeModelQuery** custom key is passed as a parameter to the query:

```
SQL

EXECUTE RetrievePackages ?
```

The actual value of the **MACAddress** custom key will be substituted for the corresponding "?". This query returns a record set with the rows entered in step 2.

A variable number of arguments cannot be passed to a stored procedure. As a result, when a computer has more than one MAC address, not all MAC addresses can be passed to the stored procedure. As an alternative, replace the stored procedure with a view that allows querying the view with a `SELECT` statement with an `IN` clause to pass all the MAC address values.

Based on the scenario presented here, if the current computer has the value `Office12.0` inserted into the table (step 2), the one row is returned (`XXX0000F:Install Office 2010 Professional Plus`). This indicates that package XXX0000F:Install Office 2001 Professional Plus will be installed by the ZTI process during the State Restore Phase.

Fully Automated LTI Deployment Scenario

The main purpose of LTI is to automate the deployment process as much as possible. Although ZTI provides full deployment automation using the MDT scripts and Windows Deployment Services, LTI is designed to work with fewer infrastructure requirements.

You can automate the Windows Deployment Wizard used in the LTI deployment process to reduce (or eliminate) the wizard pages displayed. You can skip the entire Windows Deployment Wizard by specifying the **SkipWizard** property in CustomSettings.ini. To skip individual wizard pages, use the following properties:

- **SkipAdminPassword**

- **SkipApplications**
- **SkipBDDWelcome**
- **SkipBitLocker**
- **SkipBitLockerDetails**
- **SkipTaskSequence**
- **SkipCapture**
- **SkipComputerBackup**
- **SkipComputerName**
- **SkipDomainMembership**
- **SkipFinalSummary**
- **SkipLocaleSelection**
- **SkipPackageDisplay**
- **SkipProductKey**
- **SkipSummary**
- **SkipTimeZone**
- **SkipUserData**

For more information about these individual properties, see the corresponding property in the MDT document *Toolkit Reference*.

For each wizard page skipped, provide the values for the corresponding properties that are typically collected through the wizard page in the CustomSettings.ini and BootStrap.ini files. For more information about the properties that must be configured in these files, see the section, "Providing Properties for Skipped Deployment Wizard Pages", in the MDT document *Toolkit Reference*.

Fully Automated LTI Deployment for a Refresh Computer Scenario

The following illustrates a CustomSettings.ini file used for a Refresh Computer scenario to skip all Windows Deployment Wizard pages. In this sample, the properties to provide

when skipping the wizard page are immediately beneath the property that skips the wizard page.

ini

```ini
[Settings]
Priority=Default
Properties=MyCustomProperty

[Default]
OSInstall=Y
ScanStateArgs=/v:5 /o /c
LoadStateArgs=/v:5 /c /lac /lae
SkipCapture=YES
SkipAdminPassword=YES
SkipProductKey=YES

DeploymentType=REFRESH

SkipDomainMembership=YES
JoinDomain=DomainName
DomainAdmin=Administrator
DomainAdminDomain=DomainName
DomainAdminPassword=a_secure_password

SkipUserData=yes
UserDataLocation=AUTO
UDShare=\\Servername\Sharename\Directory
UDDir=%ComputerName%

SkipComputerBackup=YES
ComputerBackuplocation=AUTO
BackupShare=\\Servername\Backupsharename
BackupDir=%ComputerName%

SkipTaskSequence=YES
TaskSequenceID=Enterprise

SkipComputerName=YES
OSDComputerName=%ComputerName%

SkipPackageDisplay=YES
LanguagePacks001={3af4e3ce-8122-41a2-9cf9-892145521660}
LanguagePacks002={84fc70d4-db4b-40dc-a660-d546a50bf226}

SkipLocaleSelection=YES
UILanguage=en-US
UserLocale=en-CA
KeyboardLocale=0409:00000409

SkipTimeZone=YES
TimeZoneName=China Standard Time

SkipApplications=YES
```

```
Applications001={a26c6358-8db9-4615-90ff-d4511dc2feff}
Applications002={7e9d10a0-42ef-4a0a-9ee2-90eb2f4e4b98}
UserID=Administrator
UserDomain=DomainName
UserPassword=P@ssw0rd

SkipBitLocker=YES
SkipSummary=YES
Powerusers001=DomainName\Username
```

Fully Automated LTI Deployment for a New Computer Scenario

The following is an example of a CustomSettings.ini file used for a New Computer scenario to skip all Windows Deployment Wizard pages. In this sample, the properties to provide when skipping the wizard page are immediately beneath the property that skips the wizard page.

ini

```
[Settings]
Priority=Default
Properties=MyCustomProperty

[Default]
OSInstall=Y
ScanStateArgs=/v:5 /o /c
LoadStateArgs=/v:5 /c /lac /lae

SkipCapture=YES
ComputerBackupLocation=\\WDG-MDT-01\Backup$\
BackupFile=MyCustomImage.wim

SkipAdminPassword=YES
SkipProductKey=YES

SkipDomainMembership=YES
JoinDomain=WOODGROVEBANK
DomainAdmin=Administrator
DomainAdminDomain=WOODGROVEBANK
DomainAdminPassword=P@ssw0rd

SkipUserData=Yes
UserDataLocation=\\WDG-MDT-01\UserData$\Directory\usmtdata

SkipTaskSequence=YES
TaskSequenceID=Enterprise

SkipComputerName=YES
OSDComputerName=%SerialNumber%
```

```
SkipPackageDisplay=YES
LanguagePacks001={3af4e3ce-8122-41a2-9cf9-892145521660}
LanguagePacks002={84fc70d4-db4b-40dc-a660-d546a50bf226}

SkipLocaleSelection=YES
UILanguage=en-US
UserLocale=en-CA
KeyboardLocale=0409:00000409

SkipTimeZone=YES
TimeZoneName=China Standard Time

SkipApplications=YES
Applications001={a26c6358-8db9-4615-90ff-d4511dc2feff}
Applications002={7e9d10a0-42ef-4a0a-9ee2-90eb2f4e4b98}

SkipBitLocker=YES
SkipSummary=YES
Powerusers001=WOODGROVEBANK\PilarA
CaptureGroups=YES
SLShare=\\WDG-MDT-01\UserData$\Logs
Home_page=https://www.microsoft.com/NewComputer
```

Calling Web Services in MDT

In earlier versions of MDT, rules processing was supported through CustomSettings.ini and databases, from which you could retrieve values from the local computer—typically using WMI—to make decisions on what needed to be done on each computer during deployment. In addition, you could make SQL queries and stored procedure calls to retrieve additional information from external databases. There were challenges with that approach, though—especially with making secure SQL Server connections.

To help with this problem, MDT has the ability to make web service calls based on simple rules defined in CustomSettings.ini. These web service requests do not require any special security context and can use whatever TCP/IP port is needed to simplify firewall configurations.

The following shows how to configure CustomSettings.ini to call a particular web service. In this scenario, the web service is chosen at random from an Internet search. It takes a postal code as input and returns the city, state, area code, and time zone (as a letter) for the specified postal code.

ini

```
[Settings]
Priority=Default, USZipService
```

```
Properties=USZip, City, State, Zip, Area_Code, Time_Zones
[Default]
USZip=98052
[USZipService]
WebService=http://www.webservicex.net/uszip.asmx/GetInfoByZIP
Parameters=USZip
```

Executing this code produces output similar to the following:

Output

```
Added new custom property USZIP
Added new custom property CITY
Added new custom property STATE
Added new custom property ZIP
Added new custom property AREA_CODE
Added new custom property TIME_ZONES
Using from [Settings]: Rule Priority = DEFAULT, USZIPSERVICE
------ Processing the [DEFAULT] section ------
Property USZIP is now = 98052
Using from [DEFAULT]: USZIP = 98052
------ Processing the [USZIPSERVICE] section ------
Using COMMAND LINE ARG: Ini file = CustomSettings.ini
CHECKING the [USZIPSERVICE] section
About to execute web service call to
http://www.webservicex.net/uszip.asmx/GetInfoByZIP: USZip=98052
Response from web service: 200 OK
Successfully executed the web service.
Property CITY is now = Redmond
Obtained CITY value from web service:  CITY = Redmond
Property STATE is now = WA
Obtained STATE value from web service:  STATE = WA
Property ZIP is now = 98052
Obtained ZIP value from web service:  ZIP = 98052
Property AREA_CODE is now = 425
Obtained AREA_CODE value from web service:  AREA_CODE = 425
------ Done processing CustomSettings.ini ------
```

There are a few minor complications to watch for when running a web service:

- Do not do anything special with proxy servers. If there is an anonymous proxy present, use it, but authenticating proxies could cause problems. In most cases, a web service will not be called.

- CustomSettings.ini or ZTIGather.xml searches for properties defined in the XML markup returned as a result of the web service call (just as with a database query or other rule). However, the XML search is case sensitive. Fortunately, the web service described here returns all uppercase property names, which is what

ZTIGather.xml expects. It is possible to remap lowercase or mixed-case entries to get around this.

- A `POST` request to the web service is recommended, so the web service call must be able to support a `POST`.

Connecting to Network Resources

During LTI and ZTI deployment processes, you might require access to a network resource on a server different from the server hosting the deployment share. You must be authenticated on the other server so that you can access shared folders or services there. For example, you can install an application from a shared folder on a server other than the server hosting the deployment share that the MDT scripts use.

> ⓘ **Note**
>
> To query SQL Server databases hosted on a server other than the server hosting the deployment share, see the **Database**, **DBID**, **DBPwd**, **Instance**, **NetLib**, **Order**, **Parameters**, **ParameterCondition**, **SQLServer**, **SQLShare**, and **Table** properties in the MDT document *Toolkit Reference*.

Using the ZTIConnect.wsf script, you can connect to other servers and access resources on them. The syntax for the ZTIConnect.wsf script is as follows (where *unc_path* is a Universal Naming Convention [UNC] path to connect to the server):

Windows Command Prompt

```
cscript.exe "%SCRIPTROOT%\ZTIConnect.wsf" /uncpath:unc_path
```

In most instances, you run the ZTIConnect.wsf script as a Task Sequencer task. Run the ZTIConnect.wsf script prior to tasks requiring access to a server other than the server hosting the deployment share.

To add the ZTIConnect.wsf script as a task to the task sequence of a build

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/Task Sequences (where *deployment_share* is the name of the deployment share to configure).

3. In the details pane, select ***task_sequence*** (where *task_sequence* is the task sequence to modify).

4. In the Actions pane, select **Properties**.

5. Select the Task Sequence tab, browse to **group** (where *group* is the group in which to run the ZTIConnec.wsf script), and select **Add**. Select **General**, and then select **Run Command Line**.

   > **Note**
   >
   > Add the task before adding any tasks that require access to resources on the target server.

6. Complete the **Properties** tab of the new task using the following information:

   Expand table

In this box	Do this
Name	Type **Connect to server** (where server is the name of the server to which to connect).
Description	Type text that explains why the connection needs to be made.
Command	Type **cscript.exe "%SCRIPTROOT%\ZTIConnect.wsf" /uncpath:unc_path** (where *unc_path* is the UNC path to a shared folder on the server).

7. Complete the **Options** tab of the new task using the following information. Unless specified, accept default values, and then select **OK**.

   Expand table

In this box	Do this
Success codes	Type **0 3010**. (The ZTIConnect.wsf script returns these codes upon successful completion.)
Conditions list box	Add any conditions that might be necessary. (In most instances this task requires no conditions.)

   After adding the task that will run the ZTIConnect.wsf script, subsequent tasks can access network resources on the server specified in the **/uncpath** option of the ZTIConnect.wsf script.

Deploying the Correct Device Drivers to Computers with the Same Hardware Devices but Different Make and Model

Variations on model numbers and names can exist with virtually no difference in the driver set. These variations in model numbers and names can unnecessarily increase time spent making multiple database entries for a given model. The following procedure shows how to define a new property using a user exit function call that returns a substring of the model number.

To create model aliases

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share* (where *deployment_share* is the name of the deployment share to configure).
3. In the Actions pane, select **Properties**.
4. In the **Properties** dialog box, select the **Rules** tab.
5. Create aliases for hardware types in the Make and Model sections of the MDT DB. Truncate the model type at the open parentheses "(" in the model name. For example, *HP DL360 (G112)* becomes *HP DL360*.
6. Add the custom variable **ModelAlias** to each section.
7. Create a new `[SetModel]` section.
8. Add the `[SetModel]` section to the **Priority** settings in the `[Settings]` section.
9. Add a line to the `ModelAlias` section to refer to a user exit script that will truncate the model name at the "(".
10. Create an **MMApplications** database lookup where **ModelAlias** is equal to **Model**.
11. Create a user exit script and place it in the same directory as the CustomSettings.ini file to truncate the model name.

    The following shows a CustomSettings.ini and the user exit script, respectively.

    CustomSettings.ini:

    ```
    ```

ini

```ini
[Settings]
Priority=SetModel, MMApplications, Default
Properties= ModelAlias
[SetModel]
ModelAlias=#SetModelAlias()#
Userexit=Userexit.vbs
[MMApplications]
SQLServer=Server1
Database=MDTDB
Netlib=DBNMPNTW
SQLShare=logs
Table= MakeModelSettings
Parameters=Make, ModelAlias
ModelAlias=Model
Order=Sequence
```

User Exit Script:

vbs

```vbs
Function UserExit(sType, sWhen, sDetail, bSkip)
  UserExit = Success
End Function

Function SetModelAlias()
  If Instr(oEnvironment.Item("Model"), "(") <> 0 Then
    SetModelAlias = Left(oEnvironment.Item("Model"), _
                      Instr(oEnvironment.Item("Model"), _
                        "(") - 1)
    oLogging.CreateEntry "USEREXIT - " & _
      "ModelAlias has been set to " & SetModelAlias, _
      LogTypeInfo
  Else
    SetModelAlias = oEnvironment.Item("Model")
    oLogging.CreateEntry " USEREXIT - " & _
      "ModelAlias has not been changed.", LogTypeInfo
  End if
End Function
```

Configuring Conditional Task Sequence Steps

In some scenarios, consider running a task sequence step conditionally based on defined criteria. Any combinations of these conditions can be added to determine whether the task sequence step should run. For example, use the value of a task sequence variable and the value of a registry setting to determine whether a task sequence step should run.

Using MDT, run a task sequence conditionally based on:

- One or more **IF** statements
- A task sequence variable
- The version of the target operating system
- The Boolean results of a WMI query
- A registry setting
- The software installed on the target computer
- The properties of a folder
- The properties of a file

Configuring a Conditional Task Sequence Step

Conditional task sequence steps are configured in the Deployment Workbench, on the **Options** tab of a task sequence step. You can add one or more conditions to the task sequence step to create the appropriate condition for running or not running the step.

> ⓘ **Note**
>
> Every conditional task sequence step needs at least one **IF** statement.

To view the Options tab of a task sequence step

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/Task Sequences (where *deployment_share* is the name of the deployment share to configure).
3. In the details pane, select **task_sequence** (where *task_sequence* is the name of the task sequence to configure).
4. In the Actions pane, select **Properties**.
5. In the ***task_sequence*** **Properties** dialog box, on the **Task Sequence** tab, select ***step*** (where *step* is the name of the task sequence step to configure), and then select the **Options** tab.

On the **Options** tab of the task sequence step, perform the following actions:

- **Add.** Select this button to add a condition to the task sequence step.
- **Remove.** Select this button to remove an existing condition in a task sequence step.
- **Edit.** Select this button to modify an existing condition in a task sequence step.

IF Statements in Conditions

All task sequence conditions include one or more **IF** statements. **IF** statements are the foundation for creating conditional task sequence steps. A task sequence step condition can include only one **IF** statement, but multiple **IF** statements can be nested beneath the top-level **IF** statement to create more complex conditions.

An **IF** statement can be based on the conditions listed in the following table, which are configured in the **IF Statement Properties** dialog box.

Expand table

Condition	Select this option to run the task sequence if
All conditions	All the conditions beneath this **IF** statement must be true.
Any conditions	Any the conditions beneath this **IF** statement are true.
None	None the conditions beneath this **IF** statement are true.

Complete the condition for running the task sequence step by adding other criteria to the conditions (for example, task sequence variables or values in a registry setting).

To add an IF statement condition to a task sequence step

1. On the ***step*** **Option** tab (where *step* is the name of the task sequence step to configure), select **Add**, and then select **If statement**.
2. In the **If Statement Properties** dialog box, select **condition** (where *condition* is one of the conditions listed in the previous table), and then select **OK**.

Task Sequence Variables in Conditions

Use the **Task Sequence Variable** condition to evaluate any task sequence variable created by a **Set Task Sequence Variable** task or by any task in the task sequence. For example, consider a network that contains Windows XP client computers that are part of

a domain and some that are in a workgroup. Knowing that the current domain policy forces all user settings to be saved on the network, user settings may need to be saved only for computers that are not part of the domain—that is, computers that are in the workgroup. In such case, add a condition to the **Capture User Files and Settings** task that targets the computers in the workgroup.

To add a condition based on a task sequence variable

1. On the ***step*** **Options** tab (where *step* is the name of the task sequence step to configure), select **Add Condition**, and then select **Task Sequence Variable**.
2. In the **Task Sequence Variable** Condition dialog box, in the **Variable** box, type **OSDJoinType**.

   > ⓘ **Note**
   >
   > This variable is set to **0** for computers that are joined to a domain and to **1** for those in a workgroup.

3. In the **Condition** box, select **equal**.
4. In the **Value** box, type **1**, and then select **OK**.

Operating System Version in Conditions

Use the **Operating System Version** condition to verify the existing operating system version of a target computer or the existing client (when capturing an image). For instance, consider a network that contains several servers that will be upgraded from Windows Server 2003 to Windows Server 2008. Network settings should be copied and applied only to servers that are running Windows Server 2003. All other servers will have the default network settings that Windows Server 2008 uses.

To add a condition based on operating system version

1. In the Task Sequence Editor, select the **Capture Network Settings** task.
2. Select **Add Condition**, and then select **Operating System Version**.
3. In the **Architecture** box, select the relevant server. For this example, select **x86**.
4. In the **Operating system** box, select the operating system and version for which to set a condition. For this example, select **x86 Windows 2003**.
5. In the **Condition** box, select the relevant condition, and then select **OK**.

File Properties in Conditions

Use the **File Properties** condition to verify the version and/or times tamp of a given file to determine whether or not to run a task or a group of tasks. In this example, the production environment contains a Windows Server 2003 image that is constantly updated and used for every new server that is added to the network. All server computers in the environment run a custom application that requires the Digital Access Object (DAO) application programming interface (API) version 3.60.6815.

All existing servers are working properly. However, each new server added to the network with the image is unable to run the application. Because it is the responsibility of a different group to maintain and update images, you decide that the deployment task sequence be changed to install the relevant version of DAO if the existing version of DAO deployed with the image is incorrect.

To add a File Properties condition to a task sequence step in Configuration Manager

1. In Configuration Manager, create a package to install DAO 3.60.6815. Call this package *DAO*, with a program called *InstallDAO*. To learn more about creating packages, see How to create a package.
2. Create an **Install Software** step to deploy the DAO package.
3. Select the **Install Software** task sequence step created in step 2, and then select the **Options** tab.
4. Select **Add Condition**, and then select **File Properties**.
5. In the **Path** box, type **C:\Program Files\Microsoft Shared\DAO\dao360.dll**.
6. Select the **Check the version** check box, and then select **not equals** for the condition.
7. In the **Version** box, type **3.60.6815**.
8. In this case, clear the **Check the timestamp** check box, and then select **OK**.

Folder Properties in Conditions

Use the **Folder Properties** condition to verify the time stamp of a given folder to determine whether to run a task or a group of tasks. For instance, consider a situation in which an internally developed application has been updated to work with Windows 8. However, not all of the computers in the network have the most recent version of the

application installed, and you must perform a data-conversion process before you can upgrade the application.

If the time stamp of the folder in which the application is installed is 12/31/2007 or earlier, then the target computer is running the incompatible version of the application, and you should run the data-conversion process on the target computer. Conditionally, run a task sequence step to run the data-conversion process on computers that have an earlier version of the application.

To add a Folder Properties condition to a task sequence step

1. In the Configuration Manager console or in the Deployment Workbench, in the task sequence editor, edit ***task_sequence*** (where *task sequence* is the task sequence you want to edit).
2. Create a **Command Line** task to perform the data-conversion process.
3. Select the task created in step 1.
4. Select **Add Condition**, and then select **Folder Properties**.
5. In the **Path** box, type the path of the folder that contains the application.
6. Select the **Check the timestamp** check box.
7. Select **Less than or equals** for the condition.
8. In the **Date** box, select **12/31/2007**.
9. In the **Time** box, select **12:00:00 AM**, and then select **OK**.

Registry Settings in Conditions

Use the **Registry Setting** condition to verify the existence of keys and values in the registry and the corresponding data stored in registry values. For instance, consider a case in which an application currently used on a small set of computers cannot run on Windows 8, and a Windows 8 deployment is in place to upgrade computers that currently are running Windows XP. Create a condition on the very first task in a sequence to check the registry for an entry for the incompatible application and to interrupt the deployment process for that computer if it is found.

To add a Registry Setting condition to a task sequence step

1. In the Configuration Manager console or in the Deployment Workbench, in the task sequence editor, edit ***task_sequence*** (where *task sequence* is the task sequence

that deploys Windows 8).

2. Select the first task in the sequence, and then select the **Options** tab.
3. Select **Add Condition**, and then select **Registry Setting**.
4. In the **Root key** list, select **HKEY_LOCAL_MACHINE**.
5. In the **Key** box, type **SOFTWARE\WOODGROVE**.
6. Select **not exists** for the condition. In this case, the task will run, and the sequence continue only if the key does not exist.
7. Optionally, the condition could check for the nonexistence of a value if the value name is typed in the **Value name** box.
8. If a condition other than **exists/not exists** was used, specify a value and value type.
9. Select **OK**.

WMI Queries in Conditions

Use the **WMI Query** condition to run any WMI query. The condition is evaluated as True if the query returns at least one result. For example, consider that a deployment team needs to upgrade the operating system of all servers of a given model—Dell 1950, for instance. You can use a WMI query to check each computer's model and proceed with the deployment only if the right model is found.

To add a WMI Query condition to a task sequence step

1. In the Configuration Manager console or in the Deployment Workbench, in the task sequence editor, edit ***task_sequence*** (where *task sequence* is the task sequence that will upgrade the servers).
2. Select the first task in the sequence, and then select the **Options** tab.
3. Select **Add Condition**, and then select **Query WMI**.
4. In the **WMI Namespace** box, type **root\cimv2**.
5. In the **WQL Query** box, type **Select * From Win32_ComputerSystem WHERE Model LIKE "%Dell%%1950%"**. Select **OK**.

Installed Software in Conditions

Use an **Installed Software** condition to check if a particular piece of software is currently installed on a target computer. Only software installed using Microsoft Installer (MSI) files can be evaluated using this condition. As an example, imagine that you want to upgrade the operating system of all servers except those running Microsoft SQL Server 2012.

To add an Installed Software condition to a task sequence step

1. In the Configuration Manager console or in the Deployment Workbench, in the task sequence editor, edit ***task_sequence*** (where *task sequence* is the task sequence that will upgrade the servers).
2. Select the first task in the sequence, and then select the **Options** tab.
3. Select **Add Condition**, and then select **Installed Software**.
4. Select **Browse**, and then select the MSI file for SQL Server 2012.
5. Select the **Match this specific product** check box to specify that only computers with SQL Server 2012 and not any other versions are the target computers this query should detect.
6. Select **OK**.

Complex Conditions

Multiple conditions can be grouped using **IF** statements to create complex conditions. For instance, imagine that a particular step should only be run for Contoso 1950 computers running Windows Server 2003 or Windows Server 2008. Written as a programmatic **IF** statement, it would look similar to the following:

```
IF ((Computer Model IS "Contoso 1950") AND (operating system=2003 OR operating system=2008))
```

To add a complex condition

1. In the Configuration Manager console or in the Deployment Workbench, in the task sequence editor, edit ***task_sequence*** (where *task sequence* is the task sequence that will upgrade the servers).
2. Select the task sequence step to which to add the condition, and then select the **Options** tab.
3. Select **Add condition**, select **If Statement**, and then select **All conditions**. Select **OK**.

4. Select the condition statement, select **Add condition**, and then select **WMI Query**.
5. Ensure **root\cimv2** is specified as the WMI namespace, and then, in the **WQL Query** box, type **SELECT * FROM Win32_ComputerSystem WHERE ComputerModel LIKE "%Contoso%1950%"**. Select **OK**.
6. Select the **IF** statement, and then select **Add condition**. Select **If statement**, and then select **Any condition**. Select **OK**.
7. Select the second **IF** statement. Select **Add condition**, and then select **Operating System Version**.
8. In the **Architecture** box, select the architecture for the servers. For this example, select **x86**.
9. In the **Operating system** box, select the operating system and version. For this example, select **x86 Windows 2003 original release**. Select **OK**.
10. Select the second **IF** statement. Select **Add condition**, and then select **Operating System Version**.
11. In the **Architecture** box, select the architecture for the servers. For this example, select **x86**.
12. In the **Operating system** box, select the operating system and version. For this example, select **x86 Windows 2008 original release**. Select **OK**.

Creating a Highly Scalable LTI Deployment Infrastructure

In this scenario, no electronic software distribution is available for the deployment infrastructure to leverage, so you use MDT to build a fully automated LTI deployment infrastructure. The scalable LTI infrastructure uses SQL Server, Windows Deployment Services, and Windows Server 2003 Distributed File System Replication (DFS-R) technologies.

Scale the LTI infrastructure by:

- Ensuring that the appropriate infrastructure exists as described in Ensuring That the Appropriate Infrastructure Exists
- Adding content to MDT as described in Adding Content to MDT

- Preparing Windows Deployment Services as described in Preparing Windows Deployment Services
- Configuring DFS-R as described in Configuring Distributed File System Replication
- Preparing for SQL Server replication as described in Preparing for SQL Server Replication
- Configuring SQL Server replication as described in Configuring SQL Server Replication

  This scenario presumes that MDT is configured on a master deployment server and that the configuration of the MDT DB has already been completed as discussed at the beginning of this document.

Ensuring That the Appropriate Infrastructure Exists

The highly scalable LTI deployment infrastructure uses a hub-and-spoke topology for replication of content; therefore, first nominate a deployment server in the production environment that will perform the role of the master deployment server. The following lists the required components for the master deployment server.

Expand table

Required component	Purpose/comment
Windows Server 2003 R2	Required to support DFS-R
MDT	Contains the master copy of the deployment share
SQL Server 2005	Must be a full version to allow replication of the MDT DB
DFS-R	Required for replication of the deployment share
Windows Deployment Services	Required to allow network PXE-based installations to be initiated

When you have selected the master deployment server, provision additional servers at each site to support LTI deployments. The following lists the required components for the child deployment server.

Expand table

Required component	Purpose/comment
Windows Server 2003 R2	Required to support DFS-R

Required component	Purpose/comment
Microsoft SQL Server 2005 Express Edition	Receives replicated copies of the MDT DB
DFS-R	Required for replication of deployment share
Windows Deployment Services	Required to allow network PXE-based installations to be initiated

> **Note**
>
> Windows Deployment Services must be set up and configured on each child server, but it is not necessary to add boot or installation images.

Adding Content to MDT

Populate the master deployment server with content using the Deployment Workbench, and create and populate the MDT DB as described in the following sections. For information on populating the database with:

- Applications, see the section, "Configuring Applications in the Deployment Workbench", in the MDT document *Using the Microsoft Deployment Toolkit*
- Operating systems, see the section, "Configuring Operating Systems in the Deployment Workbench", in the MDT document *Using the Microsoft Deployment Toolkit*
- Operating system packages, see the section, "Configuring Packages in the Deployment Workbench", in the MDT document *Using the Microsoft Deployment Toolkit*
- Device drivers, see the section, "Configuring Device Drivers in the Deployment Workbench", in the MDT document *Using the Microsoft Deployment Toolkit*
- Task sequences, see the section, "Configuring Task Sequences in the Deployment Workbench", in the MDT document *Using the Microsoft Deployment Toolkit*

> **Note**
>
> Ensure that the LiteTouchPE_x86.wim file created when the deployment share is updated has been added to Windows Deployment Services.

Preparing Windows Deployment Services

Because the LiteTouchPE_x86.wim file will be replicated on a periodic basis through the DFS-R replication group, the boot configuration data store must be updated periodically to reflect the newly replicated Windows PE environment. Perform the following steps on each of the deployment servers.

To prepare Windows Deployment Services

1. Open a Command Prompt window.
2. Type **WDSUtil/set-server/BCDRefreshPolicy/Enabled:yes/RefreshPeriod:60**, and then press ENTER.

> ⓘ **Note**
>
> In the example presented here, the refresh period is set to 60 minutes; however, you could configure this value to replicate during a period equal to that of the DFS-R.

Configuring Distributed File System Replication

When scaling the LTI deployment architecture, you use DFS-R as the basis for replicating the content from both the MDT deployment share and the Windows PE Lite Touch boot environment and from the master deployment server to the child deployment servers.

> ⓘ **Note**
>
> Ensure that DFS-R is installed before performing the following steps.

To configure DFS-R to replicate the deployment content

1. Open DFS Management console.
2. In the DFS Management console, expand DFS Management.
3. Right-click **Replication**, and then select **New Replication Group**.
4. In the New Replication Group Wizard, on the **Replication Group Type** page, select **New Multipurpose Replication Group**.
5. Select **Next**.

6. On the **Name and Domain** page, type the following information:

    - In the **Name for replication group** box, type a name for the replication group —for example, **MDT 2010 Replication Group**.
    - In the **Optional description of replication group** box, type a description of the replication group—for example, **Group for replication of MDT 2010 data**.
    - Ensure that the **Domain** box contains the correct domain name.

7. Select **Next**.
8. On the **Replication Group Members** page, perform these steps:

    a. Select **Add**.

    b. Type the names of all servers that are to be members of this replication group—for example, all child deployment servers and the master deployment server.

    c. Select **OK**.

9. Select **Next**.
10. On the **Topology Selection** page, select **Hub and Spoke**, and then select **Next**.
11. On the **Hub Members** page, select the master deployment server, and then select **Add**.
12. Select **Next**.
13. On the **Hub and Spoke Connections** page, ensure that for each child deployment server the master deployment server listed is the **Required Hub Member**.
14. Select **Next**.
15. On the **Replication Group Schedule and Bandwidth** page, specify a schedule for replicating the content between servers.
16. Select **Next**.
17. On the **Primary Member** page, in the **Primary Member** box, select the master deployment server.
18. Select **Next**.
19. On the **Folders to Replicate** page, select **Add**, and then perform these steps:

a. In the **Local Path of the folder to replicate** box, select **Browse** to go to the *X*:\Deployment folder (where *X* is the drive letter on the deployment server).

b. Select **Use name based on path**.

c. Select **OK**.

d. Select **Add**.

e. In the **Add Folder to Replicate** dialog box, select **Browse** to go to the *X*:\RemoteInstall\Boot folder.

f. Select **Use name based on path**.

20. Select **Next**.

21. On the **Local Path of Distribution on Other Members** page, perform these steps:

    a. Select all the members in the distribution group, and then select **Edit**.

    b. In the **Edit Local Path** dialog box, select **Enabled**.

    c. Type the path where the Deployment Share folder should be stored on the child deployment server—for example, ***X:\Deployment*** (where *X* is the drive letter on the deployment server).

    d. Select **OK**.

22. Select **Next**.

23. On the **Local Path of Boot on Other Members** page, perform these steps:

    a. Select all the members in the distribution group, and then select **Edit**.

    b. In the **Edit Local Path** dialog box, select **Enabled**.

    c. Type the path where the Boot folder should be stored on the child deployment server—for example, ***X:\RemoteInstall\Boot*** (where *X* is the drive letter on the deployment server).

    d. Select **OK**.

24. Select **Next**.

25. On the **Remote Settings and Create Replication Group** page, select **Create** to complete the New Replication Group Wizard.

26. On the **Confirmation** page, select **Close** to close the wizard.

> **Note**
>
> Ensure that the new replication group is now listed beneath the Replication node.

Preparing for SQL Server Replication

Before SQL Server replication can be configured, complete several pre-configuration steps to ensure that the deployment servers are correctly configured.

To prepare for SQL Server replication on the master deployment server

1. Create a folder to store the database snapshots, and then configure the folder as a share.

   > **Note**
   >
   > For more information about securing the snapshot folder, see Secure the Snapshot Folder.

2. Ensure that the SQL Server Browser service is enabled and set to Automatic.
3. In the **SQL Server Surface Area Configuration** box, select **Local and Remote connections**.

   To prepare for SQL Server replication on the child deployment server

4. In the **SQL Server Surface Area Configuration** box, select **Local and Remote connections**.
5. Optionally, create an empty database to host the replicated MDT DB.

> **Note**
>
> This database must be given the same name as the MDT DB on the master deployment server. For example, if the MDT DB on the master deployment server is called *MDTDB*, create an empty database called *MDTDB* on the child deployment server.

Configuring SQL Server Replication

After configuring the replication of files and folders required to build the deployment infrastructure, configure SQL Server to replicate the MDT DB.

> ⓘ **Note**
>
> It is also possible to maintain only a single central MDT DB; however, by maintaining a replicated version of the MDT DB, greater control can be maintained over data transferring across the wide area network (WAN).

SQL Server 2005 uses a replication model that is similar to a magazine distribution model:

1. A *magazine* is made available (published) by a *publisher*.
2. *Distributors* are used to distribute the publication.
3. *Readers* can subscribe to a publication so that publication is delivered to the subscriber periodically (a *push subscription*).

   This terminology is used through the SQL Server replication setup and configuration wizards.

Configure a SQL Server Publisher

To configure the master deployment server as a SQL Server publisher, perform these steps:

1. Open SQL Server Management Studio.
2. Right-click the **Replication** node, and then select **Configure Distribution**.
3. In the Configure Distribution Wizard, select **Next**.
4. On the **Distributor** page, select **will act as its own Distributor; SQL Server will create a distribution database and log**, and then select **Next**.
5. On the **Snapshot Folder** page, in the **Preparing for SQL Server Replication** section, type the UNC path to the snapshot folder created.
6. On the **Distribution Database** page, select **Next**.
7. On the **Publishers** page, select the master deployment server to set it as the distributor, and then select **Next**.
8. On the **Wizard Actions** page, select **Configure Distribution**, and then select **Next**.

9. Select **Finish**, and then select **Close** when the wizard is finished.

Enable the MDT DB for Replication

To enable the MDT DB for replication on the master deployment server, perform these steps:

1. In SQL Server Management Studio, right-click the **Replication** node, and then select **Publisher Properties**.
2. On the **Publisher Properties** page, perform these steps:

   a. Select **Publisher Databases**.

   b. Select the MDT DB, and then select **Transactional**.

   c. Select **OK**.

   The MDT DB is now configured for transactional and snapshot replication.

Create a Publication of the MDT DB

To create a publication of the MDT DB to which the child deployment servers can subscribe, perform these steps:

1. In SQL Server Management Studio, expand Replication, right-click **Local Publications**, and then select **New Publication**.
2. In the New Publication Wizard, select **Next**.
3. On the **Publication Database** page, select the MDT DB, and then select **Next**.
4. On the **Publication Type** page, select **Snapshot publication**, and then select **Next**.
5. On the **Articles** page, select all **Tables, Stored Procedures**, and **Views**, and then select **Next**.
6. On the **Articles Issues** page, select **Next**.
7. On the **Filter Table Rows** page, select **Next**.
8. On the **Snapshot Agent** page, perform these steps:

   a. Select **Create a snapshot immediately and keep the snapshot available to initialize subscriptions**.

b. Select **Schedule the Snapshot Agent to run at the following times**.

c. Select **Change**.

> ⓘ **Note**
>
> Specify a schedule that will occur one hour before the database replicates.

9. Select **Next**.
10. On the **Agent Security** page, select the account under which the snapshot agent will run, and then select **Next**.
11. On the **Wizard Actions** page, select **Create the publication**, and then select **Next**.
12. On the **Complete the Wizard** page, in the **publication name** box, type a descriptive publication name.
13. Select **Finish** to complete the wizard, and then select **Close** when the wizard has created the publication.

> ⓘ **Note**
>
> The publication will now be visible beneath the Local Publications node in SQL Server Management Studio.

Subscribe Child Deployment Servers to the Published MDT DB

Now that the MDT DB has been published, you can add the child deployment servers as subscribers to this publication; that is, that they will receive a copy of the database on a schedule so that during a deployment the client computers can query a database that is local to the network instead of going across the WAN.

To subscribe the child deployment servers to the MDT DB publication

1. In SQL Server Management Studio, go to Replication/Local Publications.
2. Right-click the publication created in the previous section, and then select **New Subscriptions**.
3. In the New Subscriptions Wizard, select **Next**.
4. On the **Publication** page, select the publication created in the previous section.

5. On the **Distribution Agent Location** page, select **Run all agents at the Distributor SERVERNAME (push subscriptions)**, and then select **Next**.

6. On the **Subscribers** page, add each of the child deployment servers by performing the following steps:

   a. Select **Add Subscriber**, and then select **Add SQL Server Subscriber**.

   b. Add each child deployment server.

   c. For each child deployment server added, in the **Subscription Database** box, select the empty MDT DB on that child deployment server.

   > ⓘ **Note**
   >
   > If the empty MDT DB has not yet been created, in the **Subscription Database** box, select the option to create a new database.

   > ⓘ **Note**
   >
   > This database must be given the same name as the MDT DB on the master deployment server. For example, if the MDT DB on the master deployment server is called *MDTDB*, create an empty database called *MDTDB* on the child deployment server.

7. Select **Next**.

8. On the **Distribution Agent Security** page, select **...** to open the **Distribution Agent Security** dialog box.

9. Type the details of the account to use for the distribution agent, and then select **Next**.

10. On the **Synchronization Schedule** page, perform these steps:

    a. In the **Agent Schedule** box, select **<Define schedule>**.

    b. Specify the schedule that should be used to replicate the database between master and child deployment servers, and then select **Next**.

11. On the **Initialize Subscription** page, select **Next**.

12. On the **Wizard Actions** page, select **Create the subscription(s)**, and then select **Next**.

13. Select **Finish**, and then select **Close** when the wizard has successfully finished.

    SQL Server replication is now configured, and the MDT DB will be replicated from the master deployment server to all child deployment servers that have been subscribed to it on a periodic basis.

Configure CustomSettings.ini

The LTI deployment infrastructure has now been successfully created, and each location will contain an LTI deployment server, with a replicated copy of:

- The deployment share
- The MDT DB
- The LiteTouchPE_x86 Windows PE environment that has been added to Windows Deployment Services

  Now, you can configure the CustomSettings.ini file for the deployment share to use the deployment content (deployment share and database) from its local deployment server, the server that delivers the LiteTouchPE_x86.wim environment through Windows Deployment Services.

  When the LiteTouchPE_x86.wim file is delivered from Windows Deployment Services, a registry key is configured with the name of the Windows Deployment Services server you are using. MDT captures this server name in a variable (%WDSServer%) that you can use to configure CustomSettings.ini.

  To always use the local LTI deployment server

> ⓘ **Note**
>
> The following procedure assumes that the deployment share has been created and set as the Deployment$ share.

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share* (where *deployment_share* is the name of the deployment share to configure).
3. In the Actions pane, select **Properties**.

4. Select the **Rules** tab, and then modify the CustomSettings.ini file to configure the following properties:

    - For each SQL Server section added, configure **SQLServer** to use the server name **%WDSServer%—**for example, **SQLServer=%WDSServer%**.
    - If configuring **DeployRoot**, configure **DeployRoot** to use the **%WDSServer%** variable—for example, **DeployRoot=\\%WDSServer%\Deployment$**.

5. Select **Edit Bootstrap.ini**.
6. Configure BootStrap.ini to use the **%WDSServer%** property by adding or changing the **DeployRoot** value to **DeployRoot=\\%WDSServer%\Deployment$**.
7. Select **File**, and then select **Save** to save the changes to the BootStrap.ini file.
8. Select **OK**.

    The deployment share and LiteTouchPE_x86.wim Windows PE environment need to be updated.

9. In the Actions pane, select **Update Deployment Share**.

    The Update Deployment Share Wizard starts.

10. On the **Options** page, select the desired options for updating the deployment share, and then select **Next**.
11. On the **Summary** page, verify the details are correct, and then select **Next**.
12. On the **Confirmation** page, select **Finish**.

    The following example illustrates CustomSettings.ini after performing the steps outlined in this section.

    Sample CustomSettings.ini Configured for Scalable LTI Deployment Infrastructure

ini

```
[Settings]
Priority=CSettings,CPackages, CApps, CAdmins, CRoles, Default
Properties=MyCustomProperty

[Default]
OSInstall=Y
ScanStateArgs=/v:5 /o /c
LoadStateArgs=/v:5 /c /lac
```

```
[CSettings]
SQLServer=%WDSServer%
Instance=
Database=MDTDB
Netlib=DBNMPNTW
SQLShare=
Table=ComputerSettings
Parameters=UUID, AssetTag, SerialNumber, MacAddress
ParameterCondition=OR

[CPackages]
SQLServer=%WDSServer%
Database=MDTDB
Netlib=DBNMPNTW
SQLShare=
Table=ComputerPackages
Parameters=UUID, AssetTag, SerialNumber, MacAddress
ParameterCondition=OR
Order=Sequence

[CApps]
SQLServer=%WDSServer%
Database=MDTDB
Netlib=DBNMPNTW
SQLShare=
Table=ComputerApplications
Parameters=UUID, AssetTag, SerialNumber, MacAddress
ParameterCondition=OR
Order=Sequence

[CAdmins]
SQLServer=%WDSServer%
Database=MDTDB
Netlib=DBNMPNTW
SQLShare=
Table=ComputerAdministrators
Parameters=UUID, AssetTag, SerialNumber, MacAddress
ParameterCondition=OR

[CRoles]
SQLServer=%WDSServer%
Database=MDTDB
Netlib=DBNMPNTW
SQLShare=
Table=ComputerRoles
Parameters=UUID, AssetTag, SerialNumber, MacAddress
ParameterCondition=OR
```

Selecting a Local MDT Server When Multiple Servers Exist

In this scenario, multiple MDT servers are being used to support a high volume of simultaneous deployments and deployments across multiple sites. When an LTI deployment is initialized, the default behavior is to request a path to the MDT server to connect to and access the required files to begin the deployment process.

The Windows Deployment Wizard can use the LocalServer.xml file to present a choice of known deployment servers for each location.

Use the LocationServer.xml file by:

- Understanding the purpose and use of LocationServer.xml as described in Understanding LocationServer.xml
- Creating the LocationServer.xml file as described in Creating the LocationServer.xml File
- Adding the LocationServer.xml file to the Extra Files directory as described in Adding the LocationServer.xml File to the Extra Files Directory
- Updating the BootStrap.ini file as described in Updating the BootStrap.ini File
- Updating the deployment share as described in Updating the Deployment Share

  This scenario assumes that MDT is configured on a deployment server.

Understanding LocationServer.xml

First, you must understand How MDT uses LocationServer.xml. During LTI, MDT scripts read and process the BootStrap.ini file to gather initial information about the deployment. This happens before a connection has been made to the deployment server. Therefore, the **DeployRoot** property is commonly used to specify in the BootStrap.ini file the deployment server to which it should make a connection.

If the BootStrap.ini file does not contain a **DeployRoot** property, MDT scripts load a wizard page to prompt the user for a path to the deployment server. While initializing the **HTML Application (HTA)** wizard page, MDT scripts check for the existence of the LocationServer.xml file and, if it exists, use LocationServer.xml to display available deployment servers.

Understand When to Use LocationServer.xml

MDT offers multiple ways to determine which server to connect to during an LTI deployment. Different methods for locating the deployment server are best suited for

different scenarios; therefore, it is important to understand when to use LocationServer.xml.

MDT provides several methods for automatically discovering and using the most appropriate deployment server. These methods are listed in the following table.

Expand table

Method	Details
%WDSServer%	This method is used when the MDT server is co-hosted on the Windows Deployment Services server. When an LTI deployment is initiated from Windows Deployment Services, an environmental variable—%WDSServer%—is created and populated with the name of the Windows Deployment Services server. The **DeployRoot** variable can use this variable to automatically connect to a deployment share on the Windows Deployment Services server—for example: **DeployRoot=\\%WDSServer%\Deployment$**
Location-based automation	MDT can use location-based automation in the BootStrap.ini file to determine the server to which it should deploy. Use the **Default Gateway** property to distinguish between different locations; for each **Default Gateway**, a different MDT server is specified. For more information about using location-based automation, refer to "Selecting the Methods for Applying Configuration Settings".

Each approach listed in the preceding table offers one way to automate the selection of the deployment server at a given location for certain scenarios. These approaches are targeted to specific scenarios—for example, when the MDT server is co-hosted with Windows Deployment Services.

There are other scenarios in which these approaches are not suitable—for example, if there are multiple deployment servers at a given location or automation logic is not possible (for example, the network is not segmented enough to allow location determination or the MDT server is separated from Windows Deployment Services).

In these scenarios, the LocationServer.xml file provides a flexible way to present this information at deployment time without requiring knowledge of server names and deployment share names.

Creating the LocationServer.xml File

To present a list of available deployment servers during an LTI deployment, create a LocationServer.xml file that contains details about each server. There is no default LocationServer.xml file in MDT, so create one using the following guidance.

Create a LocationServer.xml File to Support Multiple Locations

The simplest method for creating and using LocationServer.xml is to create a LocationServer.xml file and add entries for each deployment server in the environment (this can be either at the same location or at different locations).

Construct the LocationServer.xml file by creating a new section for each server, and then adding the following information:

- A unique identifier
- A location name, used to present an easily identifiable name for that location
- A UNC path to the MDT server for that location

  The following illustrates how the LocationServer.xml file is created using each of these properties using a sample LocationServer.xml file configured for multiple locations.

  Example LocationServer.xml File to Support Multiple Locations

XML

```
<?xml version="1.0" encoding="utf-8" ?>
<servers>
    <QueryDefault></QueryDefault>
    <server>
        <serverid>1</serverid>
        <friendlyname>
          Contoso HQ, Seattle, USA
        </friendlyname>
        <UNCPath>\\STLDS01\Deployment$</UNCPath>
    </server>
    <server>
        <serverid>2</serverid>
        <friendlyname>
          Contoso NYC, New York, USA
        </friendlyname>
        <UNCPath>\\NYCDS01\Deployment$</UNCPath>
    </server>
</servers>
```

Using this format, specify different server entries for each location or for situations in which there are multiple servers within a single location by specifying a different server entry for each server at that location, as shown in the following example.

Example LocationServer.xml File to Support Multiple Servers at Multiple Locations

XML

```
<?xml version="1.0" encoding="utf-8" ?>
<servers>
    <QueryDefault></QueryDefault>
    <server>
        <serverid>1</serverid>
        <friendlyname>
          Contoso HQ DS1, Seattle, USA
        </friendlyname>
        <UNCPath>\\STLDS01\Deployment$</UNCPath>
    </server>
    <server>
        <serverid>2</serverid>
        <friendlyname>
          Contoso HQ DS2, Seattle, USA
        </friendlyname>
        <UNCPath>\\STLDS02\Deployment$</UNCPath>
    </server>
</servers>
```

Create a LocationServer.xml File to Load Balance Multiple Servers at Different Locations

Using LocationServer.xml, specify multiple servers per location entry, and then perform basic load balancing so that when a location is chosen, MDT automatically selects a deployment server from the list of available servers. To provide this functionality, the LocationServer.xml file supports specifying a weighting metric.

The following illustrates a sample LocationServer.xml file configured for multiple servers at different locations.

Example LocationServer.xml File for Different Locations

XML

```
<?xml version="1.0" encoding="utf-8" ?>
<servers>
    <QueryDefault></QueryDefault>
    <server>
        <serverid>1</serverid>
        <friendlyname>
```

```
            Contoso HQ, Seattle, USA
        </friendlyname>
        <Server1>\\STLDS01\Deployment$</Server1>
        <Server2>\\STLDS02\Deployment$</Server2>
        <Server3>\\STLDS03\Deployment$</Server3>
        <Server weight="1">\\STLDS01\Deployment$</Server>
        <Server weight="2">\\STLDS02\Deployment$</Server>
        <Server weight="4">\\STLDS03\Deployment$</Server>
    </server>
    <server>
        <serverid>2</serverid>
        <friendlyname>
          Contoso NYC, New York, USA
        </friendlyname>
        <UNCPath>\\NYCDS01\Deployment$</UNCPath>
    </server>
</servers>
```

Specify the weighting metric by using the <server weight> tag, which MDT uses in the server-selection process. The likelihood of a server being selected is calculated by:

Server weight/sum of all server weights

In the previous example, the three servers at Contoso HQ are listed as 1, 2, and 4. The likelihood of a server with a weighting of 2 being selected becomes 2 in 7. Therefore, to use the weighting system, determine the capacity of the servers available at a location, and weight each server by the server's capacity in relation to each of the other servers.

Adding the LocationServer.xml File to the Extra Files Directory

After you have created the LocationServer.xml file, add it to the LiteTouch_x86 and LiteTouch_x64 Windows PE boot images in the ***X:\Deploy\Control folder***. Using the Deployment Workbench, add other files and folders to these Windows PE images by specifying an additional directory to add in the deployment share properties.

To add LocationServer.xml to the deployment share

1. Create a folder called *Extra Files* in the root deployment share folder (for example, D:\Production Deployment Share\Extra Files).

2. Create a folder structure in the Extra Files folder that mirrors the Windows PE location where the additional file should reside.

   For example, the LocationServer.xml file must reside in the \Deploy\Control folder in Windows PE; therefore, create the same folder structure under Extra Files (for example, D:\Production Deployment Share\Extra Files\Deploy\Control).

3. Copy LocationServer.xml to the *deployment_share*\Extra Files\Deploy\Control folder (where *deployment_share* is the fully qualified path to the root folder of the deployment share).

4. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.

5. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share* (where *deployment_share* is the name of the deployment share to configure).

6. In the Actions pane, select **Properties**.

7. In the ***deployment_shareProperties*** dialog box (where deployment_share is the name of the deployment share), perform these steps:

   a. Select the **Windows PE platform Settings** tab (where *platform* is the architecture of the Windows PE image to be configured).

   b. In the **Windows PE Customizations** section, in the **Extra directory to add** box, type ***path*** (where *path* is the fully qualified path to the Extra Files folder—for example, D:\Production Deployment Share\Extra Files), and then select **OK**.

Updating the BootStrap.ini File

When you create a deployment share using the Deployment Workbench, a **DeployRoot** property is automatically created and populated in the BootStrap.ini file. Because the LocationServer.xml file is used to populate the **DeployRoot** property, you must remove this value from the BootStrap.ini file.

To remove the DeployRoot property from BootStrap.ini

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.

2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share* (where *deployment_share* is the name of the deployment share to configure).

3. In the Actions pane, select **Properties**.

4. In the ***deployment_shareProperties*** dialog box (where *deployment_share* is the name of the deployment share), select the **Rules** tab, and then select **Edit BootStrap.ini**.

5. Remove the **DeployRoot** value (for example, **DeployRoot=\\Server\Deployment$**).
6. Select **File**, and then select **Save** to save the changes to the BootStrap.ini file.
7. Select **OK** to submit the changes.

Updating the Deployment Share

The deployment share must next be updated to generate a new LiteTouch_x86 and LiteTouch_x64 boot environment that contains the LocationServer.xml file and the updated BootStrap.ini file.

To update the deployment share

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share* (where *deployment_share* is the name of the deployment share to configure).
3. In the Actions pane, select **Update Deployment Share**.

   The Update Deployment Share Wizard starts.
4. On the **Options** page, select the desired options for updating the deployment share, and then select **Next**.
5. On the **Summary** page, verify the details are correct, and then select **Next**.
6. On the **Confirmation** page, select **Finish**.

> **Note**
>
> When the update process has finished, add the new LiteTouch_x86 and LiteTouch_x64 Windows PE environments back into Windows Deployment Services, or burn them to boot media to use during deployment.

Replacing an Existing Computer with a New Computer Using Lite Touch Installation

You can use MDT to deploy an image to a new computer that will substitute an existing computer in the enterprise architecture. This situation could arise when upgrading from

one operating system to another (a new operating system could require new hardware) or if the organization needs newer, faster computers for existing applications.

When replacing an existing computer with a new computer, Microsoft recommends taking into account all settings that will be migrated from one computer to another, such as user accounts and user state data. In addition, it is important to create a recovery solution in case the migration fails.

In this sample deployment, replace the existing computer (WDG-EXIST-01) with a new computer (WDG-NEW-02) in the CORP domain by capturing user state data from WDG-EXIST-01 and saving it to a network share. Then, deploy an existing image to WDG-NEW-02, and finally restore the captured user state data to WDG-NEW-02. The deployment will be performed from a deployment server (WDG-MDT-01).

In MDT, use the Standard Client Replace Task Sequence template to create a task sequence that will perform all the necessary deployment tasks.

This demonstration assumes that:

- MDT has been installed on the deployment server (WDG MDT 01)
- The deployment share has already been created and populated, including operating system images, applications, and device drivers
- An image of a reference computer has already been captured and will be deployed to the new computer (WDG NEW 02)
- A network shared folder (UserStateCapture$) has been created and shared on the deployment server (WDG MDT 01) with the appropriate share permissions

  A deployment share should exist prior to beginning this sample. For more information about creating a deployment share, see the section, "Managing Deployment Shares in the Deployment Workbench", in the MDT document *Using the Microsoft Deployment Toolkit*.

Step 1: Create a Task Sequence to Capture the User State

Create MDT task sequences in the Task Sequences node in the Deployment Workbench using the New Task Sequence Wizard. To perform the first part of the Replace Computer deployment scenario (capturing the user state on the existing computer), select the Standard Client Replace Task Sequence template in the New Task Sequence Wizard.

To create a task sequence to capture the user state in the Replace Computer deployment scenario

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/ *deployment_share*/Task Sequences (where *deployment_share* is the name of the deployment share to configure).
3. In the Actions pane, select **New Task Sequence**.

   The New Task Sequence Wizard starts.
4. Complete the New Task Sequence Wizard by using the following information. Accept the default values unless otherwise specified.

Expand table

On this wizard page	Do this
General Settings	1. In **Task sequence ID**, type **VISTA_EXIST**. 2. In **Task sequence name**, type **Perform Replace Computer Scenario on Existing Computer**. 3. Select **Next**.
Select Template	In **The following task sequence templates are available. Select the one you would like to use as a starting point**, select **Standard Client Replace Task Sequence**, and then select **Next**.
Summary	Verify that the configuration details are correct, and then select **Next**.
Confirmation	Select **Finish**.

The New Task Sequence Wizard finishes, and the **VISTA_EXIST** task sequence is added to the list of task sequences.

Step 2: Create a Task Sequence to Deploy Operating System and Restore the User State

Create MDT task sequences in the Task Sequences node in the Deployment Workbench by using the New Task Sequence Wizard. To perform the second part of the Replace Computer deployment scenario (deploying the operating system, and then restoring the user state on the existing computer), select the Standard Client Task Sequence template in the New Task Sequence Wizard.

To create a task sequence to deploy the user state in the Replace Computer deployment scenario

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/Task Sequences (where *deployment_share* is the name of the deployment share to configure).
3. In the Actions pane, select **New Task Sequence**.

   The New Task Sequence Wizard starts.
4. Complete the New Task Sequence Wizard by using the following information. Accept the default values unless otherwise specified.

Expand table

On this wizard page	Do this
General Settings	1. In **Task sequence ID**, type **VISTA_NEW**. 2. In **Task sequence name**, type **Perform Replace Computer Scenario on New Computer**. 3. Select **Next**.
Select Template	In **The following task sequence templates are available**. **Select the one you would like to use as a starting point**, select **Standard Client Task Sequence**, and then select **Next**.
Select OS	In **The following operating system images are available to be deployed with this task sequence**. Select one to use, select ***captured_vista_image*** (where *captured_vista_image* is the captured image the reference computer added to the Operating Systems node in the Deployment Workbench), and then select *Next*.
Specify Product Key	Select **Do not specify a product key at this time**, and then select **Next**.
OS Settings	1. In **Full Name**, type **Woodgrove Employee**. 2. In **Organization**, type **Woodgrove Bank**. 3. In **Internet Explorer Home Page**, type **http://www.woodgrovebank.com**. 4. Select **Next**.
Admin Password	In **Administrator Password** and **Please confirm Administrator Password**, type **P@ssw0rd**, and then select **Finish**.
Confirmation	Select **Finish**.

The New Task Sequence Wizard finishes, and the **VISTA_NEW** task sequence is added to the list of task sequences.

Step 3: Customize the MDT Configuration Files

When the MDT task sequence has been created, customize the MDT configuration files that provide the configuration settings for capturing user state information. Specifically, customize the CustomSettings.ini file by modifying the file in the properties of the deployment share created earlier in the deployment process. In a later step, the deployment share will be updated to ensure that the configuration file is updated in the deployment share.

To customize the MDT configuration files for capturing user state information

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share* (where *deployment_share* is the name of the deployment share to configure).
3. In the Actions pane, select **Properties**.

   The **Properties** dialog box appears.
4. In the **Properties** dialog box, select the **Rules** tab.
5. On the **Rules** tab, modify the CustomSettings.ini file to reflect the necessary changes as shown in the following example. Make any additional modifications the environment requires.

   Customized CustomSettings.ini File

   ini

   ```ini
   [Settings]
   Priority=Default
   Properties=MyCustomProperty

   [Default]
   OSInstall=Y

   UDShare=\\WDG-MDT-01\UserStateCapture$
   UDDir=%OSDCOMPUTERNAME%
   UserDataLocation=NETWORK
   SkipCapture=NO
   SkipAdminPassword=YES
   ```

```
SkipProductKey=YES
```

6. In the **Properties** dialog box, select **OK**.

7. Close all open windows and dialog boxes.

Step 4: Configure the Windows PE Options for the Deployment Share

Configure the Windows PE options for the deployment share in the Deployment Shares node in the Deployment Workbench.

> **Note**
>
> If the device drivers for the existing computer (WDG-EXIST-01) and the new computer (WDG-NEW-01) are included with Windows Vista, skip this step and proceed with the following step.

To configure the Windows PE options for the deployment share

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.

2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share* (where *deployment_share* is the name of the deployment share to configure).

3. In the Actions pane, select **Properties**.

   The **Properties** dialog box appears.

4. In the **Properties** dialog box, on the **Windows PE *platform* Components** tab (where *platform* is the architecture of the Windows PE image to be configured), in **Selection profile**, select ***device_drivers*** (where *device_drivers* is the name of the device driver selection profile), and then select **OK**.

Step 5: Update the Deployment Share

After configuring the Windows PE options for the deployment share, update the deployment share. Updating the deployment share updates all the MDT configuration files and generates a customized version of Windows PE. The customized version of

Windows PE is used to start the reference computer and initiate the LTI deployment process.

To update the deployment share in the Deployment Workbench

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share* (where *deployment_share* is the name of the deployment share to configure).
3. In the Actions pane, select **Update DeploymentShare**.

   The Update Deployment Share Wizard starts.
4. On the **Options** page, select the desired options for updating the deployment share, and then select **Next**.
5. On the **Summary** page, verify the details are correct, and then select **Next**.
6. On the **Confirmation** page, select **Finish**.

   The Deployment Workbench starts updating the deployment share. The Deployment Workbench creates the LiteTouchPE_x86.iso and LiteTouchPE_x86.wim files (for 32-bit target computers) or LiteTouchPE_x64.iso and LiteTouchPE_x64.wim files (for 64-bit target computers) in the *deployment_share*\Boot folder (where *deployment_share* is the shared folder used as the deployment share).

Step 6: Create the LTI Bootable Media

Provide a method for starting the computer with the customized version of Windows PE created when the deployment share was updated. The Deployment Workbench creates the LiteTouchPE_x86.iso and LiteTouchPE_x86.wim files (for 32-bit target computers) or LiteTouchPE_x64.iso and LiteTouchPE_x64.wim files (for 64-bit target computers) in the *deployment_share*\Boot folder (where *deployment_share* is the shared folder used as the deployment share). Create the appropriate LTI bootable media from one of these images.

To create the LTI bootable media

1. In Windows Explorer, navigate to *deployment_share*\Boot folder (where *deployment_share* is the shared folder used as the deployment share).

2. Based on the type of computer used for the existing computer (WDG-EXIST-01) and new computer (WDG-NEW-02), perform one of the following tasks:

    - If the reference computer is a physical computer, create a CD or DVD of the ISO file.
    - If the reference computer is a VM, start the VM directly from the ISO file or from a CD or DVD of the ISO file.

Step 7: Start the Existing Computer with the LTI Bootable Media

Start the existing computer (WDG-EXIST-01) with the LTI bootable media created earlier in the process. This CD starts Windows PE on the existing computer and initiates the MDT deployment process. At the end of the MDT deployment process, the user state migration information is stored in the UserStateCapture$ shared folder.

> **Note**
>
> You can also initiate the MDT process by starting the target computer from Windows Deployment Services. For more information, see the section, "Preparing Windows Deployment Services", in the MDT document *Using the Microsoft Deployment Toolkit*.

To start the existing computer with the LTI bootable media

1. Start WDG-EXIST-01 with the LTI bootable media created earlier in the process.

    Windows PE starts, and then the Windows Deployment Wizard starts.

2. Complete the Windows Deployment Wizard using the following information. Accept the default values unless otherwise specified.

Expand table

On this wizard page	Do this
Welcome to Deployment	Select **Run the Deployment Wizard** to install a new operating system, and then select **Next**.
Specify Credentials for connecting to network shares.	1. In **User Name**, type **Administrator**. 2. In **Password**, type **P@ssw0rd**. 3. In **Domain**, type **CORP**. 4. Select **OK**.

On this wizard page	Do this
Select a task sequence to execute on this computer.	Select *Perform Replace Computer Scenario on Existing Computer*, and then select **Next**.
Specify where to save your data and settings	Select **Next**.
Specify where to save a complete computer backup	Select **Do not back up the existing computer**, and then select **Next**.
Ready to begin	Select **Begin**.

If any errors or warnings occur, consult the MDT document *Troubleshooting Reference*.

3. In the **Deployment Summary** dialog box, select **Details**.

   If any errors or warnings occurred, review the errors or warnings and record any diagnostic information.

4. In the **Deployment Summary** dialog box, select **Finish**.

   The user state migration information is captured and is stored in the network shared folder (UserStateCapture$) created earlier in the process.

Step 8: Start the New Computer with the LTI Bootable Media

Start the new computer (WDG-NEW-02) with the LTI bootable media created earlier in the process. This CD starts Windows PE on the reference computer and initiates the MDT deployment process. At the end of the MDT deployment process, Windows Vista is deployed on the new computer and the captured user state migration information is restored to the new computer.

> ⓘ **Note**
>
> You can also initiate the MDT process by starting the target computer from Windows Deployment Services. For more information, see the section, "Preparing Windows Deployment Services", in the MDT document *Using the Microsoft Deployment Toolkit*.

To start the new computer with the LTI bootable media

1. Start WDG-NEW-02 with the LTI bootable media created earlier in the process.

   Windows PE starts, and then the Windows Deployment Wizard starts.

2. Complete the Windows Deployment Wizard by using the following information. Accept the default values unless otherwise specified.

Expand table

On this wizard page	Do this
Welcome to Deployment	Select **Run the Deployment Wizard to install a new operating system**, and then select **Next**.
Specify Credentials for connecting to network shares.	1. In **User Name**, type **Administrator**. 2. In **Password**, type **P@ssw0rd**. 3. In **Domain**, type **CORP**. 4. Select **OK**.
Select a task sequence to execute on this computer.	Select **Perform Replace Computer Scenario on New Computer**, and then select **Next**.
Configure the computer name	In **Computer name**, type **WDG-NEW-02**, and then select **Next**.
Join the computer to a domain or workgroup	Select **Next**.
Specify whether to restore user data	1. Select **Specify a location**. 2. In **Location**, type **\\WDG-MDT-01\UserStateCapture$\WDG-EXIST-01**. 3. Select **Next**.
Locale Selection	Select **Next**.
Set the Time Zone	Select **Next**.
Specify whether to capture an image	Select **Do not capture an image of this computer**, and then select **Next**.
Specify the BitLocker configuration	Select **Do not enable BitLocker for this computer**, and then select **Next**.
Ready to begin	Select **Begin**.

   If any errors or warnings should occur, consult the MDT document *Troubleshooting Reference*.

3. In the **Deployment Summary** dialog box, select **Details**.

If any errors or warnings occurred, review the errors or warnings and record any diagnostic information.

4. In the **Deployment Summary** dialog box, select **Finish**.

   Windows Vista is now installed on the new computer and the captured user state migration information is also restored.

Integrating Custom Deployment Code into MDT

It is common for a deployment team to have complex requirements, specific to their target environment, that are not met by the Deployment Workbench predefined task sequence actions or by default MDT configuration files. In this situation, implement custom code to meet their requirements.

Integrate custom deployment code into MDT by:

- Choosing a scripting language as described in Choosing the Appropriate Scripting Language
- Leveraging ZTIUtility.vbs as described in Understanding How to Leverage ZTIUtility
- Integrating custom deployment code as described in Integrating Custom Deployment Code

  The following sections assume that MDT is configured on a deployment server.

Choosing the Appropriate Scripting Language

Although any code that can be run on Windows or Windows PE can be called as an application installation or through an MDT task sequence step, Microsoft recommends using scripts in the form of .vbs or .wsf files.

The advantage of using .wsf files is built-in logging in addition to some other predefined functions already used by the ZTI and LTI processes. These functions are available in the ZTIUtility script distributed with MDT.

When referenced from a custom script, the ZTIUtility script initializes the MDT environment and setup classes. These classes are available:

- **Logging**. This class provides the logging functionality that all MDT scripts use. It also creates a single log file for each script run during deployment and a

consolidated log file of all scripts. These log files are created in a format designed to be read by CMTrace.

- **Environment**. This class configures environment variables gathered through WMI and MDT rule processing and allows them to be referenced directly from the script. This allows deployment properties to be read, giving access to all the configuration information used by the ZTI and LTI processes.

- **Utility**. This class provides general utilities that are used throughout ZTI and LTI scripts. Microsoft recommends that any time custom code is developed this class should be examined to see if any code can simply be reused. Additional information about some of the functionality provided in this class is included later in this section.

- **Database**. This class performs functions like connecting to databases and reading information from databases. In general, accessing the database class directly is not recommended; instead, rule processing should be used to perform database lookups.

- **Strings**. This class performs common string processing routines like creating a delimited list of items, displaying a hex value, trimming white space from a string, right aligning a string, left aligning a string, forcing a value to string format, forcing a value to array format, generating a random globally unique identifier (GUID), and Base64 conversions.

- **FileHandling**. This class performs functions like normalizing paths and copying, moving, and deleting files and folders.

- **clsRegEx**. This class performs regular expression functions.

  In MDT, a couple of changes have been implemented to the script architecture to make client Microsoft Visual Basic Scripting Edition (VBScript) more robust and reliable. These changes include:

- Extensive changes to ZTIUtility.vbs (the main script library), including new APIs and better error handling

- A new look to the overall structure of the ZTI_*xxx*.wsf scripts

  The overall structure of the MDT scripts has also changed. Most MDT scripts are now encapsulated within VBScript **Class** objects. The class is initialized and called with the **RunNewInstance** function.

> ⓘ **Note**

Most existing MDT 2008 Update 1 scripts will work as-is in MDT, even with the extensive changes to ZTIUtility.vbs, as most MDT scripts will include ZTIUtility.vbs.

Understanding How to Leverage ZTIUtility

The ZTIUtility.vbs file contains object classes that can be leveraged in your custom code. Integrate custom code with MDT by using the:

- Logging class defined in ZTIUtility.vbs as described in Use the ZTIUtility Logging Class
- Environment class defined in ZTIUtility.vbs as described in Use the ZTIUtility Environment Class
- Utility class defined in ZTIUtility.vbs as described in Use the ZTIUtility Utility Class

Use the ZTIUtility Logging Class

The logging class in ZTIUtiliy.vbs provides a simple mechanism for custom code to log status information, warnings, and errors in the same manner as other scripts during a ZTI or LTI deployment. This standardization also ensures that the **LTI Deployment Summary** dialog box correctly reports the status of any custom code that is run.

The following illustrates an example custom code script that uses the **oLogging.CreateEntry** and **TestAndFail** functions to log different types of messages, depending on the results of the various script actions.

Example Script Using ZTIUtility Logging: ZTI_Example.wsf

Visual Basic Script

```
<job id="ZTI_Example">
<script language="VBScript" src="ZTIUtility.vbs"/>
<script language="VBScript">

' //----------------------------------------------------------
' //
' // Copyright (c) Microsoft Corporation.  All rights reserved
' // Microsoft Deployment Toolkit Solution Accelerator
' // File: ZTI_Example.wsf
' //
' // Purpose: Example of scripting with the
' //          Microsoft Deployment Toolkit.
' //
' // Usage: cscript ZTI_Example.wsf [/debug:true]
' //
```

```
' //***************************************************

Option Explicit
RunNewInstance

'//----------------------------------------------------------
'// Main Class
'//----------------------------------------------------------
Class ZTI_Example

'//----------------------------------------------------------
'// Main routine
'//----------------------------------------------------------

Function Main()

  Dim iRetVal
  Dim sScriptPath

  iRetVal = SUCCESS

  oLogging.CreateEntry "Begin example script...", _
    LogTypeInfo

  ' %ServerA% is a generic variable available within
  ' every CustomSettings.ini file.

  sScriptPath = "\\" & oEnvironment.Item("ServerA") & _
    "\public\products\Applications\User\Technet\USEnglish"

  ' Validate a connection to server, net connect with
  ' credentials if necessary.
  iRetVal = oUtility.ValidateConnection( sScriptPath )
  TestAndFail iRetVal, 9991, "Validate Connection to [" & _
    sScriptPath & "]"

  'Run Setup Program

  iRetVal = oUtility.RunWithHeartbeat( """ & _
    sScriptPath & "\setup.exe"" /?" )
  TestAndFail iRetVal, 9991, "RunWithHeartbeat [" & _
    sScriptPath & "]"

  'Perform any cleanup from installation process

  oShell.RegWrite "HKLM\Software\Microsoft\SomeValue", _
    "Done with Execution of XXX.", "REG_SZ"

  Main = iRetVal

End Function

End Class
```

```
</script>
</job>
```

> **Note**
>
> If you want to continue using scripts that call **ZTIProcess()** with **ProcessResults()**, you can continue to do so. However, certain enhanced error-handling features will not be enabled.

Use the ZTIUtility Environment Class

The environment class in ZTIUtiliy.vbs provides access to, and the ability to update, MDT properties. In preceding example, **oEnvironment.Item("Memory")** is used to retrieve the amount of available RAM; this can also be used to retrieve the value of any of the properties described in the MDT document *Toolkit Reference*.

Use the ZTIUtility Utility Class

The ZTIUtility.vbs script contains a number of commonly used utilities that any custom deployment script can use. You can add these utilities to any script the same way as the **oLogging** and **oEnvironment** classes.

The following table details some useful functions available, and their output. For a full list of available functions, refer to the ZTIUtility.vbs file.

Expand table

Function	Output
oUtility.LocalRootPath	Returns the path of the root folder being used by the deployment process on the target computer—for example, C:\MININT
oUtility.BootDevice	Returns the system boot device—for example, MULTI(0)DISK(0)RDISK(0)PARTITION(1)
oUtility.LogPath	Returns the path to the logs folder being used during the deployment—for example, C:\MININT\SMSOSD\OSDLOGS
oUtility.StatePath	Returns the path of the currently configured state store—for example, C:\MININT\StateStore

Function	Output
oUtility.ScriptName	Returns the name of the script calling the function—for example, Z-RAMTest
oUtility.ScriptDir	Returns the path to the script that is calling the function—for example, \\server_name\Deployment$\Scripts
oUtility.ComputerName	Determines the computer name that will be used during the build process—for example, computer_name
oUtility.ReadIni(file, section, item)	Allows the specified item to be read from an .ini file
oUtility.WriteIni(file, section, item, value)	Allows the specified item to be written to an .ini file
oUtility.Sections(file)	Reads the sections of an .ini file and stores them in an object for reference
oUtility.SectionContents(file, section)	Reads the contents of the specified .ini file and stores them in an object
oUtility.RunWithHeartbeat(sCmd)	When the command is run, write heartbeat information to the logs every 0.5 seconds
oUtility.FindFile **(sFilename,sFoundPath)**	Searches for the specified file in the DeployRoot folder and standard subfolders, including Servicing, Tools, USMT, Templates, Scripts, and Control
oUtility.findMappedDrive(sServerUNC)	Checks to see whether a drive is mapped to the specified UNC path and returns the drive letter
oUtility.ValidateConnection(sServerUNC)	Checks to see whether there is an existing connection to the server specified and, if there is not, attempts to create one
MapNetworkDrive **(sShare, SDomID, sDomPwd)**	Maps a drive letter to the UNC path specified as the share and returns the drive letter used; returns an error if unsuccessful
VerifyPathExists(strPath)	Verifies that the specified path exists
oEnvironment.Substitute(sVal)	Given a string, expands any variables or functions within that string
oEnvironment.Item **(sName)**	Reads or writes a variable to a persistent store

Function	Output
oEnvironment.Exists **(sName)**	Tests to see whether the variable exists
oEnvironment.ListItem **(sName)**	Reads or writes a variable of type **array** to a persistent store
oLogging.ReportFailure **(sMessage, iError)**	Used to perform a structured exit if an unrecoverable error is detected
oLogging.CreateEvent **(iEventID, iType, sMessage, arrParms)**	Writes a message to the log file and posts the event to a defined server
oLogging.CreateEntry **(sLogMsg, iType)**	Writes a message to the log file
TestAndFail(iRc, iError, sMessage)	Exits the script with **iError** if **iRc** is false or fail
TestAndLog(iRc , sMessage)	Logs a warning only if **iRc** is false or fail

Integrating Custom Deployment Code

Custom deployment code can be integrated into the MDT process in several ways; however, regardless of the method used, the following two rules should be met:

- The custom deployment code script name should always begin with the letter Z.
- The custom deployment code should be placed in the Scripts folder on the deployment share—for example, D:\Production Deployment Share\Scripts.

  The most frequently used methods for integrating custom code that also ensure consistent logging are:

- Deploy the code as an MDT application
- Launch the code as an MDT task sequence command
- Launch the code as a user exit script

Deploy Custom Code as an MDT Application

Custom deployment code can be imported into the Deployment Workbench and managed the same way as any other application.

To create a new application to run custom deployment code

1. Copy the custom deployment code to the *deployment_share*\Scripts folder (where *deployment_share* is the fully qualified path to the deployment share).
2. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
3. In the Deployment Workbench console tree, go to Deployment Shares/*deployment_share*/Applications (where *deployment_share* is the name of the deployment share to configure).
4. In the Actions pane, select **New Application**.

   The New Application Wizard starts.
5. Complete the New Application Wizard using the following information. Accept defaults unless otherwise specified.

Expand table

On this wizard page	Do this
Application Type	Select **Application without source files or elsewhere on the network**, and then select **Next**.
Details	Complete this page based on the information from the application, and then select **Next**.
Command Details	1. In the **Command line** box, type **cscript.exe %SCRIPTROOT%\custom_code** (where *custom_code* is the name of the custom code that has been developed). 2. In the **Working directory** box, type ***working_directory*** (where working_directory is the name of the working directory of the custom code; this is typically the same folder specified in the **Command line** box). 3. Select **Next**.
Summary	Verify that the configuration settings are correct, and then select **Next**.
Confirmation	Select **Finish**.

The application appears in the Applications node in the Deployment Workbench.

Add the Custom Code as a Task Sequence Step

Custom deployment code can be called directly from any point within a task sequence; this gives access to the usual task sequence rules and options.

To add the custom deployment code to an existing task sequence

1. Copy the custom deployment code to the *deployment_share*\Scripts folder (where *deployment_share* is the fully qualified path to the deployment share).
2. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
3. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share/Task Sequences* (where *deployment_share* is the name of the deployment share to configure).
4. In the details pane, select ***task_sequence*** (where *task_sequence* is the name of the task sequence that runs the custom code).
5. In the Actions pane, select **Properties**.
6. In the ***task_sequenceProperties*** dialog box, select the **Task Sequence** tab.
7. In the console tree, go to *group* (where *group* is the group to add the task sequence step).
8. Select **Add**, select **General**, and then select **Run Command Line**.
9. In the console tree, select **Run Command Line**, and then select the **Properties** tab.
10. In the **Name** box, type ***name*** (where *name* is a descriptive name of the custom code).
11. On the **Properties** tab, in the **Command line** box, type ***command_line*** (where *command_line* is the command to run the custom code—for example, **cscript.exe %SCRIPTROOT%\CustomCode.vbs**).
12. In the **Start in** box, type ***path*** (where *path* is the fully qualified path to the working folder of the custom code; typically, this is the same path specified in the **Command line** box), and then select **OK**.

    The newly created task sequence step appears in the list of task sequence steps.

Run Custom Code as a User Exit Script

It also is possible to run the custom code as a user exit script from CustomSettings.ini using the **UserExit** directive. This provides a mechanism for information to be passed into the CustomSettings.ini rule validation process and provides a dynamic update of MDT properties

For more information on user exit scripts and the **UserExit** directive, see the section, "User Exit Scripts in the CustomSettings.ini File", in the MDT document *Using the Microsoft Deployment Toolkit*.

Installing Device Drivers Using Various Installation Methods

In this scenario, you use MDT to deploy an operating system to different types of hardware. As part of the deployment process, identify and install device drivers so that each hardware type will function correctly. There are two main types of device drivers; each must be handled differently during the deployment process:

- Device drivers that contain an .inf file that can be used to import the device driver into the Deployment Workbench
- Device drivers that are packaged as an application, and that must be installed as an application

  Using MDT, you can handle both types of drivers as part of an operating system deployment.

  Install device drivers by:

- Determining methods for installing each device driver as described in Determining Which Method to Use to Install a Device Driver
- Using the out-of-box drivers method as described in Installing Device Drivers Using the Out-of-Box Drivers Method
- Installing them as applications as described in Installing Device Drivers as Applications

  This scenario assumes that MDT is running on a deployment server.

Determining Which Method to Use to Install a Device Driver

Hardware manufacturers release device drivers in one of two forms:

- As a package that you can extract and that contains .inf files used to import the driver into the Deployment Workbench
- As an application that you must install using traditional application installation processes

  Device driver packages that can be extracted to access .inf files can use the MDT automatic driver detection and installation process by first importing the driver into the Out-of-Box Drivers node in the Deployment Workbench.

  Device driver packages that cannot be extracted to isolate .inf files or those that do not work correctly without first being installed using an application installer such as an MSI or Setup.exe file can use the MDT Install Application feature and install the device driver during the deployment process just as for any normal application.

Installing Device Drivers Using the Out-of-Box Drivers Method

You can import device driver packages that include an .inf file to the Deployment Workbench and install them automatically as part of the deployment process. To implement this type of device driver deployment, first add the device driver to the Deployment Workbench.

To add the device driver to the Deployment Workbench

1. Download the device drivers required for the hardware types to be deployed, and extract the device driver package to a temporary location.
2. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
3. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/Out-of-Box Drivers (where *deployment_share* is the name of the deployment share to configure).
4. In the Actions pane, select **Import Drivers**.

   The Import Device Driver Wizard starts.
5. On the **Specify Directory** page, in the **Drive source** directory section, select **Browse** to go to the folder that contains the new device drivers, and then select **Next**.

   > ⓘ **Note**

> The New Device Driver Wizard will search all subdirectories of the driver source directory; therefore, if there are multiple drivers to install, extract them into folders within the same root directory, and then set the driver source directory as the root directory that holds all of the driver source folders.

6. On the **Summary** page, verify that the settings are correct, and then select **Next** to import the drivers into the Deployment Workbench.
7. On the **Confirmation** page, select **Finish**.

   If the device drivers contain boot-critical drivers such as mass storage or network class drivers, the deployment share must next be updated to generate a new LiteTouch_x86 and LiteTouch_x64 boot environment that contains the new drivers.

   To add device drivers to the Lite Touch Windows PE images

8. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
9. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share* (where *deployment_share* is the name of the deployment share to configure).
10. In the Actions pane, select **Update Deployment Share**.

    The Update Deployment Share Wizard starts.

11. On the **Options** page, select the desired options for updating the deployment share, and then select **Next**.
12. On the **Summary** page, verify that the details are correct, and then select **Next**.
13. On the **Confirmation** page, select **Finish**.

Installing Device Drivers as Applications

Device drivers that are packaged as applications and that you cannot extract to a folder containing an .inf file, in addition to driver files, should be added to the Deployment Workbench as an application for installation during the deployment process.

Applications can be specified as a task sequence step or specified in CustomSettings.ini; however, device driver applications should be installed only when the task sequence is run on a computer with the devices. To ensure this, run the task sequence step for deploying the relevant device driver applications as a conditional task sequence step.

The conditional criteria can be specified for running the task sequence step using WMI queries for the device on the target computer.

Add the Device Driver Application to the Deployment Workbench

Each device driver application must first be imported into the Deployment Workbench.

> ⓘ **Note**
>
> Configure whether the application should is visible during deployment on the **Properties** dialog box of any application by selecting or clearing the **Hide this application in the Deployment Wizard** check box. Repeat this process for each device driver application used during deployment.

To add the device driver application to the Deployment Workbench

1. Download the device driver application, and save it to a temporary location.
2. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
3. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/Applications (where *deployment_share* is the name of the deployment share to configure).
4. In the Actions pane, select **New Application**.

   The New Application Wizard starts.
5. On the **Application Type** page, select **Application with source files**, and then select **Next**.
6. On the **Details** page, type relevant details about the application, and then select **Next**.
7. On the **Source** page, in **the Source directory** section, select **Browse** to go to, and then select the directory that contains the device driver application source files. Select **OK**.
8. Select **Next**.
9. On the **Destination** page, type a name for the destination directory, and then select **Next**.

10. On the **Command Details** page, in the **Command line** section, type the command that allows silent installation of the device driver application.
11. On the **Summary** page, verify the settings are correct, and then select **Next** to import the device driver application into the Deployment Workbench.
12. On the **Confirmation** page, select **Finish**.

    After the applications are imported into the Deployment Workbench, add them to the deployment process using the appropriate logic to ensure that the application installs only when running on the correct hardware. There are different methods for achieving this:

- Specify the device driver application as part of a deployment task sequence.
- Specify the device driver application in CustomSettings.ini.
- Specify the device driver application in the MDT DB.

  Each approach is discussed in more detail in the following sections.

Specify the Device Driver Application as Part of a Task Sequence

The first method for adding a device driver application to the deployment process is to use a task sequence to add steps for each device driver application.

There are two main approaches for managing device driver applications in the task sequence:

- Create a new task sequence group for each hardware model, and then add a query to run that group of actions if the computer matches a specific hardware type.
- Create a task sequence group for hardware-specific applications, and then add queries for each task sequence action so that each task sequence step is evaluated against the hardware type and will run only if a match is found.

  To create a new task sequence group for each type of hardware

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/Task Sequences (where *deployment_share* is the name of the deployment share to configure).

3. In the details pane, select ***task_sequence*** (where *task_sequence* is the deployment task sequence that will be required to install the device driver application).
4. In the Actions pane, select **Properties**.
5. In the ***task_sequenceProperties*** dialog box, on the **Task Sequence** tab, in the details pane, go to State Restore/Windows Update (Pre-Application Installation).
6. On the **Task Sequence** tab, select **Add**, and then select **New Group**.

   This creates a new task sequence group in the task sequence. Use this new task sequence group to create the steps for installing the hardware-specific device driver applications.
7. In the details pane, select **New Group**.
8. On the **Properties** tab, in the **Name** box, type ***group_name*** (where *group_name* is the name of the group; for example, *Hardware Specific Applications - Dell Computer Corporation*).
9. On the **Options** tab, select **Add**, and then select **Query WMI**.
10. In the **Task Sequence WMI Condition** dialog box, type the following details:

    - In the **WMI namespace** box, type **root\cimv2**.
    - In the **WQL query** box, type a WMI Query Language (WQL) query using the **Win32_ComputerSystem** class to ensure that the application is installed only for a specific application type—for example:

      Select * FROM Win32_ComputerSystem WHERE Model LIKE *%hardware_model%* AND Manufacturer LIKE *%hardware_manufacturer%*

      In this example, *hardware_model* is the name of the computer model (such as Latitude D620) and *hardware_manufacturer* is the name of the computer make (such as Dell Corporation).

      The % symbol is a wildcard character that is included in the names to allow administrators to return any computer models or manufactures that contains the value specified for ***hardware_model*** or ***hardware_manufacturer***.

      For more information about WMI and WQL queries, see the section, "Add WMI Queries to Task Sequence Step Conditions", in the MDT document *Using the Microsoft Deployment Toolkit*, and see Querying with WQL.

11. Select **OK** to submit the query, and then select **OK** to submit changes to the task sequence.

> ⓘ **Note**
>
> This process must be repeated for each hardware type of each device driver application to be installed.

After the hardware-specific task sequence groups have been created, device driver applications can be added to each group.

To add device driver applications to hardware-specific task sequence groups

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/Task Sequences (where *deployment_share* is the name of the deployment share to configure).
3. In the details pane, select ***task_sequence*** (where *task_sequence* is the deployment task sequence that will be required to install the device driver application).
4. In the Actions pane, select **Properties**.
5. In the ***task_sequenceProperties*** dialog box, select the **Task Sequence** tab.
6. In the details pane, go to State Restore/*hardware_specific_group* (where *hardware_specific_group* is the name of the hardware-specific group where the task sequence step will be added to install the device driver application).
7. On the **Task Sequence** tab, select **Add**, select **General**, and then select **Install Application**.

   The **Install Application** task sequence step appears in the details pane.
8. In the details pane, select **Install Application.**
9. On the **Properties** tab, select **Install a single application**, and in the **Application to install** list, select ***hardware_application*** (where *hardware_application* is the application for installing the hardware-specific application).

> ⓘ **Note**

This process must be repeated for each device driver application that needs to be used during a deployment.

Specify the Device Driver Application in CustomSettings.ini

When an LTI or ZTI deployment begins, one of the first actions to be completed is the processing of the BootStrap.ini and CustomSettings.ini control files. Both of these files contain rules that can be used to dynamically customize the deployment.

Because of the way MDT processes the CustomSettings.ini file, you can use it to add applications based on specific conditions. This logic will be used to add device driver-specific applications during deployment based on specific hardware types. Applications are referenced in CustomSettings.ini by the application's GUID, located in the Applications.xml file in the deployment share.

To locate an imported application's GUID

1. In the deployment share of the deployment server, open the Control folder—for example, D:\Production Deployment Share\Control.
2. Locate and open the Applications.xml file.
3. Locate the required application.
4. Locate the application GUID by locating the line enclosed in the application `<guid>` tags; for example, `<application guid={c303fa6e-3a4d-425e-8102-77db9310e4d0}>`.

   As part of the initialization process, both the LTI and ZTI process gather information about the computer on which it is running. As part of this process, WMI queries are performed and the values from the **Win32_ComputerSystem** class for make and manufacturer are populated as variables **%Make%** and **%Model%,** respectively.

   These values can be used during processing the CustomSettings.ini file to dynamically read sections of the file depending on the make and model detected. The following sample shows an example of the CustomSettings.ini file.

   Sample CustomSettings.ini Configured for a Hardware-Specific Application Installation

ini

```ini
[Settings]
Priority=Make, Default
```

```
Properties=MyCustomProperty

[Default]
OSInstall=Y

[Dell Computer Corporation]
Subsection=Dell-%Model%

[Dell-Latitude D620]
MandatoryApplications001={1D7DF331-47B7-472C-87B3-442597EC2F7D}

[Dell-Latitude D610]
MandatoryApplications001={c303fa6e-3a4d-425e-8102-77db9310e4d0}
```

Use the following properties to specify applications in CustomSettings.ini:

- **Applications**. This property can be used when deployment administrators do not want to present an application wizard as part of the deployment process by specifying **SkipApplications=YES** in CustomSettings.ini.
- **MandatoryApplications**. This property can be used if deployment administrators want to present the application wizard during the deployment to allow deployment engineers to select additional applications to be installed during the deployment.

  If the application wizard is used without the **MandatoryApplications** property (for example, **SkipApplications=NO**), it will overwrite applications specified by the **Applications** property.

  The previous sample shows how to use the **%Make%** and **%Model%** variable values to dynamically manipulate how the applications list is built. The values for the make and model of each type of hardware can be located using one of the following methods:

- **The System Information tool**. Use the System Summary node in this tool to identify the **System Manufacturer** (make) and **System Model** (model).
- **Windows PowerShell**. Use the **Get-WMIObject -class Win32_ComputerSystem** cmdlet to determine the make and model of the computer.
- **Windows Management Instrumentation Command Line**. Use **CSProduct Get Name, Vendor** to return the name (model) and vendor (make) of the computer.

  To modify CustomSettings.ini to add hardware-specific logic

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.

2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share* (where *deployment_share* is the name of the deployment share to configure).

3. In the Actions pane, select **Properties**.

4. Select the **Rules** tab.

5. Information typed on this tab is stored in the CustomSettings.ini file. Modify the CustomSettings.ini file entries to add logic for each hardware model that has a device driver-specific application, as described in Specify the Device Driver Application as Part of a Task Sequence.

6. Select **OK** to submit the changes.

7. In the details pane, select *deployment_share* (where *deployment_share* is the name of the deployment share to configure).

8. In the Actions pane, select **Update Deployment Share**.

   The Update Deployment Share Wizard starts.

9. On the **Options** page, select the desired options for updating the deployment share, and then select **Next**.

10. On the **Summary** page, verify the details are correct, and then select **Next**.

11. On the **Confirmation** page, select **Finish**.

    By default, all available applications are displayed in the Windows Deployment Wizard during an LTI deployment. Because device driver-specific applications are applicable only to specific hardware types, you might not want them displayed all the time. By specifying the device driver-specific application package in CustomSettings.ini, the application can be hidden using the **Hide the application in the Deployment Wizard** option in the application configuration.

    To hide an application in the Deployment Wizard

12. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.

13. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/Applications (where *deployment_share* is the name of the deployment share to configure).

14. In the details pane, select ***device_driver_application*** (where *device_driver_application* is the application to be hidden from the Deployment Wizard).

15. In the Actions pane, select **Properties**.

16. On the **General** tab, select the **Hide the application in the Deployment Wizard** check box.

17. Select **Apply**, and then close the **Properties** dialog box.

Specify the Device Driver Application in the MDT DB

The MDT DB is a database version of the CustomSettings.ini file and can be queried at deployment time for information to be used during the deployment. For more information about using the MDT DB, see "Selecting the Methods for Applying Configuration Settings".

When querying the MDT DB at deployment time, three methods are available for identifying the target computer:

- Search for the individual computer (using the MAC address, asset tag, or similar).
- Search for the location of the computer (using the default gateway).
- Search for the make and model of the computer (using WMI manufacturer or make and model queries).

  For each database entry you create, you can specify deployment properties, applications, whether to use Configuration Manager packages, and administrators. By creating make and model entries in the database, you can add the required hardware-specific device driver applications.

  To create entries in the MDT DB to allow installation of device driver applications

> ⓘ **Note**
>
> Repeat this process for each hardware make and model that requires a device driver application.

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.

2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/**deployment_share**/Advanced Configuration/Database/Make and Model (where *deployment_share* is the name of the deployment share to configure).

3. In the Actions pane, select **New**.

4. In the **Properties** dialog box, on the **Identity** tab, in the **Make** box, type ***make_name*** (where *make_name* is an easily identified name to associate with the manufacturer of the target computer).

5. In the **Model** box, type ***model_name*** (where *model_name* is an easily identified name to associate with the model of the target computer).

6. On the **Applications** tab, add each of the device driver applications required for that model of hardware.

Initiating MDT Using Windows Deployment Services

Windows Server 2008 uses Windows Deployment Services as an updated and redesigned version of Remote Installation Services, the default deployment tool in Windows Server 2003 with SP2. Using Windows Deployment Services, you can deploy Windows operating systems—particularly Windows 7, Windows Server 2008 or later operating systems—across a network using either a computer's PXE-enabled network adapter or boot media.

Before deploying Windows Deployment Services, determine which of the following integration options best suits your environment:

- Option 1. Boot computers in PXE to initiate the LTI process.
- Option 2. Deploy an operating system image from the Windows Deployment Services image store.
- Option 3. Use multicasting with MDT and the Windows Server 2008 Windows Deployment Services server role.

Option 1: Boot Computers in PXE to Initiate the LTI Process

Help minimize the cost of managing operating system deployments by starting the MDT deployment process using Windows Deployment Services in conjunction with Dynamic Host Configuration Protocol. This removes the requirement of creating and delivering bootable media to each target computer.

Create and Import the Deployment Workbench Windows PE Image into Windows Deployment Services

When creating a new MDT deployment share or modifying an existing MDT deployment share, you can create a customized Windows PE boot image. When the deployment share is updated, the Windows PE boot image is automatically generated and updated with information about the deployment share, and it will inject any additional drivers or components specified during the deployment share configuration.

The Windows PE boot image is generated as both an ISO image file, which you can write to a CD or DVD, and a bootable WIM file. You can import the WIM file to Windows Deployment Services so that computers that can boot in PXE can download and run the LTI Windows PE boot image across a network used to initialize an installation.

To create a bootable Windows PE image in the Deployment Workbench

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share* (where *deployment_share* is the name of the deployment share to configure).
3. In the Actions pane, select **Properties**.

   In the ***deployment_shareProperties*** dialog box, select the **Windows PE *platform* Settings** tab (where platform is the architecture of the Windows PE image to be configured).
4. In the **Lite Touch Boot Image Settings** area, select the **Generate a Lite Touch bootable RAM disk ISO image** check box.
5. Select the **Windows PE *platform* Components** tab (where *platform* is the architecture of the Windows PE image to be configured).
6. In the **Driver Injection** section, select the appropriate driver types to include.

   > ⓘ **Note**

This step is not necessary if Windows PE already includes the necessary device drivers.

7. In the **Driver Injection** section, in the **Selection profile** list, select the appropriate driver selection profile.
8. In the **Properties** dialog box, select **OK**.

> ⓘ **Note**
>
> This step is not necessary if Windows PE already includes the necessary device drivers.

9. In the details pane, select ***deployment_share*** (where *deployment_share* is the name of the deployment share to configure).
10. In the Actions pane, select **Update Deployment Share**.

    The Update Deployment Share Wizard starts.
11. On the **Options** page, select the desired options for updating the deployment share, and then select **Next**.
12. On the **Summary** page, verify the details are correct, and then select **Next**.
13. On the **Confirmation** page, select **Finish**.

    When this process is complete, the Boot folder in the deployment share will contain a number of boot images—for example:

    D:\Production Deployment Share\Boot\LiteTouchPE_x64.iso

    D:\Production Deployment Share\Boot\LiteTouchPE_x64.wim

    D:\Production Deployment Share\Boot\LiteTouchPE_x86.iso

    D:\Production Deployment Share\Boot\LiteTouchPE_x86.wim

    You can write the ISO files that have been generated directly to CD or DVD or use them to initialize the LTI process on new hardware. You can import the boot WIM files into Windows Deployment Services, as well, so that new computers can initialize the LTI deployment process without requiring any physical media.

    To import the Windows PE image into Windows Deployment Services

14. Start the Windows Deployment Services console, and then connect to Windows Deployment Services.

15. In the console tree, right-click **Boot Images**, and then select **Add Boot Image**.

16. Browse to the WIM image to be imported—for example, D:\Production Deployment Share\Boot\LiteTouchPE_x86.wim.

17. The import process automatically reads the metadata from the boot image, but the **Image Name** and **Image Description** values can also be edited; the **Image Name** affects the boot option information displayed by Windows Boot Manager when the client boots in PXE.

18. When the boot image has been imported, any computer that boots in PXE and receives a reply from Windows Deployment Services will be able to download the LTI boot image and initiate an LTI installation.

    Installing and configuring Windows Deployment Services is not covered in this guide. For additional information about Windows Deployment Services, see the Windows Deployment Services Guide.

Use Windows Deployment Services to Automatically Detect the Deployment Server

An additional option is available when using Windows Deployment Services to host MDT boot images when the MDT deployment share is hosted on the same server as Windows Deployment Services.

When a PXE client loads the MDT boot image, the name of the Windows Deployment Services server hosting the boot image is captured and placed in the MDTProperty **WDSServer**. You can then reference this property in the boot image's BootStrap.ini file and in the deployment share's CustomSettings.ini file by the **DeployRoot** property. Doing so results in a client that boots from Windows Deployment Services automatically using the deployment share hosted on the Windows Deployment Services server. This eliminates the need to specify a server name in any configuration file.

To set the local Windows Deployment Services server as the deployment server

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.

2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/Advanced

Configuration/Database (where *deployment_share* is the name of the deployment share to configure).

3. In the Actions pane, select **Properties**.
4. Select the **Rules** tab.

   Information typed on this tab is stored in the CustomSettings.ini file.

5. Configure the **DeployRoot** property to use the **%WDSServer%** variable—for example, **DeployRoot=\\%WDSServer%\Deployment$**.
6. Select **Edit Bootstrap.ini**.
7. Configure BootStrap.ini to use the **%WDSServer%** property by adding or changing the **DeployRoot** value to **DeployRoot=\\%WDSServer%\Deployment$**.
8. On the **File** menu, select **Save** to save the changes to the BootStrap.ini file.
9. Select **OK**.

   The deployment share needs to be updated.

10. In the details pane, select **deployment_share** (where *deployment_share* is the name of the deployment share to configure).
11. In the Actions pane, select **Update Deployment Share**.

    The Update Deployment Share Wizard starts.

12. On the **Options** page, select the desired options for updating the deployment share, and then select **Next**.
13. On the **Summary** page, verify the details are correct, and then select **Next**.
14. On the **Confirmation** page, select **Finish**.
15. Import the updated boot WIM into Windows Deployment Services.

Option 2: Deploy an Operating System Image from the Windows Deployment Services Store

If you are already using Windows Deployment Services for operating system deployment, extend the functionality of MDT by configuring it to reference the Windows Deployment Services operating system images already in use rather than using its own store and to supplement Windows Deployment Services deployments with driver

management, application deployment, update installation, rule processing, and other MDT functionality. After MDT has reference a Windows Deployment Services operating system image, you can treat it like any operating system that has been staged to an MDT deployment share.

To reference a Windows Deployment Services operating system image

> ⓘ **Note**
>
> The following steps require that at least one operating system image has previously been imported into the Windows Deployment Services server.

1. Update MDT to be able to access Windows Deployment Services images by copying the following files from the Sources folder of the Windows media to the C:\Program Files\Microsoft Deployment Toolkit\bin folder on the Windows Deployment Services server:

   - Wdsclientapi.dll
   - Wdscsl.dll
   - Wdsimage.dll
   - Wdstptc.dll (this is only applicable if copying from the Windows Server 2008 source directories)

   > ⓘ **Note**
   >
   > The Windows source directory being used must match the platform of the operating system running on the computer where MDT is installed.

2. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.

3. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/Operating Systems (where *deployment_share* is the name of the deployment share to configure).

4. In the Actions pane, select **Import Operating System**.

   The New OS Wizard starts.

5. On the **OS Type** page, select **Windows Deployment Services images**, and then select **Next**.
6. On the **WDS Server** page, type the name of the Windows Deployment Services server to be referenced—for example, **WDSSvr001**—and then select **Next**.
7. On the **Summary** page, verify the settings are correct, and then select **Next**.
8. On the **Confirmation** page, select **Finish**.

   All of the images available on the Windows Deployment Services server will now be available to MDT task sequences.

> ⓘ **Note**
>
> Importing images from Windows Deployment Services does not copy the source files from the Windows Deployment Services server to the deployment share. MDT continues to use the source files from their original location.

Option 3: Use Multicasting with MDT and the Windows Server 2008 Windows Deployment Services Role

With the release of Windows Server 2008, Windows Deployment Services was enhanced to support the deployment of images using multicast transmissions. MDT also includes updates to integrate MDT with Windows Deployment Services multicasting.

In addition, an updated Windows Automated Installation Kit (Windows AIK), version 1.1, includes Wdsmcast.exe. This allows multicast sessions to be joined manually and allows the client launching Wdsmcast.exe to copy files from an active multicast session.

The LTIApply.wsf script uses Wdsmcast.exe when it accesses operating system source files from the deployment share. LTIApply.wsf looks for Wdsmcast.exe on the deployment share either in the *deployment_share*\Tools\x86 or the *deployment_share*\Tools\x64 folder (where *deployment_share* is the name of the file system folder that contains the deployment share), depending on the version of Windows PE that is running.

When LTIApply.wsf runs it will always attempt to access and download WIM images from an existing multicast stream, but it will fall back to a standard file copy if a multicast stream does not exist.

> ⓘ **Note**

This process applies only to WIM image files.

The deployment server prerequisites for preparing for MDT multicasting are:

- The deployment server must be running Windows Server 2008 or later
- The Windows Deployment Services role must be installed from the Server Management console
- Windows AIK 1.1 for Windows Server 2008 must be installed
- MDT must be installed
- As with any deployment using MDT, at least one operating system WIM image must have been imported, either as a full set of source files or as a custom image with setup files

> **Note**
>
> It is important to use the latest version of Windows AIK for multicasting; the copy of Windows PE included in earlier versions of Windows AIK—for example, Windows AIK 1.0—does not support downloading from a multicast server.

To configure MDT for multicasting from an existing deployment share

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share* (where *deployment_share* is the name of the deployment share to configure).
3. In the Actions pane, select **Properties**.
4. On the **General** tab, select the **Enable multicast for this deployment share (requires Windows Server 2008 Windows Deployment Services)** check box.
5. Select **OK**.
6. In the Actions pane, select **Update Deployment Share**.

   The Update Deployment Share Wizard starts.
7. On the **Options** page, select the desired options for updating the deployment share, and then select **Next**.

8. On the **Summary** page, verify the details are correct, and then select **Next**.
9. On the **Confirmation** page, select **Finish**.

   The deployment share is now configured for Windows Deployment Services multicast transmission.

   This process creates an Auto-Cast Windows Deployment Services multicast transmission that directly uses the existing MDT deployment share. MDT does not create Scheduled-Cast transmissions. Also note that no additional images are imported into Windows Deployment Services and that it is not possible to use multicast for boot images, because the multicast client cannot be loaded until after Windows PE is running.

   To verify that the multicast transmission has been generated in Windows Deployment Services

10. Select **Start**, point to **Administrative Tools**, and then select **Windows Deployment Services**.
11. In the Windows Deployment Services console tree, right-click **Servers**, and then select **Add Server**.
12. In the **Add Servers(s)** dialog box, select **Local computer**, and then select **OK**.
13. In the Windows Deployment Services console tree, select **Servers**, then select ***server_name*** (where *server_name* is the name of the computer running Windows Deployment Services). Select **Multicast Transmissions**.
14. In the details pane, a new Auto-Cast transmission for the deployment share will be listed—for example, **BDD Share Deployment$**.
15. Verify that the status of the **BDD Share Deployment$** Auto-Cast transmission is set to **Active**.

    After a computer has been deployed, verify that the operating system was downloaded from a multicast transmission by examining the BDD.log file in the \Windows\Temp\DeploymentLogs folder.

    There will be two entries in the logs folder, both beginning with **Multicast transfer**; check them to verify that the transfer was successful. For more information on multicast transmissions with MDT and Windows Deployment Services, see the section, "Enable Windows Deployment Services Multicast Deployment for LTI Deployments", in the MDT document *Using the Microsoft Deployment Toolkit*.

Performing Staged Deployments Using MDT (OEM Preload)

In many organizations, computers are loaded with the operating system image before deployment to the production network. In some instances, loading the operating system image is performed by a team within the organization that is responsible for building the computers in a staging environment. In other instances, loading the operating system image is performed by the computer hardware vendor, also known as an *original equipment manufacturer* (OEM).

> ⓘ **Note**
>
> The OEM preload process is supported in MDT only for deployments performed using LTI. For Configuration Manager, use the prestaged media feature.

Overview of the OEM Preload Process in MDT

The OEM preload process is divided into three phases:

- **Phase 1**. Create a media-based image of the reference computer to be applied in the staging environment.
- **Phase 2**. Apply the reference computer image to the target computer in a staging environment.
- **Phase 3**. Complete deployment of the target computer in the production environment.

  Phase 1 and Phase 3 are typically performed by the deployment organization. Depending on the use of the OEM preload process in the organization, Phase 2 may be performed by the organization or by the computer hardware vendor that supplies the computers. If the organization performs Phase 2, then the staging environment is within the organization. If an OEM performs Phase 2, then the staging environment is in the OEM's environment.

Overview of MDT Configuration Files in the OEM Preload Process

Separate MDT configuration files (CustomSettings.ini and Bootstrap.ini) are used by the task sequences run during Phase 1 and Phase 3 of the OEM preload process. However, both configuration files exist simultaneously in different folder structures.

In the first phase, the configuration files are used during the creation of the reference computer and are stored in the folder specific to the task sequence used in that phase. The configuration files used in the third and last phase of the OEM preload process are stored in the folder that is specific to the task sequence used in that phase.

When making modifications to the configuration files, ensure that changes to the configuration file are made that corresponds to the appropriate task sequence in each OEM preload process phase.

Overview of MDT Log Files in the OEM Preload Process

Separate MDT log files are generated during Phase 1 and Phase 3 of the OEM preload process:

- The MDT log files for Phase 1 are stored in the C:\MININT and C:\SMSTSLog folders.
- The MDT log files for Phase 3 are stored in the %WINDIR%\System32\CCM\Logs folder for x86-based deployments or in the %WINDIR%\SysWow64\CCM\Logs folder for x64-based deployments.

  Use the appropriate folder when diagnosing or troubleshooting MDT-related deployment problems.

Staged Deployments Using LTI

For LTI deployments, perform the OEM preload process using a *Removable media (Media)* deployment share type. Other deployment share types are not supported for the OEM preload process.

To perform the OEM preload process, create a task sequence based on the Litetouch OEM Task Sequence task sequence template, in addition to any task sequences that will be used to deploy the target operating system. Then, create a *Removable media (Media)* deployment share that will ultimately create an ISO file of the deployment share contents, specifically the LiteTouchPE_x86.iso file or LiteTouchPE_x64.iso file (based on the target computer's processor platform). The deployment share update process also creates a folder structure that can be used to create Universal Disk Format media.

LTI OEM Preload Process—Phase 1: Create a Media-Based Image

The deployment organization performs the first phase in the OEM preload process. The final deliverable of this phase is a bootable image (such as an ISO file) or media (such as

a DVD) that is sent to the OEM or to the staging environment within the deployment organization. Most of these steps are performed in the Deployment Workbench.

To create a media-based image for delivery to the OEM or to the staging environment within the deployment organization

1. Populate the following nodes for the deployment share in the Deployment Workbench:

    - Operating Systems
    - Applications
    - Packages
    - Out-of-Box Drivers

    For more information about performing this step, see the section, "Managing Deployment Shares in the Deployment Workbench", in the MDT document *Using the Microsoft Deployment Toolkit*.

2. Create a new task sequence based on the Litetouch OEM Task Sequence task sequence template in the Deployment Workbench.

    For more information about performing this step, see the section, "Configuring Task Sequences in the Deployment Workbench", in the MDT document *Using the Microsoft Deployment Toolkit*.

3. Create one or more task sequences that will be used to deploy the target operating system on the target computer after deployment in the production environment.

    For more information about performing this step, see the section, "Configuring Task Sequences in the Deployment Workbench", in the MDT document *Using the Microsoft Deployment Toolkit*.

4. Create a selection profile that includes the applications, operating systems, drivers, packages, and task sequences required for the OEM deployment.

    For more information about performing this step, see the section, "Manage Selection Profiles", in the MDT document *Using the Microsoft Deployment Toolkit*.

5. Create deployment media.

    For more information on performing this step, see the section, "Manage LTI Deployment Media", in the MDT document *Using the Microsoft Deployment Toolkit*.

6. Update the deployment media created in the Deployment Workbench in the previous step.

   When you update the deployment media, the Deployment Workbench creates the LiteTouchMedia.iso file. For more information about performing this step, see the section, "Manage LTI Deployment Media", in the MDT document *Using the Microsoft Deployment Toolkit*.

7. Burn a DVD of the LiteTouchMedia.iso file created in the previous step.

   > ⓘ **Note**
   >
   > If delivering the ISO file to the OEM or to the organization's staging environment, this step is not necessary.

8. Deliver the ISO file or the DVD to the OEM or to the organization's staging environment.

LTI OEM Preload Process—Phase 2: Apply the Image to the Target Computer

The second phase of the OEM preload process is performed by the OEM or by the deployment team in the staging environment of the deployment organization. During this phase of the process, the .iso file or DVD created in Phase 1 is applied to the target computers. The deliverable of this phase is the image deployed on the target computers so that they are ready for deployment in the production environment.

To apply the image to the target computers

1. Start a target computer with the media created in the Phase 1.

   Windows PE starts, and then the Windows Deployment Wizard starts.

2. In the Windows Deployment Wizard, select the **OEM Preinstallation Task Sequence for Staging Environment** task sequence.

   The task sequence will start and the contents of the bootable media will be copied to the local hard disk of the target computer.

3. When the Windows Deployment Wizard is complete for the **OEM Preinstallation Task Sequence for Staging Environment** task sequence, the hard disk will be ready to initiate the remainder of the deployment process by running the Windows

Deployment Wizard for the other task sequences that are used to deploy the operating system.

The **OEM Preinstallation Task Sequence for Staging Environment** task sequence is responsible for deploying the image to the target computer and initiating the LTI process. The Windows Deployment Wizard will start a second time to run the task sequences used to deploy the operating system on the target computer.

4. Clone the contents of the first hard disk to as many target computers in the staging environment as required.

5. The target computers are delivered to the production environment for deployment.

LTI OEM Preload Process—Phase 3: Complete Target Computer Deployment

The third and final phase of the OEM preload process is performed in the deployment organization's production environment. During this phase of the process, the target computer is started and the bootable media image, placed on the hard disk in the staging environment during the previous phase, starts.

To complete deployment of the target computers in the production environment

1. Start the target computer.

   Windows PE starts, and then the Windows Deployment Wizard starts.

2. Complete the Windows Deployment Wizard using the specific configuration information for each target computer.

   For more information about completing this step, see the section, "Running the Deployment Wizard", in the MDT document *Using the Microsoft Deployment Toolkit*.

   When this phase is complete, the target computer will be ready to use in the production environment.

Using Windows PowerShell to Perform Common Tasks

The MDT administration tasks in the Deployment Workbench are performed by underlying Windows PowerShell cmdlets, which you can use to automate administrative

tasks such as those in the following sections.

You can automate MDT administration by performing the following steps:

- Create a new deployment share as described in Creating a New Deployment Share.
- Create a folder in a deployment share as described in Creating a Folder.
- Delete a folder from a deployment share as described in Deleting a Folder.
- Import a device driver into a deployment share as described in Importing a Device Driver.
- Deleting a device driver from a deployment share as described in Deleting a Device Driver.
- Import an operating system package into a deployment share as described in Importing an Operating System Package.
- Deleting an operating system package from a deployment share as described in Deleting an Operating System Package.
- Import an operating system into a deployment share as described in Importing an Operating System.
- Deleting an operating system from a deployment share as described in Deleting an Operating System.
- Create an application in a deployment share as described in Creating an Application.
- Delete an application from a deployment share as described in Deleting an Application.
- Create a task sequence in a deployment share as described in Creating a Task Sequence.
- Delete a task sequence from a deployment share as described in Deleting a Task Sequence.
- Create an MDT DB as described in Creating an MDT DB.
- Create a selection profile as described in Creating a Selection Profile.
- Update a deployment share as described in Updating a Deployment Share.

- Create a linked deployment share as described in Creating a Linked Deployment Share.
- Update a linked deployment share as described in Updating a Linked Deployment Share.
- Delete a linked deployment share as described in Deleting a Linked Deployment Share.
- Create deployment media as described in Creating Media.
- Generate deployment media as described in Generating Media.
- Delete deployment media as described in Deleting Media.

Creating a New Deployment Share

The following Windows PowerShell commands create a new deployment share at D:\Production Deployment Share named *Production$*. The new deployment share will be displayed in the Deployment Workbench as Production.

PowerShell

```
Add-PSSnapIn Microsoft.BDD.PSSnapIn
New-PSDrive -Name "DS002" -PSProvider "MDTProvider" -Root "D:\Production
Deployment Share" -Description "Production" -NetworkPath
"\\Deployment_Server\Production$" -Verbose | add-MDTPersistentDrive -Verbose
```

Creating a Folder

The following Windows PowerShell commands create an Adobe folder in the Deployment Workbench console tree at Deployment Workbench/Deployment Shares/Production/Applications.

PowerShell

```
Add-PSSnapIn Microsoft.BDD.PSSnapIn
New-PSDrive -Name "DS002" -PSProvider MDTProvider -Root "D:\Production
Deployment Share"
New-item -path "DS002:\Applications" -enable "True" -Name "Adobe" -Comments
"This folder contains Adobe software" -ItemType "folder" -Verbose remove-
psdrive DS001 -Verbose
```

> ⓘ Note

Adding "remove-psdrive" to the script ensures that the background process finishes before proceeding.

Deleting a Folder

The following Windows PowerShell commands delete the Deployment Workbench/Deployment Shares/Production/Applications/Adobe folder.

PowerShell

```
Add-PSSnapIn Microsoft.BDD.PSSnapIn
New-PSDrive -Name "DS002" -PSProvider MDTProvider -Root "D:\Production
Deployment Share"
Remove-item -path "DS002:\Applications\Adobe" -Verbose
```

> **Note**
>
> The script will fail if the folder is not empty.

Importing a Device Driver

The following Windows PowerShell commands will import the Dell 2407 WFP monitor device driver into the Production deployment share.

PowerShell

```
Add-PSSnapIn Microsoft.BDD.PSSnapIn
New-PSDrive -Name "DS002" -PSProvider MDTProvider -Root "D:\Production
Deployment Share"
Import-mdtdriver -path "DS002:\Out-of-Box Drivers\Monitor" -SourcePath
"D:\Drivers\Dell\2407 WFP" -Verbose
```

Deleting a Device Driver

The following Windows PowerShell command deletes the Dell 2407 WFP monitor driver from the Production deployment share.

PowerShell

```
Remove-item -path "DS002:\Out-of-Box Drivers\Dell Inc. Monitor 2407WFP.INF
1.0" -Verbose
```

Importing an Operating System Package

The following Windows PowerShell commands import all operating system packages located under D:\Updates\Microsoft\Vista. These operating system packages will be stored in the Production deployment share, which is in D:\Production Deployment Share.

PowerShell

```
Add-PSSnapIn Microsoft.BDD.PSSnapIn
New-PSDrive -Name "DS002" -PSProvider MDTProvider -Root "D:\Production
Deployment Share"
Import-mdtpackage -path "DS002:\Packages" -SourcePath
"D:\Updates\Microsoft\Vista" -Verbose
```

Deleting an Operating System Package

The following Windows PowerShell command deletes the specified operating system package from the Production deployment share.

PowerShell

```
Remove-item -path "DS002:\Packages\Package_1_for_KB940105 neutral x86
6.0.1.0 KB940105" -Verbose
```

Importing an Operating System

The following Windows PowerShell commands import the Windows Vista operating system located in D:\Operating Systems\Windows Vista x86. The operating system will be stored in the Production deployment share, which is in D:\Production Deployment Share.

PowerShell

```
Add-PSSnapIn Microsoft.BDD.PSSnapIn
New-PSDrive -Name "DS002" -PSProvider MDTProvider -Root "D:\Production
Deployment Share"
Import-mdtoperatingsystem -path "DS002:\Operating Systems" -SourcePath
"D:\Operating Systems\Windows Vista x86" -DestinationFolder "Windows Vista
x86" -Verbose
```

Deleting an Operating System

The following Windows PowerShell command deletes the Windows Vista HOMEBASIC operating system from the Production deployment share.

```
PowerShell

Remove-item -path "DS002:\Operating Systems\Windows Vista HOMEBASIC in
Windows Vista x86 install.wim" -Verbose
```

Creating an Application

The following Windows PowerShell commands create the Adobe Reader 9 application using source files from D:\Software\Adobe\Reader 9. The application will be stored in the Production deployment share, which is in D:\Production Deployment Share.

```
PowerShell

Add-PSSnapIn Microsoft.BDD.PSSnapIn
New-PSDrive -Name "DS002" -PSProvider MDTProvider -Root "D:\Production
Deployment Share"
Import-MDTApplication -path "DS002:\Applications" -enable "True" -Name
"Adobe Reader 9" -ShortName "Reader" -Version "9" -Publisher "Adobe" -
Language "" -CommandLine "setup.exe" -WorkingDirectory ".\Applications\Adobe
Reader 9" -ApplicationSourcePath "D:\Software\Adobe\Reader 9" -
DestinationFolder "Adobe Reader 9" -Source ".\Applications\Adobe Reader 9" -
Verbose
```

Deleting an Application

The following Windows PowerShell command deletes the Adobe Reader 9 application from the Production deployment share.

```
PowerShell

Remove-item -path "DS002:\Applications\Adobe Reader 9" -Verbose
```

Creating a Task Sequence

The following Windows PowerShell commands create the **Windows Vista Production Build** task sequence in the Production deployment share, which is located in D:\Production Deployment Share.

```
PowerShell
```

```
Add-PSSnapIn Microsoft.BDD.PSSnapIn
New-PSDrive -Name "DS002" -PSProvider MDTProvider -Root "D:\Production
Deployment Share"
Import-mdttasksequence -path "DS002:\Task Sequences" -Name "Windows Vista
Business Production Build" -Template "Client.xml" -Comments "Approved for
use in the production environment.  This task sequence uses the Standard
Client task sequence template" -ID "Vista_Ref" -Version "1.0" -
OperatingSystemPath "DS002:\Operating Systems\Windows Vista BUSINESS in
Windows Vista x86 install.wim" -FullName "Fabrikam User" -OrgName "Fabrikam"
-HomePage "http://www.Fabrikam.com" -AdminPassword "secure_password" -
Verbose
```

Deleting a Task Sequence

The following Windows PowerShell command deletes the **Windows Vista Production Build** task sequence from the Production deployment share.

PowerShell

```
Remove-item -path "DS002:\Task Sequences\Windows Vista Business Production
Build" -force -Verbose
```

Creating an MDT DB

The following Windows PowerShell commands create a new MDT DB on the *deployment_server* server for the Production deployment share. The database connection will be via TCP/IP.

PowerShell

```
Add-PSSnapIn Microsoft.BDD.PSSnapIn
New-PSDrive -Name "DS002" -PSProvider MDTProvider -Root "D:\Production
Deployment Share"
New-MDTDatabase -path "DS002:" -SQLServer "DeploymentServer" -Netlib
"DBMSSOCN" -Database "MDT2010" -SQLShare "DB_Connect" -Force -Verbose
```

Creating a Selection Profile

The following Windows PowerShell commands create a new Applications selection profile.

PowerShell

```
Add-PSSnapIn Microsoft.BDD.PSSnapIn
New-PSDrive -Name "DS002" -PSProvider MDTProvider -Root "D:\Production
Deployment Share"
New-item -path "DS002:\Selection Profiles" -enable "True" -Name
"Applications" -Comments "" -Definition "<SelectionProfile><Include
path="Applications" /></SelectionProfile>" -ReadOnly "False" -Verbose
```

Updating a Deployment Share

The following Windows PowerShell commands update the Production deployment share, which is in D:\Production Deployment Share.

PowerShell

```
Add-PSSnapIn Microsoft.BDD.PSSnapIn
New-PSDrive -Name "DS002" -PSProvider MDTProvider -Root "D:\Production
Deployment Share"
Update\-MDTDeploymentShare \-path "DS002:" \-Verbose
```

Creating a Linked Deployment Share

The following Windows PowerShell commands create a deployment share that is linked to the Production deployment share and resides under the *remote_server_name*\Deployment$ share. The Everything selection profile is used to determine which content is replicated to the linked deployment share. Content from the Production deployment share will be merged with content that already exists in the *remote_server_name*\Deployment$ share.

PowerShell

```
Add-PSSnapIn Microsoft.BDD.PSSnapIn
New-PSDrive -Name "DS002" -PSProvider MDTProvider -Root "D:\Production
Deployment Share"
New-item -path "DS002:\Linked Deployment Shares" -enable "True" -Name
"LINKED001" -Comments "" -Root "\\RemoteServerName\Deployment$" -
SelectionProfile "Everything" -Replace "False" -Verbose
```

Updating a Linked Deployment Share

The following Windows PowerShell commands update the LINKED001 deployment share.

PowerShell

```
Add-PSSnapIn Microsoft.BDD.PSSnapIn
New-PSDrive -Name "DS002" -PSProvider MDTProvider -Root "D:\Production
Deployment Share"
Replicate-MDTContent -path "DS002:\Linked Deployment Shares\LINKED001" -
Verbose
```

Deleting a Linked Deployment Share

The following Windows PowerShell commands delete the LINKED001 deployment share.

PowerShell

```
Add-PSSnapIn Microsoft.BDD.PSSnapIn
Remove-item -path "DS002:\Linked Deployment Shares\LINKED001" -Verbose
```

Creating Media

The following Windows PowerShell commands create a source folder that contains content used to create bootable media. The Production deployment share will be used as the source. The Everything selection profile determines what content is placed in the media content folder. The LiteTouchMedia.iso file will be created when the media is generated. The media will support both x86 and x64 platforms.

PowerShell

```
Add-PSSnapIn Microsoft.BDD.PSSnapIn
New-PSDrive -Name "DS002" -PSProvider MDTProvider -Root "D:\Production
Deployment Share"
New-item -path "DS002:\Media" -enable "True" -Name "MEDIA001" -Comments
"some comment here" -Root "D:\Media" -SelectionProfile "Everything" -
SupportX86 "True" -SupportX64 "True" -GenerateISO "True" -ISOName
"LiteTouchMedia.iso" -Verbose
New-PSDrive -Name "MEDIA001" -PSProvider "MDTProvider" -Root
"D:\Media\Content" -Description "Embedded media deployment share" -Force -
Verbose
```

Generating Media

The following Windows PowerShell commands create the LiteTouchMedia.iso file in D:\Media, which will use content from the MEDIA001 media source folder.

PowerShell

```
Add-PSSnapIn Microsoft.BDD.PSSnapIn
New-PSDrive -Name "DS002" -PSProvider MDTProvider -Root "D:\Production
Deployment Share"
Generate-MDTMedia -path "DS002:\Media\MEDIA001" -Verbose
```

Deleting Media

The following Windows PowerShell command deletes the MEDIA001 media from the Production deployment share.

PowerShell

```
Remove-item -path "DS002:\Media\MEDIA001" -Verbose
```

Delaying Domain Join to Avoid Application of Group Policy Objects

Group Policy is a rich and flexible technology providing the capability to efficiently manage a large number of Active Directory Domain Services (AD DS) computer and user objects through a centralized, one-to-many model. Group Policy settings are contained in a Group Policy object (GPO) and linked to one or more AD DS service containers—sites, domains, and organizational units (OUs).

Some organizations have Group Policy settings that are restrictive and could cause problems during operating system deployments. For example, the following Group Policy settings can interrupt an automated logon process:

- Autologon restrictions
- Administrator account renaming
- Legal banners and captions
- Restrictive security policies (for example, the Specialized Security - Limited Functionality [SSLF] policy)

  One option to overcome the issues that a GPO might cause during deployment is to join the computer to the domain as late as possible in the deployment process. This join can be done using a custom task sequence step that runs the ZTIDomainJoin.wsf script.

To join the target computer to the domain, the ZTIDomainJoin.wsf script uses the **DomainAdmin**, **DomainAdminDomain**, **DomainAdminPassword**, **JoinDomain**, and **MachineObjectOU** properties. You can declare these properties using the Windows Deployment Wizard, deployment share rules, the MDT DB, and Configuration Manager computer and collection rules. The account used must have the rights required to create and delete computer objects in the domain.

Typically, the ZTIConfigure.wsf script updates the Unattend.xml or Unattend.txt file with the values that these properties specify. These settings are then parsed by the Windows Setup program, and the system attempts to join to the domain early in the deployment process. Doing so subjects the target computer to settings specified in domain GPOs and can possibly cause the deployment process to fail.

To intentionally delay joining the target computer to the domain during the deployment process, you can remove certain elements from the Unattend.xml file. The ZTIConfigure.wsf script will skip over writing properties to the Unattend.xml file if the associated property element is missing from the file.

> ⓘ **Note**
>
> This sample work-around is only valid when deploying the Windows 7, Windows Server 2008, or Windows Server 2008 R2 operating systems.

Prepare the unattend.xml file so the target computer does not attempt to join the domain during Windows Setup

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/Task Sequences/*task_sequence* (where *deployment_share* is the name of the deployment share and *task_sequence* is the name of the task sequence to be configured).
3. In the Actions pane, select **Properties**.
4. On the **OS Info** tab, select **Edit Unattend.xml**.

   The Windows System Image Manager (Windows SIM) starts.
5. In the **Answer File** pane, go to **4 specialize/Identification/Credentials**. Right-click **Credentials**, and then select **Delete**.
6. Select **Yes**.

7. Save the answer file, and then exit Windows SIM.
8. Select **OK** on the task sequence **Properties** dialog box.

   With the `Credentials` elements missing from the unattend.xml file, the ZTIConfigure.wsf script is not able to populate the domain join information in the Unattend.xml file, which will prevent Windows Setup from attempting to join the domain.

   To add a task sequence step that joins the target computer to the domain

9. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
10. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/*deployment_share*/Task Sequences/*task_sequence* (where *deployment_share* is the name of the deployment share and *task_sequence* is the name of the task sequence to be configured).
11. In the Actions pane, select **Properties**.
12. On the **Task Sequence** tab, go to and expand the State Restore node.
13. Verify that the **Recover From Domain** task sequence step is present. If yes, proceed to step 9.
14. In the task sequence **Properties** dialog box, select **Add**, go to **Settings**, and select **Recover From Domain**.
15. Add the **Recover From Domain** task sequence step to the task sequence editor. Verify that the step is in the desired location in the task sequence.
16. Verify that the settings for the **Recover From Domain** task sequence step are configured to meet your needs.
17. Select **OK** on the task sequence **Properties** dialog box to save the task sequence.

Feedback

Was this page helpful? Yes No

Provide product feedback

Quick Start Guide for Lite Touch Installation

Article • 02/12/2024

Microsoft Deployment Toolkit (MDT) 2013 provides technology for deploying Windows operating systems, and Microsoft Office. This guide helps you quickly evaluate MDT 2013 by providing condensed, step-by-step instructions for using it to install the Windows 8.1 operating system through Lite Touch Installation (LTI) using bootable media (DVD or USB flash drive). This guide demonstrates how to perform the New Computer deployment scenario using an MDT 2013 deployment share. The New Computer deployment scenario covers the deployment of Windows 8.1 to a new computer. This scenario assumes that there is no user data or profile to preserve.

> **Note**
>
> In this document, *Windows* applies to the Windows 8.1, Windows 8, Windows 7, Windows Server® 2012 R2, Windows Server 2012, and Windows Server 2008 R2 operating systems unless otherwise noted. MDT does not support ARM processor-based versions of Windows. Similarly, *MDT* refers to MDT 2013 unless otherwise stated.

After using this guide to evaluate MDT, review the rest of the MDT guidance to learn more about the technology's advanced features.

Prerequisites

To deploy operating systems and applications using MDT, the environment must meet the following software and computer configuration prerequisites.

Required Software

To complete this guide, the following software is required:

- Windows 8.1

  If you decide to complete this guide on an operating system other than Windows 8.1, MDT requires the following elements:

  - Microsoft .NET Framework version 3.5 with Service Pack 1

    - Windows PowerShell™ version 2.0

      Windows 8.1 includes these features.

- Windows Assessment and Deployment Kit (Windows ADK) for Windows 8.1
- Networking services, including Domain Name System and Dynamic Host Configuration Protocol

> **Note**
>
> The Task Sequencer used in MDT deployments requires that the Create Global Object right be assigned to credentials used to access and run the Deployment Workbench and the deployment process. This right is normally available to accounts with Administrator-level permissions (unless explicitly removed). Also, the Specialized Security - Limited Functionality (SSLF) security profile removes the Create Global Object right and should not be applied to computers being deployed using MDT until the MDT process is complete.

Computer Configuration

To complete this guide, set up the computers listed in the following table. These computers can be either physical computers or virtual machines (VMs) with the system resources designated.

Expand table

Computer name	Description
WDG-MDT-01	This computer runs MDT and Windows 8.1 and is installed in a domain named *mdt2013.corp.woodgrovebank.com* with a network basic input/output system (NetBIOS) name of *MDT2013*. The system resources of the computer are: - Processor running at 1.4 gigahertz (GHz) or faster. - 1 gigabyte (GB) or greater physical memory. - One disk partition that has 16 GB or more available disk space that will become the drive C partition. - One CD-ROM or DVD-ROM drive that will be assigned the drive letter D.
WDG-REF-01	This is the reference computer and runs no current operating system. The system resources of the computer are: - Processor running at 1.4 GHz or faster.

Computer name	Description
	- 1 GB or more of physical memory. - 16 GB or more of available disk space.
WDG-CLI-01	This is the target computer and runs no current operating system. The system resources of the computer are: - Processor running at 1.4 GHz or faster. - 1 GB or more of physical memory. - 16 GB or more of available disk space.

> **Note**
>
> This guide assumes that you are evaluating MDT on 64-bit (x64) physical or virtual computers. If evaluating MDT on 32-bit (x86) platforms, download and install the x86 editions of MDT and the components that this guide describes.

Step 1: Obtain the Required Software

This guide assumes that the 64-bit version of Windows 8.1 is installed on a computer named *WDG-MDT-01*. If the computer you are using has a different name, substitute the name of that computer for *WDG-MDT-01*.

> **Note**
>
> This section assumes that you are creating a new infrastructure for MDT.

The following software is required to perform LTI deployments:

- MDT 2013
- Windows ADK for Windows 8.1
- Windows 8.1 distribution files
- Device drivers required for the target computer, WDG-CLI-01
- Device drivers required for the reference computer, WDG-REF-01

Step 2: Prepare the MDT Environment

In this step, you prepare the MDT environment prior to creating the reference computer and deploying a captured image of the reference computer to the target computer (WDG-CLI-01).

Prepare the MDT environment by:

- Installing MDT as described in Step 2-1: Install MDT
- Installing Windows ADK as described in Step 2-2: Install Windows ADK

Step 2-1: Install MDT

To install MDT, complete the following steps:

1. Double-click **MicrosoftDeploymentToolkit2013_x64.msi** (for 64-bit operating systems) or **MicrosoftDeploymentToolkit2013_x86.msi** (for 32-bit operating systems), and then select **Install**.

   The Microsoft Deployment Toolkit 2013 Setup Wizard starts.

2. Complete the Microsoft Deployment Toolkit 2013 Setup Wizard using the following information. Accept the default values unless otherwise specified.

   Expand table

On this wizard page	Do this
Welcome to the Microsoft Deployment Toolkit 2013 Setup Wizard	Select **Next**.
End-User License Agreement	Select **I accept the terms in the License Agreement**, and then select **Next**.
Custom Setup	Select **Next**.
Ready to install Microsoft Deployment Toolkit 2013	Select **Install**.
Installing Microsoft Deployment Toolkit 2013	The progress for installing MDT is displayed.
Completing the Microsoft Deployment Toolkit 2013 Setup Wizard	Select **Finish**.

   The Microsoft Deployment Toolkit 2013 Setup Wizard finishes, and MDT is installed on WDG-MDT-01.

Step 2-2: Install Windows ADK

To install Windows ADK, perform the following steps:

1. Mount the Windows ADK distribution files on a physical or virtual CD-ROM drive.
2. In Windows Explorer, go to the root of the CD-ROM drive, and then double-click **adksetup.exe**.

   The Assessment and Deployment Kit Setup Wizard starts.
3. Complete the Assessment and Deployment Kit Setup Wizard using the following information.

Expand table

On this wizard page	Do this
Specify Location	Select **Next**.
Join the Customer Experience Improvement Program (CEIP)	Select **Yes** if you want to participate or **No** if not. Then, select **Next**.
License Agreement	Select **Accept**.
Select the features you want to install	Ensure that only the check boxes for the following features are selected, and then select **Next**: - Deployment Tools - Windows Preinstallation Environment (Windows PE) - Windows User State Migration Tool **Note:** MDT does not require the other features, but they can be installed, if desired.
Installing features	The progress for installing the features is displayed.
Welcome to the Assessment and Deployment Kit	Select **Close**.

4. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
5. Close all open windows.

> **Note**
>
> After installing Windows ADK, log off, and then log on again to the computer so that the PATH environment variable is updated to include the %Program

Files%\Windows Imaging folder.

Step 3: Configure MDT to Create the Reference Computer

When you have prepared the MDT environment, create the reference computer. The reference computer is the template for deploying new images to the target computers. Configure this computer exactly as the target computers will be configured. You will deploy Windows 8.1 to the reference computer (WDG-REF-01), capture an image of the reference computer, and then deploy the captured image to the target computer (WDG-CLI-01).

Configure MDT to create a reference computer by:

- Creating an MDT deployment share as described in Step 3-1: Create an MDT Deployment Share
- Adding operating system files to the deployment share as described in Step 3-2: Add Operating System Files to the Deployment Share
- Adding device drivers to the deployment share as described in Step 3-3: Add Device Drivers to the Deployment Share
- Creating a task sequence for the reference computer as described in Step 3-4: Create a Task Sequence for the Reference Computer
- Enabling monitoring of the LTI deployment process as described in Step 3-5: Enable LTI Deployment Process Monitoring
- Updating the deployment share as described in Step 3-6: Update the Deployment Share

Step 3-1: Create an MDT Deployment Share

Before deployment can begin, create an MDT deployment share in the Deployment Workbench. This deployment share is the repository for the operating system images, language packs, applications, device drivers, and other software deployed to the target computers.

To create a deployment share in the Deployment Workbench

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares.
3. In the Actions pane, select **New Deployment Shares**.

   The New Deployment Share Wizard starts.
4. Complete the New Deployment Share Wizard using the following information.

Expand table

On this wizard page	Do this
Path	In **Deployment share path**, type **C:\DeploymentShare$**, and then select **Next**.
Share	Select **Next**.
Descriptive Name	Select **Next**.
Options	Select **Next**.
Summary	Select **Next**.
Progress	The progress for creating the deployment share is displayed.
Confirmation	Select **Finish**.

The New Deployment Share Wizard finishes, and the new deployment share—MDT Deployment Share (C:\DeploymentShare$)—appears in the details pane.

Step 3-2: Add Operating System Files to the Deployment Share

MDT acts as a repository for the operating system files deployed to the reference computer (WDG-REF-01) and target computer (WDG-CLI-01). Add the operating system in the Operating Systems node in the Deployment Workbench using the Import Operating System Wizard.

To add the Windows 8.1 operating system files to the deployment share

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.

2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/MDT Deployment Share (C:\DeploymentShare$)/Operating Systems.

3. In the Actions pane, select **Import Operating System**.

   The Import Operating System Wizard starts.

4. Complete the Import Operating System Wizard using the following information.

Expand table

On this wizard page	Do this
On this wizard page	**Do this**
OS Type	Select **Full set of source files**, and then select **Next**.
Source	In **Source directory**, type ***source_path*** (where *source_path* is the fully qualified path to the Windows 8.1 distribution files), and then select **Next**.
Destination	Select **Next**.
Summary	Select **Next**.
Progress	The progress for importing the operating system is displayed.
Confirmation	Select **Finish**.

The Import Operating System Wizard finishes. Windows 8.1 is added to the list of operating systems in the details pane and copied to the *deployment_share*\Operating Systems*operating_system* folder (where *deployment_share* is the shared network folder you created earlier in the process and *operating_system* is the name of the operating system you added to the deployment share).

Step 3-3: Add Device Drivers to the Deployment Share

After you have added Windows 8.1 to the Deployment Workbench, add any device drivers required for the reference computer (WDG-REF-01) and the target computer (WDG-CLI-01). These device drivers will be added to Windows PE and deployed with Windows 8.1. Add the device drivers in the Out-of-box Drivers node in the Deployment Workbench by using the New Driver Wizard, which copies the device driver files to the

deployment share in Out-of-Box Drivers*device_driver* (where *device_driver* is the name of the device driver you added to the deployment share).

> **Note**
>
> If the device drivers for the reference computer (WDG-REF-01) and the target computer (WDG-CLI-01) are included with Windows 8.1, skip this step and proceed with the following step.

To add the device drivers for the reference and target computers to the distribution share

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/MDT Deployment Share (C:\DeploymentShare$)/Out-of-Box Drivers.
3. In the Actions pane, select **Import Drivers**.

   The Import Driver Wizard starts.
4. Complete the Import Driver Wizard using the following information.

Expand table

On this wizard page	Do this
Specify Directory	In **Driver source directory**, type ***driver_path*** (where *driver_path* is the fully qualified path to the folder containing the device drivers), and then select **Next**.
Summary	Select **Next**.
Progress	The progress for importing the device drivers is displayed.
Confirmation	Select **Finish**.

The Import Driver Wizard finishes. The device drivers are added to the list of operating systems in the details pane and are copied to the *deployment_share*\Out-of-box Drivers folder (where *deployment_share* is the deployment share you created earlier in the process).

Step 3-4: Create a Task Sequence for the Reference Computer

Create MDT task sequences in the Task Sequences node in the Deployment Workbench using the New Task Sequence Wizard. MDT includes the Standard Client Task Sequence template, which you can use to deploy the target operating system to the reference computer (WDG-REF-01).

To create a task sequence for deploying the reference computer

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/MDT Deployment Share (C:\DeploymentShare$)/Task Sequences
3. In the Actions pane, select **New Task Sequence**.

   The New Task Sequence Wizard starts.
4. Complete the New Task Sequence Wizard using the following information. Accept the default values unless otherwise specified.

Expand table

On this wizard page	Do this
General Settings	1. In **Task sequence ID**, type **WIN8_REFERENCE**. 2. In **Task sequence name**, type **Deploy Windows 8.1 to Reference Computer**. 3. In **Task sequence comments**, type **Task sequence for deploying Windows 8.1 to the reference computer (WDG-REF-01)**. 4. Select **Next**.
Select Template	In **The following task sequence templates are available. Select the one you would like to use as a starting point**, select **Standard Client Task Sequence**, and then select **Next**.
Select OS	In **The following operating system images are available to be deployed with this task sequence**. **Select one to use**, select **Windows 8.1 *edition*** (where *edition* is the edition of Windows 8.1 added to the Operating Systems node in the Deployment Workbench), and then select **Next**.
Specify Product Key	Select **Do not specify a product key at this time**, and then select **Next**.

On this wizard page	Do this
OS Settings	1. In **Full Name**, type **Woodgrove Bank Employee**. 2. In **Organization**, type **Woodgrove Bank**. 3. In **Internet Explorer Home Page**, type **http://www.woodgrovebank.com**. 4. Select **Next**.
Admin Password	In **Administrator Password** and **Please confirm Administrator Password**, type **P@ssw0rd**, and then select **Next**.
Summary	Select **Next**.
Progress	The progress for creating the task sequence is displayed.
Confirmation	Select **Finish**.

The Import Task Sequence Wizard finishes, and the **Deploy Windows 8.1 to Reference Computer** task sequence is added to the list of task sequences.

Step 3-5: Enable LTI Deployment Process Monitoring

Prior to deploying the reference computer (WDG-REF-01) with the LTI bootable media you created earlier in the process, enable monitoring of the LTI deployment process. You monitor the LTI deployment process in the Monitoring node in the deployment share. You enable monitoring on the **Monitoring** tab on the deployment share properties sheet. Later in the process, you will monitor the LTI deployment process.

To enable monitoring of the LTI deployment process

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares.
3. In the details pane, select **MDT Deployment Share (C:\DeploymentShare$)**.
4. In the Actions pane, select **Properties**

   The **MDT Deployment Share (C:\DeploymentShare$) Properties** dialog box opens.

5. In the **MDT Deployment Share (C:\DeploymentShare$) Properties** dialog box, on the **Monitoring** tab, select the **Enable monitoring for this deployment share** check box, and then select **Apply**.

6. In the **MDT Deployment Share (C:\DeploymentShare$) Properties** dialog box, on the **Rules** tab, notice that the **EventService** property has been added to the CustomSettings.ini file, and then select **OK**.

7. Close all open windows and dialog boxes.

Step 3-6: Update the Deployment Share

After configuring the deployment share, update it. Updating the deployment share updates all the MDT configuration files and generates a customized version of Windows PE. You use the customized version of Windows PE to start the reference computer and initiate LTI deployment.

To update the deployment share in the Deployment Workbench

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.

2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares.

3. In the details pane, select **MDT Deployment Share (C:\DeploymentShare$)**.

4. In the Actions pane, select **Update Deployment Share**.

   The Update Deployment Share Wizard starts.

5. Complete the Update Deployment Share Wizard using the following information. Accept the default values unless otherwise specified.

Expand table

On this wizard page	Do this
Options	Select **Next**.
Summary	Select **Next**.
Progress	The progress for updating the deployment share is displayed.
Confirmation	Select **Finish**.

The Deployment Workbench starts updating the MDT Deployment Share (C:\DeploymentShare$) deployment share. The Deployment Workbench also creates the LiteTouchPE_x64.iso and LiteTouchPE_x64.wim files (for 64-bit target computers) or LiteTouchPE_x86.iso and LiteTouchPE_x86.wim files (for 32-bit target computers) in the *deployment_share*\Boot folder (where *deployment_share* is the network shared folder used as the deployment share).

Step 4: Deploy Windows 8.1 and Capture an Image of the Reference Computer

After creating the task sequence to deploy Windows 8.1 to the reference computer, initiate operating system deployment and image capture by starting the reference computer with the LTI bootable media.

Deploy Windows 8.1 and capture an image of the reference computer by:

- Creating the LTI bootable media as described in Step 4-1: Create the LTI Bootable Media
- Starting the reference computer with the LTI bootable media and monitoring the LTI deployment process as described in Step 4-2: Start the Reference Computer with the LTI Bootable Media

Step 4-1: Create the LTI Bootable Media

You need to provide a method for starting the computer with the customized version of Windows PE you created when you updated the deployment share. The Deployment Workbench creates the LiteTouchPE_x64.iso and LiteTouchPE_x64.wim files (for 64-bit target computers) or LiteTouchPE_x86.iso and LiteTouchPE_x86.wim files (for 32-bit target computers) in the *deployment_share*\Boot folder (where *deployment_share* is the network shared folder used as the deployment share). Create the appropriate LTI bootable media from one of these images.

To create the LTI bootable media

1. In Windows Explorer, go to C:\DeploymentShare$\Boot.
2. Based on the type of computer used for the reference computer (WDG-REF-01), perform one of the following tasks:
    - If the reference computer is a physical computer, create a physical CD or DVD of the LiteTouchPE_x64.iso or LiteTouchPE_x86.iso file.

- If the reference computer is a VM, start the VM directly from the LiteTouchPE_x64.iso or LiteTouchPE_x86.iso file or from a CD or DVD of the International Standard Organization (ISO) files.

Step 4-2: Start the Reference Computer with the LTI Bootable Media

Start the reference computer (WDG-REF-01) with the LTI bootable media you created earlier in the process. The LTI bootable media starts Windows PE on the reference computer and initiates deployment. At the end of the MDT deployment process, Windows 8.1 is deployed on the reference computer.

> ⓘ **Note**
>
> You can use a 32-bit boot image to deploy both 32-bit and 64-bit operating systems; however, a 64-bit boot image can only be used to deploy 64-bit operating systems.

You could also initiate the process by starting the target computer from Windows Deployment Services. For more information, see the section, "Preparing Windows Deployment Services", in the MDT document, *Using the Microsoft Deployment Toolkit*.

To start the reference computer with the LTI bootable media

1. Start WDG-REF-01 with the LTI bootable media you created earlier in the process.

   Windows PE starts, and then the Windows Deployment Wizard starts.

2. Complete the Windows Deployment Wizard using the following information. Accept the default values unless otherwise specified.

Expand table

On this wizard page	Do this
Welcome	Select **Run the Deployment Wizard to install a new Operating System**.
Credentials	1. In **User Name**, type **Administrator**. 2. In **Password**, type **P@ssw0rd**. 3. In **Domain**, type **MDT2013**. 4. Select **OK**.

On this wizard page	Do this
Task Sequence	Select **Deploy Windows 8.1 to Reference Computer**, and then select **Next**.
Computer Details	In **Computer name**, **type WDG-REF-01**, and then select **Next**.
Move Data and Settings	Select **Next**.
User Data (Restore)	Select **Next**.
Locale and Time	Select **Next**.
Capture Image	Select **Capture an image of this reference computer**, and then select **Next**.
Ready	1. Select **Details** to view the information provided in the wizard. 2. Select **Begin**.

To monitor the reference computer deployment process, complete the following steps on WDG-MDT-01

3. On WDG-MDT-01, select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.

4. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/MDT Deployment Share (C:\DeploymentShare$)/Monitoring.

5. In the details pane, view the deployment process for **WDG-REF-01**.

6. In the Actions pane, periodically select **Refresh**.

   The status of the deployment process is updated in the details pane.

   Continue to monitor the deployment process until the process is complete.

7. In the details pane, select **WDG-REF-01**.

8. In the Actions pane, select **Properties**.

   The **WDG-REF-01 Properties** dialog box is displayed.

9. In the **WDG-REF-01 Properties** dialog box, on the **Identity** tab, view the monitoring information provided about the deployment process as described in the following table.

Expand table

Information	Description
ID	Unique identifier for the computer being deployed.
Computer Name	The name of the computer being deployed.
Deployment status	The current status of the computer being deployed; the status can be one of the following: - **Running**. The task sequence is healthy and running. - **Failed**. The task sequence failed, and the deployment process was unsuccessful. - **Completed**. The task sequence has finished. - **Unresponsive**. The task sequence has not updated its status in the past four hours and is assumed to be nonresponsive.
Step	The current task sequence step being run.
Progress	The overall progress of the task sequence. The progress bar indicates how many task sequence steps have been run out of the total number of task sequence steps.
Start	The time the deployment process started.
End	The time the deployment process ended.
Elapsed	The length of time the deployment process has been running or took to run if the deployment process has finished.
Errors	The number of errors encountered during the deployment process.
Warnings	The number of warnings encountered during the deployment process.
Remote Desktop	This button allows you to establish a remote desktop connection with the computer being deployed using the Windows Remote Desktop feature. This method assumes that: - The target operating system is running and has remote desktop support enabled - **mstsc.exe** is in the path **Note:** This button is always visible but may not be able to establish a remote desktop session if the monitored computer is running Windows PE, has not completed installation of the target operating system, or does not have the Remote Desktop feature enabled.
VM Connection	This button allows you to establish a remote desktop connection to a VM running in HyperV®. This method assumes that:

Information	Description
	- The deployment is being performed to a VM running on Hyper-V - **vmconnect.exe** is located in the %ProgramFiles%\Hyper-V folder **Note:** This button appears when ZTIGather.wsf detects that Hyper-V integration components are running on the monitored computer. Otherwise, this button will not be visible.
DaRT Remote Control	This button allows you to establish a remote control session using the remote viewer feature in the Diagnostics and Recovery Toolkit (DaRT). This method assumes that: - DaRT has been deployed to the target computer and is currently running - **DartRemoteViewer.exe** is located in the %ProgramFiles%\Microsoft DaRT 7\v7 folder **Note:** This button appears when ZTIGather.wsf detects that DaRT is running on the monitored computer. Otherwise, this button will not be visible.
Automatically refresh this information every 10 seconds	Check box that controls whether the information in the dialog box is automatically refreshed. If the check box is: - Selected, the information is refreshed every 10 seconds - Cleared, the information is not automatically refreshed and must be manually refreshed using the **Refresh Now** button
Refresh Now	This button immediately refreshes the information displayed in the dialog box.

10. In the **WDG-REF-01 Properties** dialog box, select **OK**.

11. Close the Deployment Workbench.

    To complete the reference computer deployment process, perform the following steps on WDG-REF-01:

12. On WDG-REF-01, in the **Deployment Summary** dialog box, select **Details**.

    If any errors or warnings occur, review the errors or warnings, and record any diagnostic information.

13. In the **Deployment Summary** dialog box, select **Finish**.

    Windows 8.1 is now installed on the reference computer, and the captured Windows Imaging Format (WIM) file of the reference computer (WIN7_REFERENCE.wim) is stored in the *deployment_share*\Captures folder (where

deployment_share is the shared folder used as the deployment share).If errors or warnings occur, consult the MDT document *Troubleshooting Reference*.

Step 5: Configure MDT to Deploy Windows 8.1 to the Target Computer

When you have captured an image of the reference computer (MDT-REF-01), deploy it to the target computer (MDT-CLI-01). You import the captured image into the Deployment Workbench using the Import Operating System Wizard. Then, you create a task sequence to deploy the captured image to the target computer.

Configure MDT to deploy Windows 8.1 to the target computer by:

- Adding the captured image of the reference computer to the Deployment Workbench as described in Step 5-1: Add the Captured Image of the Reference Computer to the Deployment Workbench
- Creating a task sequence for the target computer as described in Step 5-2: Create a Task Sequence for the Target Computer

Step 5-1: Add the Captured Image of the Reference Computer to the Deployment Workbench

To deploy the captured image of the reference computer to the target computer, add the captured image to the list of operating systems in the Operating Systems node in the Deployment Workbench. The Import Operating System Wizard copies the operating system files to the *deployment_share*\Operating Systems*operating_system* folder (where *deployment_share* is the deployment share folder created earlier in the process and *operating_system* is the name of the operating system added to the deployment share).

To add the captured image of the reference computer to the Deployment Workbench

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/MDT Deployment Share (C:\DeploymentShare$)/Operating Systems.
3. In the Actions pane, select **Import Operating system**.

   The Import Operating System Wizard starts.

4. Complete the Import Operating System Wizard using the information in the following table.

Expand table

On this wizard page	Do this
OS Type	Select **Custom image file**, and then select **Next**.
Image	In **Source file**, type **C:\DeploymentShare$\Captures\WIN8_REFERENCE.wim**, and then select **Next**.
Setup	Select **Next**.
Destination	Select **Next**.
Summary	Select **Next**.
Progress	The progress for importing the operating system is displayed.
Confirmation	Select **Finish**.

The Import Operating System Wizard finishes. The captured image of the reference computer (WDG-REF-01) operating system is added to the list of operating systems in the details pane and is copied to the *deployment_share*\Operating Systems*operating_system* folder (where *deployment_share* is the deployment share folder created earlier in the process and *operating_system* is the name of the operating system added to the deployment share).

5. Close all open windows and dialog boxes.

Step 5-2: Create a Task Sequence for the Target Computer

Create an MDT task sequence for the target computer in the Task Sequences node in the Deployment Workbench using the New Task Sequence Wizard. This task sequence is used to deploy the captured image of the reference computer to the target computer.

To create a task sequence for deploying the captured image to the target computer

1. Select **Start**, and then point to **All Programs**. Point **to Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.

2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/MDT Deployment Share

(C:DeploymentShare$)/Task Sequences.

3. In the Actions pane, select **New Task Sequence**.

   The New Task Sequence Wizard starts.

4. Complete the New Task Sequence Wizard using the following information. Accept the default values unless otherwise specified.

Expand table

On this wizard page	Do this
General Settings	1. In **Task sequence ID**, type **WIN8_TARGET**. 2. In **Task sequence name**, type **Deploy Captured Image to Target Computer**. 3. In **Task sequence comments**, type **Task sequence for captured Windows 8.1 image from reference computer (WDG-REF-01) to the target computer (WDG-CLI-01)**. 4. Select **Next**.
Select Template	In **The following task sequence templates are available. Select the one you would like to use as a starting point**, select Standard **Client Task Sequence**, and then select **Next**.
Select OS	In **The following operating system images are available to be deployed with this task sequence. Select one to use**, select **WIN8_RERENCEDrive in "WIN8_REFERENCE\WIN8_REFERENCE.wim"**, and then select **Next**.
Specify Product Key	Select **Do not specify a product key at this time**, and then select **Next**.
OS Settings	1. In **Full Name**, type **Woodgrove Bank Employee**. 2. In **Organization**, type **Woodgrove Bank**. 3. In **Internet Explorer Home Page**, type **http://www.woodgrovebank.com**. 4. Select **Next**.
Admin Password	In **Administrator Password** and **Please confirm Administrator Password**, type **P@ssw0rd**, and then select **Next**.
Summary	Select **Next**.
Progress	The progress for creating the task sequence is displayed.
Confirmation	Select **Finish**.

The Import Task Sequence Wizard finishes, and the **WIN8_TARGET** task sequence is added to the list of task sequences.

5. Close all open windows and dialog boxes.

Step 6: Deploy the Captured Image of the Reference Computer to the Target Computer

When you have captured an image of the reference computer and created and configured the appropriate task sequence, deploy the captured image. Configure MDT to provide all the necessary configuration settings for deployment to the target computer. After initiating the deployment process, the image of the reference computer running Windows 8.1 is automatically deployed to the target computer and configured with the settings defined.

Deploy the captured image of the reference computer to the target computer by:

- Starting the target computer with the LTI bootable media and monitor the LTI deployment process as described in Step 6-1: Start the Target Computer with the LTI Bootable Media

Step 6-1: Start the Target Computer with the LTI Bootable Media

Start the target computer (WDG-CLI-01) with the LTI bootable media you created earlier in the process. This CD starts Windows PE on the target computer and initiates deployment. At the end of the process, Windows 8.1 is deployed on the target computer.

> **Note**
>
> You can also initiate deployment by starting the target computer from Windows Deployment Services. For more information, see the MDT document *Using the Microsoft Deployment Toolkit*.

To start the target computer with the LTI bootable media

1. Start WDG-CLI-01 with the LTI bootable media you created earlier in the process.

   Windows PE starts, and then the Windows Deployment Wizard starts.

2. Complete the Windows Deployment Wizard using the following information. Accept the default values unless otherwise specified.

Expand table

On this wizard page	Do this
Welcome	Select **Run the Deployment Wizard to install a new Operating System**.
Credentials	1. In **User Name**, type **Administrator**. 2. In **Password**, type **P@ssw0rd**. 3. In **Domain**, type **MDT2013**. 4. Select **OK**.
Task Sequence	Select **Deploy Captured Image to Target Computer**, and then select **Next**.
Computer Details	In **Computer name**, type **WDG-CLI-01**, and then select **Next**.
Move Data and Settings	Select **Next**.
User Data (Restore)	Select **Next**.
Locale and Time	**Select Next**.
Capture Image	Select **Next**.
BitLocker	Select **Next**.
Ready	1. Select **Details** to view the information provided in the wizard. 2. Select **Begin**.

The wizard starts, and then the operating system deployment starts.

To monitor the target computer deployment process, complete the following steps on WDG-MDT-01

3. On WDG-MDT-01, select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
4. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/MDT Deployment Share (C:\DeploymentShare$)/Monitoring
5. In the Actions pane, periodically select **Refresh**.
6. In the details pane, view the deployment process for **WDG-CLI-01**.
7. In the Actions pane, periodically select **Refresh**.

The status of the deployment process is updated in the details pane. Continue to monitor the deployment process until the process is complete.

8. Close the Deployment Workbench.

   To complete the target computer deployment process, perform the following steps on WDG-CLI-01

9. On WDG-CLI-01, in the **Deployment Summary** dialog box, select **Details**.

   If any errors or warnings occur, review the errors or warnings, and record any diagnostic information.

10. In the **Deployment Summary** dialog box, select **Finish**.

    At the end of the MDT deployment process, the **Deployment Summary** dialog box appears. The image of Windows 8.1 captured from the reference computer is now installed on the target computer. If any errors or warnings occur, consult the MDT document *Troubleshooting Reference*.

Feedback

Was this page helpful? Yes No

Provide product feedback

Quick Start Guide for Microsoft System Center 2012 R2 Configuration Manager

Article • 02/12/2024

Microsoft Deployment Toolkit (MDT) 2013 provides technology for deploying Windows operating systems, and Microsoft Office. This quick start guide helps you quickly evaluate MDT 2013 by providing condensed, step-by-step instructions for using it to install the Windows 8.1 operating system with Microsoft System Center 2012 R2 Configuration Manager. This quick start guide demonstrates how to perform the New Computer deployment scenario, which covers the deployment of Windows 8.1 to a new computer. This scenario assumes that there is no user data or profile to preserve.

> **Note**
>
> In this document, *Windows* applies to the Windows 8.1, Windows 8, Windows 7, Windows Server® 2012 R2, Windows Server 2012, and Windows Server 2008 R2 operating systems unless otherwise noted. MDT does not support ARM processor-based versions of Windows. Similarly, *MDT* refers to MDT 2013 unless otherwise stated.

After using this guide to evaluate MDT, review the rest of the MDT guidance to learn more about the technology's advanced features.

Prerequisites

Zero Touch Installation installations using Configuration Manager have the following prerequisites.

Required Software

To complete this guide, the following software is required:

- Windows Server 2008 R2
- Microsoft SQL Server 2008 R2
- SQL Server 2008 R2 Service Pack 1 (SP1)
- SQL Server 2008 R2 SP1 Cumulative Update 6 (CU6)

- Windows 8.1
- System Center 2012 R2 Configuration Manager
- Microsoft .NET Framework version 3.5 with SP1
- Windows PowerShell version 2.0
- Windows Preinstallation Environment (Windows PE), which is included in Configuration Manager
- Networking services, including Domain Name System (DNS) and Dynamic Host Configuration Protocol (DHCP)
- Active Directory Domain Services (AD DS)

  See the Supported Configurations for Configuration Manager for additional software combinations that can be used for installing Configuration Manager.

> **Note**
>
> The Task Sequencer used in MDT deployments requires that the Create Global Object right be assigned to credentials used to access and run the Deployment Workbench and the deployment process. This right is normally available to accounts with Administrator-level permissions (unless explicitly removed). Also, the Specialized Security - Limited Functionality (SSLF) security profile removes the Create Global Object right and should not be applied to computers deployed using MDT.

Computer Configuration

To complete this guide, set up the computers listed in the following table. These computers can be either physical computers or virtual machines (VMs) with the system resources designated.

Expand table

Computer	Description and system resources
WDG-MDT-01	This computer runs the MDT infrastructure and Configuration Manager. The computer runs Windows Server 2008 R2 with the following networking services installed: - AD DS

Computer	Description and system resources
	- DNS Server - DHCP Server - Windows Deployment Services The system resources of the computer are as follows: - Quad-core processor running at 2.66 gigahertz (GHz) or faster - 4 gigabytes (GB) or more of physical memory - A disk partition that has 40 GB or more of available disk space; it will become the drive C partition - One CD-ROM or DVD-ROM drive that will be assigned the drive letter D - A disk partition that has 40 GB or more of available disk space; it will become partition E.
WDG-REF-01	This is the reference computer, which runs no current operating system. The system resources of the computer are as follows: - Processor running at 1.4 GHz or faster - 1 GB or more of physical memory - 16 GB or more of available disk space
WDG-CLI-01	This is the target computer, which runs no current operating system. The system resources of the computer are as follows: - Processor running at 1.4 GHz or faster - 1 GB or more of physical memory - 16 GB or more of available disk space

The resources listed in the preceding table reflect the system resources recommended to perform the steps in this guide. For information on the minimum system resource requirements for:

- Windows Server 2008 R2, see Installing Windows Server 2008 R2
- SQL Server 2008 R2, see Hardware and Software Requirements for Installing SQL Server 2008 R2

> **Note**
>
> This guide assumes that MDT is being evaluated on 64-bit (x64) physical or virtual computers. If evaluating MDT on 32-bit (x86) platforms, download and install the x86 editions of MDT and the components that this guide describes.

Step 1: Prepare the Prerequisite Infrastructure

For purposes of this guide, all the prerequisite infrastructure services run on the computer named *WDG-MDT-01*. Install the prerequisite software, server roles, and services on this computer before installing MDT.

> **Note**
>
> This section assumes that you are creating a new Configuration Manager infrastructure for MDT. If you are using an existing Configuration Manager infrastructure, review the steps in this section and substitute existing resource names for the resources created in this section (such as the computer name and shared network folders). After reviewing this section, proceed to **Step 2: Prepare the MDT Environment**.

Prepare the prerequisite infrastructure before installing MDT by:

- Installing Windows Server 2008 R2 as described in Step 1-1: Install Windows Server 2008 R2
- Creating the required folders and network shares as described in Step 1-2: Create the Required Folders and Network Shares
- Obtaining the software required to perform the steps in this guide as described in Step 1-3: Obtain the Required Software
- Installing the AD DS server role as described in Step 1-4: Install the AD DS Server Role
- Installing the DHCP Server server role as described in Step 1-5: Install the DHCP Server Server Role
- Installing the Web Services (IIS) server role as described in Step 1-6: Install the Web Services (IIS) Server Role
- Adding the required Windows Server 2008 R2 features as described in Step 1-7: Add the Required Windows Server 2008 R2 Features
- Creating the user and service accounts required to perform the steps in this guide as described in Step 1-8: Create the Required User and Service Accounts
- Installing SQL Server 2008 R2 for Configuration Manager to use as described in Step 1-9: Install SQL Server 2008 R2
- Adding the site server to the Administrators security group as described in Step 1-10: Add the Site Server to the Administrators Security Group

- Installing Configuration Manager as described in Step 1-11: Install Configuration Manager
- Configuring the network access account that Configuration Manager clients use to access Configuration Manager distribution points as described in Step 1-12: Configure the Network Access Account
- Configuring the Configuration Manager site boundaries and boundary groups as described in Step 1-13: Configure the Configuration Manager Site Boundaries and Boundary Groups
- Configuring the publishing of site information in AD DS and DNS as described in Step 1-14: Configure the Publishing of Site Information in AD DS and DNS

Step 1-1: Install Windows Server 2008 R2

Information for Installing Windows Server 2008 R2. Accept default values unless otherwise specified.

Expand table

When prompted for	Provide these values
Where do you want to install Windows?	**Disk 0 Unallocated Space**
Password	Any strong password
Computer name	**WDG-MDT-01**
Format for volumes C and E	**NTFS**
TCP/IP configuration	Configure with a static IP address configuration, with the other TCP/IP configuration options as appropriate for the environment

Step 1-2: Create the Required Folders and Network Shares

The MDT deployment process requires additional folders that are used as the source for files or to store files created during the MDT deployment process. Some of these folders need to be shared so that they can be accessed from other computers.

To create the required folders and share

1. The MDT Deployment Process Requires several folders. Create the following folders and shares with the specified permissions for each share.

Expand table

Create this folder	With this share name	With these share permissions
E:\Source$	Source$	**Administrators:** Co-owner **Everyone:** Read
E:\Images$	Images$	**Administrators:** Co-owner **Everyone:** Read
E:\Capture$	Capture$	**Administrators:** Co-owner **Everyone:** Read
E:\Packages$	Packages$	**Administrators:** Co-owner **Everyone:** Read

2. Create the following folders:
    - E:\CMDownloads
    - E:\Source$\CustomSettings
    - E:\Source$\Drivers
    - E:\Source$\Windows_8-1
    - E:\Source$\MDT_2013
    - E:\Source$\SQL2008R2
    - E:\Source$\SQL2008R2SP1
    - E:\Source$\SQL2008R2CU6
    - E:\Source$\ConfigMgr
    - E:\Packages$\Drivers
3. Copy the device drivers for the reference computer (WDG-REF-01) and the target computer (WDG-CLI-01) to E:\Source$\Drivers.

> **Note**
>
> The processes in this guide assume that the reference computer and target computer have the same devices and do not require different devices drivers.

Step 1-3: Obtain the Required Software

Besides Windows Server 2008 R2, Windows 8.1, and Configuration Manager, certain software is required to evaluate MDT based on the processes in this guide. The following table lists the software required to perform deployments using MDT, where to obtain the software, and where to place the software on WDG-MDT-01.

Expand table

Obtain this software	Place in this folder
MDT 2013	E:\Source$\MDT_2013
Windows 8.1 distribution files from the product media	E:\Source$\Windows_8-1
Device drivers required for the reference and target computers (WDG-REF-01 and WDG-CLI-01)	E:\Source$\Drivers
SQL Server 2008 R2 from the product media	E:\Source$\SQL2008R2
SQL Server 2008 R2 SP1, available at https://www.microsoft.com/download/details.aspx?id=26113	E:\Source$\SQL2008R2SP1
SQL Server 2008 R2 SP1 CU6, available at https://support.microsoft.com/kb/2679367	E:\Source$\SQL2008R2SP1CU6
Configuration Manager from the product media	E:\Source$\ConfigMgr

Step 1-4: Install the AD DS Server Role

AD DS is required to provide authentication and act as a repository for configuration values for the Microsoft products and technologies that MDT uses, such as SQL Server 2008 R2 and Configuration Manager.

To install AD DS, run the DCPROMO Wizard to configure the computer as a domain controller. Install AD DS using the following information, accepting any defaults unless otherwise specified.

Expand table

When prompted	Do this
For the domain type	Create a new domain in a new forest.
For the fully qualified domain name	Type **mdt2013.corp.woodgrovebank.com**.
For the forest functional level	Select **Windows Server 2008 R2**.
To install the DNS Server service as part of the domain controller installation process	Select **Yes**.

Step 1-5: Install the DHCP Server Server Role

The DHCP Server server role is required to provide automatic IP configuration for the target computers. Install DHCP Server using the following information, accepting any defaults unless otherwise specified.

> **Note**
>
> If you are using a virtualized environment, disable any DHCP configuration that the computer-virtualization software provides. Ensure that the DHCP Server service running WDG-MDT-01 is the only provider of IP configuration using DHCP.

Expand table

On this wizard page	Do this
Authorize DHCP server in Active Directory	Authorize WDG-MDT-01 to provide client IP configuration.
DHCP scopes	Create an appropriate scope that can be used to automatically configure TCP/IP for WDG-REF-01 and WDG-CLI-01.
DHCPv6 stateless mode configuration	Disable DHCPv6 stateless mode for this server.

Step 1-6: Install the Web Services (IIS) Server Role

Install the Web Services (IIS) server role with the role services listed in the following table. These role services are required for SQL Server 2008 R2 and Configuration Manager. Unless otherwise specified, use the default values.

Expand table

Role service	Status
Web Server	Installed
Common HTTP Features	Installed
Static Content	Installed
Default Document	Installed
Directory Browsing	Installed
HTTP Errors	Installed
HTTP Redirection	Installed
WebDAV Publishing	Installed
Application Development	Installed
ASP.NET	Installed
.NET Extensibility	Installed
ASP	Not installed
CGI	Not installed
ISAPI Extensions	Installed
ISAPI Filters	Installed
Server Side Includes	Not installed
Health and Diagnostics	Installed
HTTP Logging	Installed
Logging Tools	Installed
Request Monitor	Installed
Tracing	Installed
Custom Logging	Not installed
ODBC Logging	Not installed
Security	Installed
Basic Authentication	Not installed

Role service	Status
Windows Authentication	Installed
Digest Authentication	Not installed
Client Certificate Mapping Authentication	Not installed
IIS Client Certificate Mapping Authentication	Not installed
URL Authorization	Not installed
Request Filtering	Installed
IP and Domain Restriction	Not installed
Performance	Installed
Static Content Compression	Installed
Dynamic Content Compression	Not installed
Management Tools	Installed
IIS Management Console	Installed
IIS Management Scripts and Tools	Not installed
Management Service	Not installed
IIS 6 Management Compatibility	Installed
IIS 6 Metabase Compatibility	Installed
IIS 6 WMI Compatibility	Installed
IIS 6 Scripting Tools	Not installed
IIS 6 Management Console	Not installed
FTP Publishing Service	Not installed
FTP Server	Not installed
FTP Management Console	Not installed
IIS Hostable Web Core	Not installed

Step 1-7: Add the Required Windows Server 2008 R2 Features

In addition to installing the required Windows Server 2008 R2 server roles, add the following required features in Server Manager in the **Features Summary** section:

- Background Intelligent Transfer Service
- Remote Differential Compression

Step 1-8: Create the Required User and Service Accounts

Configuration Manager and SQL Server 2008 R2 require user accounts during the installation process. Use the following information for creating these accounts.

Expand table

Create this account	With these settings
SQL Server Agent service account	1. In **First name**, type **SQL Agent**. 2. In **Last name**, type **Service Account**. 3. In **User logon name**, type **SQLAgent**. 4. In **Password** and **Confirm password**, type **P@ssw0rd**. 5. Clear the **User must change password at next logon** check box. 6. Select the **Password never expires** check box. 7. Make the account a member of the Domain Admins security group. 8. In **Description**, type **Service account used to run SQL Server 2008 R2 Agent service**.
SQL Server Database Engine service account	1. In **First name**, type **SQL DB Engine**. 2. In **Last name**, type **Service Account**. 3. In **User logon name**, type **SQLDBEngine**. 4. In **Password** and **Confirm password**, type **P@ssw0rd**. 5. Clear the **User must change password at next logon** check box. 6. Select the **Password never expires** check box. 7. Make the account a member of the Domain Admins security group. 8. In **Description**, type **Service account used to run SQL Server 2008 R2 database engine**.
SQL Server Reporting Services service account	1. In **First name**, type **SQL Reporting**. 2. In **Last name**, type **Service Account**. 3. In **User logon name**, type **SQLReport**. 4. In **Password** and **Confirm password**, type **P@ssw0rd**. 5. Clear the **User must change password at next logon** check box. 6. Select the **Password never expires** check box. 7. Make the account a member of the Domain Admins

Create this account	With these settings
	security group. 8. In **Description**, type **Service account used to run SQL Server 2008 R2 reporting services**.
Configuration Manager Client Network Access account	1. In **First name**, type **CM 2012**. 2. In **Last name**, type **Client Network Access**. 3. In **User logon name**, type **CMNetAccess**. 4. In **Password** and **Confirm password**, type **P@ssw0rd**. 5. Clear the **User must change password at next logon** check box. 6. Select the **Password never expires** check box. 7. In **Description**, type **Service account used as the network access account for Configuration Manager Client**.

Step 1-9: Install SQL Server 2008 R2

Before installing Configuration Manager, install SQL Server 2008 R2 with SP1 and CU6 for SP1.

> **Note**
>
> To enable all SQL Server 2008 R2 features, install the Web Services (IIS) server role before installing SQL Server 2008 R2.

To install SQL Server 2008 R2

1. Start the SQL Server Installation Center.
2. In the SQL Server Installation Center, in the navigation pane, select **Installation**.
3. In the details pane, select **New installation or add features to an existing installation**.

   SQL Server 2008 R2 Setup Wizard starts.
4. Install SQL Server 2008 R2 using the following information, accepting the defaults unless otherwise specified.

Expand table

On this wizard page	Do this
Setup Support Rules	Select **OK**.
Product Key	Select **Next**.
License Terms	Select the **I accept the license terms** check box, and then select **Next**.
Setup Support Files	Select **Install**.
Setup Support Rules	Ensure that no critical results exist for the rules, and then select **Next**.
Setup Role	Select **SQL Server Feature Installation**, and select **Next**.
Feature Selection	1. Select **Database Engine Services** check box. 2. Select **Reporting Services** check box. 3. Select **Full-Text Search** check box. 4. Select **Management Tools - Complete** check box. 5. Select **Next**.
Installation Rules	Select **Next**.
Instance Configuration	Select **Next**.
Disk Space Requirements	Select **Next**.
Server Configuration	1. For **SQL Server Agent**, in **Account Name**, type **MDT2013\SQLAgent**, and in **Password**, type **P@ssw0rd**. 2. For **SQL Server Database Engine**, in **Account Name**, type **MDT2013\SQLDBEngine**, in **Password**, type **P@ssw0rd**. 3. For **SQL Server Reporting Services**, in **Account Name**, type **MDT2013\SQLReport**, in **Password**, type **P@ssw0rd**. 4. Select **Next**.
Database Engine Configuration	Select **Add Current User**, and select **Next**.
Reporting Services Configuration	Select **Next**.
Error Reporting	Select **Next**.
Installation Configuration Rules	Select **Next**.
Ready to Install	Select **Install**.
Complete	Select **Close**.

5. Close the SQL Server Installation Center.

To install SQL Server 2008 R2 SP1

1. In Windows Explorer, go to E:\Source$\SQL2008R2SP1, and double-click **SQLServer2008R2SP1-KB2528583-x64-ENU.exe**.

   The **Extracting Files** dialog box displays the file-extraction process. When the process is complete, the SQL Server 2008 R2 Service Pack 1 Update Setup Wizard starts.

2. Install SQL Server 2008 R2 SP1 using the following information, accepting the defaults unless otherwise specified.

Expand table

On this wizard page	Do this
SQL Server 2008 R2 update	Select **Next**.
License Terms	Select the **I accept the license terms** check box, and then select **Next**.
Select Features	Select **Next**.
Check Files In Use	Select **Next**.
Ready to update	Select **Update**.
Update Progress	The progress is displayed on the wizard page as the update is performed and completes.
Complete	Select **Close**.

To install SQL Server 2008 R2 SP1 CU6

1. In Windows Explorer, go to E:\Source$\SQL2008R2SP1CU6, and double-click **446622_intl_x64_zip.exe**.

   The **Microsoft Self-Extractor** dialog box appears.

2. In the **Microsoft Self-Extractor** dialog box, select **Continue**.

3. In the **Microsoft Self-Extractor** dialog box, in **Select the folder where you want to unzip the files to**, type **E:\Source$\SQL2008R2SP1CU6**, and then select **OK**.

> **Note**
>
> You can select the ellipsis (...) to browse for the E:\Source$\SQL2008R2SP1CU6 folder.

The extraction process is displayed. When the process is complete, the completion status is displayed.

4. In the **Microsoft Self-Extractor** dialog box, select **OK**.

5. In Windows Explorer, go to E:\Source$\SQL2008R2SP1CU6, and double-click **SQLServer2008R2-KB2679367-x64.exe**.

   The **Extracting Files** dialog box displays the file-extraction process. When the process is complete, the SQL Server 2008 R2 Service Pack 1 CU6 Update Setup Wizard starts.

6. Install SQL Server 2008 R2 SP1 CU6 using the following information, accepting the defaults unless otherwise specified.

Expand table

On this wizard page	Do this
SQL Server 2008 R2 update	Select **Next**.
License Terms	Select the **I accept the license terms** check box, and then select **Next**.
Select Features	Select **Next**.
Check Files In Use	Select **Next**.
Ready to update	Select **Update**.
Update Progress	The progress is displayed on the wizard page as the update is performed.
Complete	Select **Close**.

The **Install a SQL Server 2008 R2 Update** dialog box appears prompting you to restart the computer to complete the setup.

7. In the **Install a SQL Server 2008 R2 update** dialog box, select **OK**.

8. Restart the computer.

9. After installing SQL Server 2008 R2 SP1 CU6, the SQL Server build number should be 10.51.2811.0.

> **Tip**
>
> You can verify the SQL Server build number by viewing the SQL Server updates applied in the Programs and Features Control Panel item by selecting **View installed updates**.

Step 1-10: Add the Site Server to the Administrators Security Group

When all computers are in the same forest, manually add the site server computer account to the local Administrators group on each computer. Complete this step before configuring the computer as a site system.

To add the site server to the Administrators security group

1. Select **Start**, point to **Administrative Tools**, and then select **Active Directory Users and Computers**.
2. In the Active Directory Users and Computers console tree, go to mdt2013.corp.woodgrovebank.com/Builtin.
3. In the preview pane, right-click **Administrators**, and then select **Properties**.
4. In the **Administrators Properties** dialog box, select the **Members** tab, and then select **Add**.
5. In the **Select Users, Contacts, Computers, or Groups** dialog box, select **Object Types**.
6. In the **Object Types** dialog box, in **Object types**, select **Computers**, and then select **OK**.
7. In the **Select Users, Contacts, Computers, or Groups** dialog box, in **Enter the object names to select**, type **WDG-MDT-01**. Select **Check Names**, and then select **OK**.
8. Close any open windows.

Step 1-11: Install Configuration Manager

When the other products and technologies have been installed, install Configuration Manager. Before doing so, however, extend the Active Directory schema so that computers can locate the distribution points, service locator points, and other server roles. Also, you can extend the schema after you have installed Configuration Manager. For more information about how to extend the Active Directory schema for Configuration Manager, see the section, "Extend the Active Directory Schema," in the Configuration Manager Documentation Library, which is installed with Configuration Manager.

After extending the Active Directory schema, install Configuration Manager. The configuration of WDG-MDT-01 supports Configuration Manager for this sample. The configuration of computers in the production network may vary. To find out more about the prerequisites for installing Configuration Manager, see Supported Configurations for Configuration Manager.

To install Configuration Manager

1. Start the System Center 2012 R2 Configuration Manager Setup splash screen.
2. On the System Center 2012 R2 Configuration Manager Setup splash screen, select the **Install** link.

   The Microsoft System Center 2012 R2 Configuration Manager Setup Wizard starts.
3. Complete the Microsoft System Center 2012 R2 Configuration Manager Setup Wizard using the following information, accepting the defaults unless otherwise specified.

Expand table

On this wizard page	Do this
Before You Begin	Select **Next**.
Getting Started	Select **Next**.
Product Key	In **Enter your 25-character product key**, type ***product_key*** (where *product_key* is your product key for Configuration Manager).
Microsoft Software License Terms	Select the **I accept these license terms** check box, and then select **Next**.
Prerequisite Licenses	1. In the **Microsoft SQL Server 2008 R2 Express** section, select the **I accept these License Terms** check box. 2. In the **Microsoft SQL Server 2008 Native Client**

On this wizard page	Do this
	section, select the **I accept these License Terms** check box. 3. In the **Microsoft Silverlight 4** section, select the **I accept these License Terms and automatic updates of Silverlight** check box. 4. Select **Next**.
Update Prerequisite Components	In **Download and use the latest updates. Updates will be saved to the following location**, type **E:\CMDownloads**, and then select **Next**.
Server Language Selection	Select **Next**.
Client Language Selection	Select **Next**.
Site and Installation Settings	1. In **Site code**, type **NYC**. 2. In **Site name**, type **New York City Site**. 3. Select **Next**.
Primary Site Installation	1. Select **Install the primary site as a stand-alone site**. 2. Select **Next**. The **Configuration Manager** dialog box appears, confirming that you want to install this site as a stand-alone site. 3. In the **Configuration Manager** dialog box, select **Yes**.
Database Information	Select **Next**.
SMS Provider Settings	Select **Next**.
Client Computer Communication Settings	Select **Configure the communication method on each site system role**, and then select **Next**.
Site System Roles	Select **Next**.
Customer Experience Improvement Program Configuration	1. Select the appropriate participation in the Customer Experience Improvement program for your organization. 2. Select **Next**.
Settings Summary	Select **Next**.
Prerequisite Check	Select **Begin Install**.
Install	1. Monitor the installation process until it is complete. 2. Select **Close**.

4. Close all open windows and dialog boxes.

   When the wizard is complete, Configuration Manager is installed.

Step 1-12: Configure the Network Access Account

The Configuration Manager client needs an account to provide credentials when accessing the Configuration Manager distribution points, MDT deployment shares, and shared folders. This account is called the *Network Access account*. The CMNetAccess account was created earlier in the process to use as the Network Access account.

To configure the Network Access account

1. Select **Start**, point to **All Programs**, and then point to **Microsoft System Center 2012**. Point to **Configuration Manager**, and then select **Configuration Manager Console**.

2. In the Configuration Manager console, in the navigation pane, select **Administration**.

3. In the Administration workspace, go to Overview/Site Configuration/Sites.

4. In the preview pane, select **NYC - New York City Site**.

5. On the Ribbon, select **Settings**, select **Configure Site Components**, and then select **Software Distribution**.

6. In the **Software Distribution Component Properties** dialog box, select the **Network Access Account** tab.

7. In **Network Access Account**, select **Specify the account that accessed network locations**, select **Set**, and then select **New Account**.

   The **Windows User Account** dialog box appears.

8. Complete the **Windows User Account** dialog box using the following information, and then select **OK**.

   Expand table

For this	Do this
User name	Type **MDT2013\CMNetAccess**.
Password	Type **P@ssw0rd**.
Confirm password	Type **P@ssw0rd**.

9. In the **Software Distribution Component Properties** dialog box, select **OK**.

10. Close any open windows.

Step 1-13: Configure the Configuration Manager Site Boundaries and Boundary Groups

The Configuration Manager client needs to know the boundaries for the site. Unless the site boundaries are specified, the client assumes that the computer running Configuration Manager is in a remote site. Add a site boundary based on the IP subnet that WDG-MDT-01, WDG-REF-01, and WDG-CLI-01 use. Then add the site boundary to a site boundary group.

To create a Configuration Manager site boundary

1. Select **Start**, point to **All Programs**, and then point to **Microsoft System Center 2012**. Point to **Configuration Manager**, and then select **Configuration Manager Console**.

2. In the Configuration Manager console, in the navigation pane, select **Administration**.

3. In the Administration workspace, go to Overview/Hierarchy Configuration/Boundaries.

4. On the Ribbon, select **Create Boundary**.

   The **Create Boundary** dialog box opens.

5. Complete the **Create Boundary** dialog box using the following information, and then select **OK**.

   > **Note**
   >
   > For this sample, the site boundary is specified by network address. However, you can also specify site boundaries using an AD DS site name or an IP address range.

Expand table

For this	Do this
Description	Type **IP Subnet Boundary**.
Type	Select **IP subnet**.

For this	Do this
Network	Type ***network_address*** (where *network_address* is the network address of the subnet where the computers are installed).
Subnet mask	Type ***subnet_mask*** (where *subnet_mask* is the subnet mask of the subnet where the computers are installed).

To add the Configuration Manager site boundary to a site boundary group

1. In the Configuration Manager console, in the navigation pane, select **Administration**.
2. In the Administration workspace, go to Overview/Hierarchy Configuration/Boundary Groups.
3. On the Ribbon, select **Create Boundary Group**.

   The **Create Boundary Group** dialog box opens.
4. Complete the **General** tab of the **Create Boundary Group** dialog box using the following information.

Expand table

For this	Do this
Name	Type **New York City Boundary Group**.
Description	Type **This is the boundary group for the site boundaries at the New York City site**.
Boundaries	1. Select **Add**. The **Add Boundaries** dialog box appears. 2. In the **Add Boundaries** dialog box, select ***site_boundary*** (where *site_boundary* is the site boundary you created earlier in the process), and then select **OK**. The site boundary appears in the list of boundaries.

5. Complete the **References** tab of the **Create Boundary Group** dialog box using the following information, and then select **OK**.

Expand table

For this	Do this
Site assignment	Select the **Use this boundary group for site assignment** check box.
Content location	1. Select **Add**. The **Add Site Systems** dialog box appears. 2. In the **Add Site Systems** dialog box, select **\\WDG-MDT-01.mdt2013.corp.woodgrovebank.com**, and then select **OK**. The site system server appears in the list of site system servers.

6. Close any open windows.

Step 1-14: Configure the Publishing of Site Information in AD DS and DNS

The Configuration Manager client needs to locate the various Configuration Manager server roles. Modify the site properties to publish the site information in AD DS and in DNS.

To configure the publishing of site information in AD DS and in DNS

1. Select **Start**, point to **All Programs**, and then point to **Microsoft System Center 2012**. Point to **Configuration Manager**, and then select **Configuration Manager Console**.

2. In the Configuration Manager console, in the navigation pane, select **Administration**.

3. In the Administration workspace, go to Overview/Site Configuration/Sites.

4. In the preview pane, select **NYC - New York City Site**.

5. On the Ribbon, select **Properties**.

6. In the **New York City Site Properties** dialog box, on the **Publishing** tab, verify that the **mdt2013.corp.woodgrovebank.com** Active Directory forest is listed, and then select **Cancel**.

7. Close any open windows.

Step 2: Prepare the MDT Environment

The first step in the deployment process is to prepare the MDT environment. When this step is complete, you can create the reference computer and deploy a captured image

of it to the target computer (WDG-CLI-01) using Configuration Manager integration with MDT.

Prepare the MDT environment by:

- Installing MDT as described in Step 2-1: Install MDT
- Enabling Configuration Manager console integration by running the Configure ConfigMgr Integration script as described in Step 2-2: Enable Configuration Manager Console Integration

Step 2-1: Install MDT

To install MDT, complete the following steps:

1. In Windows Explorer, go to E:\Source$\MDT_2013.
2. Double-click **MicrosoftDeploymentToolkit2013_x64.msi** (for 64-bit operating systems) or **MicrosoftDeploymentToolkit2013_x86.msi** (for 32-bit operating systems), and then select **Install**.

   The Microsoft Deployment Toolkit 2013 Setup Wizard starts.
3. Complete the Microsoft Deployment Toolkit 2013 Setup Wizard using the information in the following table. Accept the default values unless otherwise specified.

Expand table

On this wizard page	Do this
Welcome to the Microsoft Deployment Toolkit 2013 Setup Wizard	Select **Next**.
End-User License Agreement	Select **I accept the terms in the License Agreement**, and then select **Next**.
Custom Setup	Select **Next**.
Ready to install Microsoft Deployment Toolkit 2013	Select **Install**.
Installing Microsoft Deployment Toolkit 2013	The progress for installing MDT is displayed.
Completing the Microsoft Deployment Toolkit 2013 Setup Wizard	Select **Finish**.

The Microsoft Deployment Toolkit 2013 Setup Wizard finishes, and MDT is installed on WDG-MDT-01.

Step 2-2: Enable Configuration Manager Console Integration

Before you can use the Configuration Manager integration features of MDT, run the Configure ConfigMgr Integration script. This script copies the appropriate integration files to the folder in which Configuration Manager is installed. The script also adds Windows Management Instrumentation (WMI) classes for the new MDT custom actions. The classes are added by compiling a new Managed Object Format (.mof) file that contains the new class definitions.

To enable Configuration Manager console integration

> **Note**
>
> Ensure that the Configuration Manager console is closed while performing these steps.

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Configure ConfigMgr Integration**.

   The Configure ConfigMgr Integration Wizard starts.

2. Complete the Configure ConfigMgr Integration Wizard using the information in the following table. Accept the default values unless otherwise specified.

Expand table

On this wizard page	Do this
Options	1. Verify that the **Install the MDT console extensions for System Center 2012 R2 Configuration Manager** check box is selected. 2. Verify that the **Add the MDT task sequence actions to a System Center 2012 R2 Configuration Manager server** check box is selected. 3. In **Site server name**, verify that the value is **WDG-MDT-01.mdt2013.corp.woodgrovebank.com**. 4. In **Site code**, verify that the value is **NYC**. 5. Select **Next**.
Confirmation	Select **Finish**.

The Configure ConfigMgr Integration Wizard finishes, and MDT is integrated with Configuration Manager.

Step 3: Create and Configure a Task Sequence to Create a Reference Computer

When you have prepared the MDT environment, create the reference computer. The reference computer is the template for deploying new images to the target computers. Configure this computer (WDG-REF-01) exactly as you will configure the target computers. You will then capture an image of the reference computer and deploy the image to the target computers.

Create the reference computer, WDG-REF-01, by:

- Creating an MDT task sequence to deploy Windows 8.1 to the reference computer as described in Step 3-1: Create an MDT Task Sequence for Deploying the Reference Computer
- Selecting the distribution points for the new packages and images that the Create MDT Task Sequence Wizard creates as described in Step 3-2: Select the Distribution Points for the New Packages and Images
- Adding the necessary device drivers to a new drive package and to the appropriate boot images as described in Step 3-3: Add the Necessary Device Drivers
- Enable monitoring of the MDT deployment process as described in Step 3-4: Enable MDT Deployment Process Monitoring
- Configuring the MDT configuration files for the reference computer—specifically, the CustomSettings.ini file—as described in Step 3-5: Customize the MDT Configuration Files for the Reference Computer
- Updating the Configuration Manager distribution points for the Custom Settings Files package as described in Step 3-6: Update the Distribution Points for the Custom Settings Files Package
- Customizing the task sequence for the reference computer as described in Step 3-7: Customize the Task Sequence for the Reference Computer

Step 3-1: Create an MDT Task Sequence for Deploying the Reference Computer

Use the Create MDT Task Sequence Wizard in the Configuration Manager console to create task sequences in Configuration Manager that are integrated with MDT. MDT includes the Standard Client Task Sequence template, which you can use to deploy the reference computer.

The Create MDT Task Sequence Wizard substitutes the packages and images selected for the placeholders in the task sequence templates. After completing the wizard, the new task sequence references the appropriate packages and images.

> **Note**
>
> Always use the Create MDT Task Sequence Wizard to create task sequences based on the MDT task sequence templates. Although you can manually import the task sequence templates, Microsoft does not recommend this process.

To create a task sequence for deploying the reference computer

1. Select **Start**, point to **All Programs**, and then point to **Microsoft System Center 2012**. Point to **Configuration Manager**, and then select **Configuration Manager Console**.

2. In the Configuration Manager console, in the navigation pane, select **Software Library**.

3. In the Software Library workspace, go to Overview/Operating Systems/Task Sequences.

4. On the Ribbon, on the **Home** tab, in the **Task Sequences** group, select **Create MDT Task Sequence**.

   The Create MDT Task Sequence Wizard starts.

5. Complete the Create MDT Task Sequence Wizard using the information in the following table. Accept the default values unless otherwise specified.

Expand table

On this wizard page	Do this
Choose Template	Select **Client Task Sequence**, and then select **Next**.
Choose Template: General	1. In **Task sequence name**, type **Windows 8.1 Reference Deployment**. 2. In **Task sequence comments**, type **Task sequence for deploying**

On this wizard page	Do this
	Windows 8.1 to the reference computer (WDG-REF-01), and then select **Next**.
Choose Template: Details	1. Select **Join a workgroup**. 2. In **Workgroup**, type **WORKGROUP**. 3. In **User name**, type **Woodgrove Bank Employee**. 4. In **Organization name**, type **Woodgrove Bank**. 5. In **Product key**, type ***product_key*** (where *product_key* is the product key for Windows 8.1). 6. Select **Next**.
Choose Template: Capture Settings	a. Select **This task sequence may be used to capture and image**. b. In **Capture destination**, type **\\WDG-MDT-01\Capture$\WDG-REF-01.wim**. c. In **Capture account**, select **Set**. d. Complete the **Windows User Account** dialog box by performing the following steps: i. In **User name**, type **MDT2013\Administrator**. ii. In **Password** and **Confirm password**, type **P@ssw0rd**. e. Select **OK**. f. Select **Next**.
Boot Image	1. Select **Create a new boot image package**. 2. In **Package source folder to be created**, type **\\WDG-MDT-01\Packages$\WINPE_Custom**, and then select **Next**.
Boot Image: General Settings	1. In **Name**, type **Windows PE Custom**. 2. In **Version**, type **1.00**. 3. In **Comments**, type **Customized version of Windows PE to be used in deployment of reference and target computers**, and then select **Next**.
Boot Image: Options	Under **Platform**, select **x64**, and then select **Next**.
Boot Image: Components	Select **Next**.
Boot Image: Customization	Select **Next**.
MDT Package	1. Select **Create a new Microsoft Deployment Toolkit Files package**. 2. In **Package source folder to be created**, type **\\WDG-MDT-01\Packages$\MDT_Files**, and then select **Next**.
MDT Package: MDT Details	1. In **Name**, type **MDT Files**. 2. In **Version**, type **1.00**.

On this wizard page	Do this
	3. In **Comments**, type **Provides access to MDT files during Configuration Manager deployment process**, and then select **Next**.
OS Image	1. Select **Create a new OS install package**. 2. In **OS installation folder location**, type **\\WDG-MDT-01\Source$\Windows_7**. 3. In **Package source folder to be created**, type **\\WDG-MDT-01\Packages$\Windows_7**, and then select **Next**.
OS Image: Image Details	1. In **Name**, type **Windows 8.1**. 2. In **Version**, type **1.00**. 3. In **Comments**, type **Windows 8.1 package used to deploy to reference computers**, and then select **Next**.
Deployment Method	Select **Next**.
Client Package	Select **Create a new ConfigMgr client package**, and then select **Next**.
USMT Package	1. Select **Create a new USMT package**. 2. In **Package source folder to be created**, type **\\WDG-MDT-01\Packages$\USMT**, and then select **Next**.
USMT Package: USMT Details	1. In **Name**, type **USMT**. 2. In **Version**, type **1.00**. 3. In **Comments**, type **USMT files used to capture and restore user state migration information**, and then select **Next**.
Settings Package	1. Select **Create a new settings package**. 2. In **Package source folder to be created**, type **\\WDG-MDT-01\Packages$\CustomSettings_Reference**, and then select **Next**.
Settings Package: Settings Details	1. In **Name**, type **MDT Reference Computer Custom Settings**. 2. In **Version**, type **1.00**. 3. In **Comments**, type **Configuration settings for MDT deployment process (such as CustomSettings.ini) for the reference computer**, and then select **Next**.
Sysprep Package	Select **Next**.
Summary	1. Review the information in the **Details** box that you provided while completing the previous wizard pages. 2. Select **Next**.
Progress	The progress for creating the task sequence is displayed.
Confirmation	Select **Finish**.

The new task sequence appears in the preview pane.

Step 3-2: Select the Distribution Points for the New Packages and Images

The Create MDT Task Sequence Wizard creates a number of packages and images. After these packages and images are created, select the distribution points from which the packages and images will be copied and available to target computers.

> **Note**
>
> In this sample, there is only one distribution point (WDG-MDT-01). However, most production networks have multiple distribution points. When performing this step in a production environment, select the appropriate distribution points for the network.

To select the distribution points for software distribution packages

1. Select **Start**, point to **All Programs**, and then point to **Microsoft System Center 2012**. Point to **Configuration Manager**, and then select **Configuration Manager Console**.
2. In the Configuration Manager console, in the navigation pane, select **Software Library**.
3. In the Software Library workspace, go to Overview/Operating Systems/Task Sequences.
4. In the preview pane, select **Windows 8.1 Reference Deployment**.
5. On the Ribbon, on the **Home** tab, in the **Deployment** group, select **Distribute Content**.

   The Distribute Content Wizard starts.
6. Complete the Distribute Content Wizard using the information in the following table. Accept default values unless otherwise specified.

Expand table

On this wizard page	Do this
General	Select **Next**.

On this wizard page	Do this
General: Content	Select **Next**.
General: Content Destination	1. Select **Add**, and then select **Distribution Point**. The **Add Distribution Points** dialog box appears. 2. In the **Add Distribution Points** dialog box, select **WDG-MDT-01.mdt2013.corp.woodgrovebank.com**, and then select **OK**. WDG-MDT-01.corp.woodgrovebank.com appears in the **Content destination** list. 3. Select **Next**.
Summary	1. Review the information in the **Details** box that you provided while completing the previous wizard pages. 2. Select **Next**.
Progress	The progress for distributing the software is displayed.
Completion	Select **Close**.

7. Close all open windows and dialog boxes.

Step 3-3: Add the Necessary Device Drivers

When the MDT task sequence has been created, add any device drivers required for the reference computer (WDG-REF-01) to the Windows PE boot image and to the Windows 8.1 image. Add the device drivers in the Drivers node in the Configuration Manager console. Create a package that contains the device drivers, and inject the drivers into the custom Windows PE image created earlier in the process.

After creating the package that contains the device drivers, select the distribution point to which the package will be deployed.

To add the necessary device drivers

1. Select **Start**, point to **All Programs**, and then point to **Microsoft System Center 2012**. Point to **Configuration Manager**, and then select **Configuration Manager Console**.

2. In the Configuration Manager console, in the navigation pane, select **Software Library**.

3. In the Software Library workspace, go to Overview/Operating Systems/Drivers.

4. On the Ribbon, on the **Home** tab, in the **Create** group, select **Import Driver**.

The Import New Driver Wizard starts.

5. Complete the Import New Driver Wizard using the information in the following table. Accept the default values unless otherwise specified.

Expand table

On this wizard page	Do this
Locate Driver	In **Source folder**, type **\\WDG-MDT-01\Source$\Drivers**, and then select **Next**.
Locate Driver: Driver Details	Select **Next**.
Locate Driver: Add Driver to Package	a. Select **New Package**. b. Complete the **New Driver Package** dialog box by performing the following steps: i. In **Name**, type ***device_driver_name*** Package (where *device_driver_name* is a descriptive name for the device drivers). ii. In **Comment**, type **Device drivers that are necessary for the reference and target computers**. c. In **Driver package source**, type **\\WDG-MDT-01\Packages$\Drivers**, and then select **OK**. d. Select **Next**.
Locate Driver: Add Driver to Boot Images	1. In the list of images, select the **Windows PE Custom** check box. 2. Select the **Update distribution points when finished** check box, and then select **Next**.
Summary	1. Review the information in the **Details** box that you provided while completing the previous wizard pages. 2. Select **Next**.
Progress	The progress for importing the device drivers is displayed.
Confirmation	Select **Close**.

To select the distribution points for the driver package

1. Select **Start**, point to **All Programs**, and then point to **Microsoft System Center 2012**. Point to **Configuration Manager**, and then select **Configuration Manager Console**.

2. In the Configuration Manager console, in the navigation pane, select **Software Library**.
3. In the Software Library workspace, go to Overview/Operating Systems/Driver Packages.
4. In the preview pane, select ***device_driver_name* Package** (where *device_driver_name* is a descriptive name for the device drivers).
5. On the Ribbon, on the **Home** tab, in the **Deployment** group, select **Distribute Content**.

   The Distribute Content Wizard starts.
6. Complete the Distribute Content Wizard using the following information. Accept the default values unless otherwise specified.

Expand table

On this wizard page	Do this
General	Select **Next**.
General: Content	Select **Next**.
General: Content Destination	1. Select **Add**, and then select **Distribution Point**. The **Add Distribution Points** dialog box appears. 2. In the **Add Distribution Points** dialog box, select **\\WDG-MDT-01.mdt2013.corp.woodgrovebank.com**, and then select **OK**. \\WDGMDT01.mdt2013.corp.woodgrovebank.com appears in the **Content destination** list. 3. Select **Next**.
Summary	1. Review the information in the **Details** box that you provided while completing the previous wizard pages. 2. Select **Next**.
Progress	The progress for distributing the software is displayed.
Completion	Select **Close**.

7. Close all open windows and dialog boxes.

Step 3-4: Enable MDT Deployment Process Monitoring

Prior to deploying the reference computer (WDG-REF-01) with the task sequence bootable media, enable MDT monitoring of the ZTI deployment process. You enable monitoring on the **Monitoring** tab on the deployment share **Properties** dialog box. Later in the process, you will monitor the ZTI deployment process using the Deployment Workbench or the **Get-MDTMonitorData** cmdlet.

To enable MDT monitoring of the ZTI deployment process

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares.
3. In the Actions pane, select **New Deployment Shares**.

   The New Deployment Share Wizard starts.
4. Complete the New Deployment Share Wizard using the following information

Expand table

On this wizard page	Do this
On this wizard page	**Do this**
Path	In **Deployment share path**, type **C:\DeploymentShare$**, and then select **Next**.
Share	Select **Next**.
Descriptive Name	Select **Next**.
Options	Select **Next**.
Summary	Select **Next**.
Progress	The progress for creating the deployment share is displayed.
Confirmation	Select **Finish**.

   The New Deployment Share Wizard finishes, and the new deployment share—MDT Deployment Share (C:\DeploymentShare$)—appears in the details pane.
5. In the details pane, select **MDT Deployment Share (C:\DeploymentShare$)**.
6. In the Actions pane, select **Properties**.

The **MDT Deployment Share (C:\DeploymentShare$) Properties** dialog box opens.

7. In the **MDT Deployment Share (C:\DeploymentShare$) Properties** dialog box, on the **Monitoring** tab, select the **Enable monitoring for this deployment share** check box, and then select **Apply**.

8. In the **MDT Deployment Share (C:\DeploymentShare$) Properties** dialog box, on the **Rules** tab, notice that the **EventService** property has been added to the CustomSettings.ini file, and then select **OK**.

   The **EventService** property is as follows:

   ```ini
   EventService=http://WDG-MDT-01:9800
   ```

9. Close all open windows and dialog boxes.

Step 3-5: Customize the MDT Configuration Files for the Reference Computer

When the MDT task sequence has been created, customize the MDT configuration files that provide the configuration settings for deploying Windows 8.1 to the target computer. Specifically, customize the CustomSettings.ini file.

When the CustomSettings.ini file customization is finished, save the updated files to the source folder for the MDT Reference Computer Custom Settings package created earlier in the process (E:\Packages$\CustomSettings_Reference). Then, add the **DoCapture** and **EventService** properties and corresponding values to the CustomSettings.ini file so that the MDT deployment process captures an image of the reference computer (WDG-REF-01) after deploying Windows 8.1.

To customize the MDT configuration files for the reference computer

1. In Windows Explorer, go to E:\Packages$\CustomSettings_Reference, and then double-click **CustomSettings.ini**.

2. Open Microsoft Notepad, and then add the following lines to the end of the CustomSettings.ini file:

   ```ini
   ```

```
DoCapture=YES
EventService=http://WDG-MDT-01:9800
```

Example of CustomSettings.ini file after adding the DoCapture property:

```ini
[Settings]
Priority=Default
Properties=MyCustomProperty

[Default]
OSInstall=Y
SkipCapture=YES
SkipAdminPassword=NO
SkipProductKey=YES
DoCapture=YES
EventService=http://WDG-MDT-01:9800
```

3. In Notepad, save the file, and then exit Notepad.

Step 3-6: Update the Distribution Points for the Custom Settings Files Package

When the source folder has been updated for the MDT Reference Computer Custom Settings package in Configuration Manager, update the distribution points for the MDT Reference Computer Custom Settings Files package. Updating the distribution points copies the updated version of the CustomSettings.ini file to the deployment shares specified in the package.

To update the distribution points for the Custom Settings package

1. Select **Start**, point to **All Programs**, and then point to **Microsoft System Center 2012**. Point to **Configuration Manager**, and then select **Configuration Manager Console**.

2. In the Configuration Manager console, in the navigation pane, select **Software Library**.

3. In the Software Library workspace, go to Overview/Application Management/Packages.

4. In the preview pane, select **MDT Reference Computer Custom Settings**.

5. On the Ribbon, on the **Home** tab, in the **Deployment** group, select **Update Distribution Points**.

   The **Configuration Manager** dialog box opens, notifying you that you are going to update the package on all distribution points.

6. In the **Configuration Manager** dialog box, select **OK**.

7. Close all open windows and dialog boxes.

   Configuration Manager starts updating the distribution points with the latest versions of the CustomSettings.ini file. This process could take several minutes. Check the status of the package until the **Last Update** value of the package status has been updated to a recent date and time.

Step 3-7: Customize the Task Sequence for the Reference Computer

For most deployments, the **Windows 8.1 Reference Deployment** task sequence created earlier in the process performs all the necessary steps without modification. In this sample, modify the task sequence to set the password for the local Administrator account to a known value. By default, the task sequence sets the password for the local Administrator account to a random value. Further customization of the task sequence may be required depending on the environment.

To customize the Windows 8.1 Reference Deployment task sequence

1. Select **Start**, point to **All Programs**, and then point to **Microsoft System Center 2012**. Point to **Configuration Manager**, and then select **Configuration Manager Console**.

2. In the Configuration Manager console, in the navigation pane, select **Software Library**.

3. In the Software Library workspace, go to Overview/Operating Systems/Task Sequences.

4. In the preview pane, select **Windows 8.1 Reference Deployment**.

5. On the Ribbon, on the **Home** tab, in the **Task Sequence** group, select **Edit**.

   The **Windows 8.1 Reference Deployment Task Sequence Editor** dialog box opens.

6. In the **Windows 8.1 Reference Deployment Task Sequence Editor** dialog box, go to PostInstall/Apply Windows Settings.

7. On the **Properties** tab, select **Enable the account and specify the local administrator password**.
8. On the **Properties** tab, in **Password** and **Confirm Password**, type **P@ssw0rd**, and then select **Apply**.
9. Make any additional modifications to the task sequence that the environment requires, and then select **OK**.
10. Close all open windows and dialog boxes.

Step 4: Deploy Windows 8.1 and Capture an Image of the Reference Computer

When you have created the task sequence to deploy Windows 8.1 to the reference computer and captured an image of the reference computer, start the task sequence. Create the operating system capture by using the Task Sequence Media Wizard in the Configuration Manager console.

Deploy Windows 8.1 and capture an image of the reference computer by:

- Adding the reference computer to the Configuration Manager site database as described in Step 4-1: Add the Reference Computer to the Configuration Manager Site Database
- Creating a collection that contains the reference computer you just added as described in Step 4-2: Create a Collection That Contains the Reference Computer
- Deploy the reference computer task sequence as described in Step 4-3: Deploy the Reference Computer Task Sequence
- Using the Task Sequence Media Wizard to create a task sequence bootable media disk as described in Step 4-4: Create the Task Sequence Bootable Media
- Starting the reference computer with the task sequence bootable media disk as described in Step 4-5: Start the Reference Computer with the Task Sequence Bootable Media

Step 4-1: Add the Reference Computer to the Configuration Manager Site Database

To deploy an operating system without stand-alone media to a new computer that Configuration Manager does not currently manage, add the new computer to the

Configuration Manager site database prior to initiating the operating system deployment process. Configuration Manager can automatically discover computers on the network that have a Windows operating system installed; however, if the computer has no operating system installed, use the Import Computer Information Wizard to import the new computer information.

To add the reference computer to the Configuration Manager site database

1. Select **Start**, point to **All Programs**, and then point to **Microsoft System Center 2012**. Point to **Configuration Manager**, and then select **Configuration Manager Console**.

2. In the Configuration Manager console, in the navigation pane, select **Assets and Compliance**.

3. In the Assets and Compliance workspace, go to Overview/Devices.

4. On the Ribbon, on the **Home** tab, in the **Create** group, select **Import Computer Information**.

   The Import Computer Information Wizard starts.

5. Complete the Import Computer Information Wizard using the following information. Accept the default values unless otherwise specified.

Expand table

On this wizard page	Do this
Select Source	Select **Import single computer**, and then select **Next**.
Select Source: Single Computer	1. In **Computer Name**, type **WDG-REF-01**. 2. In **MAC address**, type ***mac_address*** (where *mac_address* is the media access control [MAC] address of the primary network adapter for the reference computer, WDG-REF-01). 3. Select **Next**.
Select Source: Data Preview	Select **Next**.
Select Source: Choose Target Collection	Select **Next**.
Summary	1. Review the information in the **Details** box that you provided while completing the previous wizard pages.

On this wizard page	Do this
	2. Select **Next**.
Progress	The progress for importing the computer is displayed.
Confirmation	Select **Close**.

For more information on adding a new computer to the Configuration Manager site database, see the section, "To import computer information for a single computer," in the section "How to Deploy Operating Systems in Configuration Manager," in the Configuration Manager Documentation Library, which is installed with Configuration Manager.

Step 4-2: Create a Collection That Contains the Reference Computer

In the Configuration Manager console, create a collection that includes the reference computer (WDG-REF-01). This computer collection is used later when advertising the task sequence created earlier in the process.

To create a collection that includes the reference computer

1. Select **Start**, point to **All Programs**, and then point to **Microsoft System Center 2012**. Point to **Configuration Manager**, and then select **Configuration Manager Console**.
2. In the Configuration Manager console, in the navigation pane, select **Assets and Compliance**.
3. In the Assets and Compliance workspace, go to Overview/Device Collections.
4. On the Ribbon, on the **Home** tab, in the **Create** group, select **Create**, and then select Create Device **Collection**.

   The Create Device Collection Wizard starts.
5. Complete the Create Device Collection Wizard using the following information. Accept the default values unless otherwise specified.

Expand table

On this wizard page	Do this
General	a. In **Name**, type **Microsoft Deployment - Reference Computer**. b. In **Comment**, type **Computer that is to be the reference computer for the target computers to be deployed**. c. In **Limited Collection**, select **Browse**. The **Select Collection** dialog box appears. Complete the dialog box by performing the following steps: i. In **Name**, select **All Systems**. ii. Select **OK**. d. Select **Next**.
Membership Rules	a. Select **Add Rule**, and then select **Direct Rule**. The Create Direct Membership Rule Wizard starts. b. Complete the Create Direct Membership Rule Wizard by performing the following steps: i. On the **Welcome** page, select **Next**. ii. On the **Search for Resources** page, in **Resource class**, select **System Resource**; in **Attribute name**, select **Name**; in **Value**, type **WDG-REF-01**; and then select **Next**. iii. On the **Select Resources** page, select **WDG-REF-01**, and then select **Next**. iv. On the **Summary** page, select **Next**. v. On the **Progress** page, view the progress for creating the new membership rule. vi. On the **Completion** page, select **Close**. c. Select **Next**.
Summary	1. Review the information in the **Details** box that you provided while completing the previous wizard pages. 2. Select **Next**.
Progress	The progress for creating the device collection is displayed.
Completion	Select **Close**.

For more information, see the section, "How to Create Collections in Configuration Manager," in the Configuration manager Documentation Library, which is installed with Configuration Manager.

Step 4-3: Deploy the Reference Computer Task Sequence

In the Configuration Manager console, deploy the task sequence created earlier in the process to the device collection that includes the reference computer created earlier in the process.

To deploy the task sequence

1. Select **Start**, point to **All Programs**, and then point to **Microsoft System Center 2012**. Point to **Configuration Manager**, and then select **Configuration Manager Console**.

2. In the Configuration Manager console, in the navigation pane, select **Software Library**.

3. In the Software Library workspace, go to Overview/Operating Systems/Task Sequences.

4. In the preview pane, select **Windows 8.1 Reference Deployment**.

5. On the Ribbon, on the **Home** tab, in the **Deployment** group, select **Deploy**.

   The Deploy Software Wizard starts.

6. Complete the Deploy Software Wizard using the following information. Accept the default values unless otherwise specified.

Expand table

On this wizard page	Do this
General	1. In **Collection**, select **Browse**. 2. In the **Browse Collection** dialog box, select **Microsoft Deployment - Reference Computer**, and then select **OK**. 3. In **Comment**, type **Deploy Windows 8.1 to the reference computer and then capture an image of the reference computer**. 4. Select **Next**.
Deployment Settings	1. In **Purpose**, select **Available**. 2. Select the **Make available to boot media and PXE** check box. 3. Select **Next**.
Deployment Settings: Schedule	Select **Next**.
Deployment Settings: User Experience	Select **Next**.

On this wizard page	Do this
Deployment Settings: Alerts	Select **Next**.
Deployment Settings: Distribution Points	Select **Next**.
Summary	1. Review the information in the **Details** box that you provided while completing the previous wizard pages. 2. Select **Next**.
Progress	The progress for deploying the task sequence is displayed.
Completion	Select **Close**.

For more information, see the section, "How to Deploy a Task Sequence," in the Configuration manager Documentation Library, which is installed with Configuration Manager.

Step 4-4: Create the Task Sequence Bootable Media

To initiate the MDT process, provide a method for starting the computer with Windows PE and the necessary software by creating the task sequence bootable media disk. Use the Task Sequence Media Wizard in the Configuration Manager console to create bootable media for storage on a USB flash drive, CD, or DVD.

To create a task sequence bootable media disk

1. Select **Start**, point to **All Programs**, and then point to Microsoft System Center 2012. Point to **Configuration Manager**, and then select **Configuration Manager Console**.

2. In the Configuration Manager console, in the navigation pane, select **Software Library**.

3. In the Software Library workspace, go to Overview/Operating Systems/Task Sequences.

4. On the Ribbon, on the **Home** tab, in the **Create** group, select **Create Task Sequence Media**.

   The Create Task Sequence Media Wizard starts.

5. Complete the Create Task Sequence Media Wizard using the following information. Accept the default values unless otherwise specified.

Expand table

On this wizard page	Do this
Select Media Type	1. Select **Bootable media**. 2. Clear the **Allow unattended operating system deployment** check box. 3. Select **Next**.
Select Media Type: Media Management	Select **Site-based media**, and then select **Next**.
Select Media Type: Media Type	In **Media file**, type **\\WDG-MDT-01\Capture$\CM2012_TS_Boot_Media.iso**, and then select **Next**.
Select Media Type: Security	In **Password** and **Confirm password**, type **P@ssw0rd**, and then select **Next**.
Select Media Type: Boot Image	1. In **Boot image**, select **Browse**. 2. In the **Select a Boot Image** dialog box, select **Windows PE Custom**, and then select **OK**. 3. In **Distribution point**, select **\\WDG-MDT-01.mdt2013.corp.woodgrovebank.com**, and then select **OK**. 4. In **Management point**, select **\\WDG-MDT-01.mdt2013.corp.woodgrovebank.com**, and then select **OK**. 5. Select **Next**.
Select Media Type: Customization	Select **Next**.
Summary	1. Review the information in the **Details** box that you provided while completing the previous wizard pages. 2. Select **Next**.
Progress	The progress for creating the task sequence media is displayed.
Completion	Select **Close**.

The wizard creates the CM2012_TS_Boot_Media.iso file in the WDG-MDT-01Capture$ shared folder.

6. If WDG-REF-01 is a physical computer, create a CD or DVD of the International Organization for Standardization (ISO) file. If WDG-REF-01 is a VM, start the VM directly from the ISO file.

For more information on creating the task sequence bootable media disk, see the section, "How to Create Bootable Media," in the Configuration manager Documentation Library, which is installed with Configuration Manager.

Step 4-5: Start the Reference Computer with the Task Sequence Bootable Media

Start the reference computer (WDG-REF-01) with the task sequence bootable media disk created earlier in the process. This medium starts Windows PE on the reference computer and initiates the MDT process. At the end of the MDT process, Windows 8.1 is deployed on the reference computer and an image of the reference computer is saved to \WDG-MDT-01\Capture$\WDG-REF-01.wim.

> **Note**
>
> You can also initiate the MDT process by starting the target computer from Windows Deployment Services.

To start the reference computer with the task sequence bootable media

1. Start WDG-REF-01 with the task sequence bootable media created earlier in the process.

   Windows PE starts, and then the Task Sequence Wizard starts.

2. Complete the Task Sequence Wizard using the following information. Accept the default values unless otherwise specified.

Expand table

On this wizard page	Do this
Welcome to the Task Sequence Wizard	In **Password**, type **P@ssw0rd**, and then select **Next**.
Select a Task Sequence	In the list box, select **Windows 8.1 Reference Deployment**, and then select **Next**.

To monitor the reference computer deployment process using the Deployment Workbench, complete the following steps on WDG-

MDT-01

1. On WDG-MDT-01, select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/MDT Deployment Share (C:\DeploymentShare$)/Monitoring.
3. In the details pane, view the deployment process for **WDG-REF-01**.
4. In the Actions pane, periodically select **Refresh**.

   The status of the deployment process is updated in the details pane. Continue to monitor the deployment process until the process is complete.
5. In the details pane, select **WDG-REF-01**.
6. In the Actions pane, select **Properties**.

   The **WDG-REF-01 Properties** dialog box is displayed.
7. In the **WDG-REF-01 Properties** dialog box, on the **Identity** tab, view the monitoring information provided about the deployment process as follows:

Expand table

Information	Description
ID	Unique identifier for the computer being deployed.
Computer Name	The name of the computer being deployed.
Deployment status	The current status of the computer being deployed; the status can be one of the following: - **Running**. The task sequence is healthy and running. - **Failed**. The task sequence failed, and the deployment process was unsuccessful. - **Completed**. The task sequence has finished. - **Unresponsive**. The task sequence has not updated its status in the past four hours and is assumed to be nonresponsive.
Step	The current task sequence step being run.
Progress	The overall progress of the task sequence. The progress bar indicates how many task sequence steps have been run out of the total number of task sequence steps.

Information	Description
Start	The time the deployment process started.
End	The time the deployment process ended.
Elapsed	The length of time the deployment process has been running or took to run if the deployment process has finished.
Errors	The number of errors encountered during the deployment process.
Warnings	The number of warnings encountered during the deployment process.
Remote Desktop	This button allows you to establish a remote desktop connection with the computer being deployed using the Windows Remote Desktop feature. This method assumes that: - The target operating system is running and has remote desktop support enabled - **mstsc.exe** is in the path **Note:** This button is always visible but may not be able to establish a remote desktop session if the monitored computer is running Windows PE, has not completed installation of the target operating system, or does not have the Remote Desktop feature enabled.
VM Connection	This button allows you to establish a remote desktop connection to a VM running in HyperV®. This method assumes that: - The deployment is being performed to a VM running on Hyper-V - **vmconnect.exe** is located in the %ProgramFiles%\Hyper-V folder **Note:** This button appears when ZTIGather.wsf detects that Hyper-V integration components are running on the monitored computer. Otherwise, this button will not be visible.
DaRT Remote Control	This button allows you to establish a remote control session using the remote viewer feature in the Diagnostics and Recovery Toolkit (DaRT). This method assumes that: - DaRT has been deployed to the target computer and is currently running - **DartRemoteViewer.exe** is located in the %ProgramFiles%\Microsoft DaRT 7\v7 folder **Note:** This button appears when ZTIGather.wsf detects that DaRT is running on the monitored computer. Otherwise, this button will not be visible.
Automatically refresh this	Check box that controls whether the information in the dialog box is automatically refreshed. If the check box is:

Information	Description
information every 10 seconds	- Selected, the information is refreshed every 10 seconds - Cleared, the information is not automatically refreshed and must be manually refreshed using the **Refresh Now** button
Refresh Now	This button immediately refreshes the information displayed in the dialog box.

8. In the **WDG-REF-01 Properties** dialog box, select **OK**.

9. Close the Deployment Workbench.

To monitor the reference computer deployment process using the Get-MDTMonitorData cmdlet, complete the following steps on WDG-MDT-01

1. On WDG-MDT-01, select **Start**, the select **Administrative Tools**, and then select **Windows PowerShell Modules**.

   The Windows PowerShell Modules command prompt opens.

2. Create a PowerShell drive that uses the MDT PowerShell provider by running the New-PSDrive cmdlet as shown in the following example:

   ```
   PowerShell

   New-PSDrive -Name DS001 -PSProvider mdtprovider -Root
   d:\DeploymentShare$
   ```

3. View the MDT monitoring process by running the **Get-MDTMonitorData** cmdlet as shown in the following example:

   ```
   PowerShell

   Get-MDTMonitorData -Path DS001:
   ```

   This command returns the monitoring data collected by the MDT monitoring service running on the same computer that hosts the deployment share, as shown in the following example output:

   ```
   PowerShell

   Name               : WDG-REF-01
   PercentComplete    : 96
   Settings           :
   ```

```
Warnings           : 0
Errors             : 0
DeploymentStatus   : 1
StartTime          : 6/7/2012 6:45:39 PM
EndTime            :
ID                 : 1
UniqueID           : 94a0830e-f2bb-421c-b1e0-6f86f9eb9fa1
CurrentStep        : 130
TotalSteps         : 134
StepName           : Gather
LastTime           : 6/7/2012 8:46:32 PM
DartIP             :
DartPort           :
DartTicket         :
VMHost             : XYL-DC-02
VMName             : WDG-REF-01
ComputerIdentities : {}
```

4. Close the Windows PowerShell console.

   If any problems occur during the deployment, consult the MDT document *Troubleshooting Reference*. When completed, a captured image of the reference computer should exist in \\WDG-MDT-01\Capture$\WDG-REF-01.wim.

Step 5: Create and Configure a Task Sequence to Deploy the Target Computer

After the task sequence to deploy the reference computer (WDG-REF-01) finishes, a captured image of the reference computer is stored in \\WDG-MDT-01\Capture$\WDG-REF-01.wim. Now, create a task sequence that will deploy the captured image of the reference computer to the target computer (WDG-CLI-01). When this step is complete, you can deploy the captured image of the reference computer to the target computer.

Create and configure a task sequence to deploy the target computer by:

- Importing the .wim file captured in the previous step into Configuration Manager using the Add Operating System Image Wizard as described in Step 5-1: Import the Captured .wim File into Configuration Manager
- Using the Create MDT Task Sequence Wizard to create an MDT task sequence template to deploy the captured image of the reference computer to the target computer as described in Step 5-2: Create an MDT Task Sequence to Deploy the Captured Image
- Selecting the distribution points for the new packages and images that the Create MDT Task Sequence Wizard creates as described in Step 5-3: Select the Distribution

Points for the New Packages and Images

- Customizing the MDT configuration files for the target computer—specifically, the CustomSettings.ini file—as described in Step 5-4: Customize the MDT Configuration Files
- Updating the Configuration Manager distribution points for the Custom Settings package as described in Step 5-5: Update the Distribution Points for the Custom Settings Package
- Customizing the task sequence for the target computer as described in Step 5-6: Customize the Task Sequence for the Target Computer

Step 5-1: Import the Captured .wim File into Configuration Manager

After the image of the reference computer (WDG-REF-01) is captured in to the .wim file, import the captured .wim file into Configuration Manager. Import the captured .wim file into the Operating System Images node using the Add Operating System Image Wizard.

The captured .wim contains two images, one for each partition on the reference computer. Identify which of the images has the captured Windows 8.1 operating system by checking the image description containing *Windows 8.1*. You will use the image index when you create the task sequence for deploying the captured image to the target computer.

To import the captured .wim file into Configuration Manager

1. Select **Start**, point to **All Programs**, and then point to **Microsoft System Center 2012**. Point to **Configuration Manager**, and then select **Configuration Manager Console**.
2. In the Configuration Manager console, in the navigation pane, select **Software Library**.
3. In the Software Library workspace, go to Overview/Operating Systems/Operating System Images.
4. On the Ribbon, in the **Create** group, select **Add Operating System Image**.

   The Add Operating System Image Wizard starts.
5. Complete the Add Operating System Image Wizard using the following information. Accept the default values unless otherwise specified.

Expand table

On this wizard page	Do this
On this wizard page	**Do this**
Data Source	In **Path**, type **\\WDG-MDT-01\Capture$\WDG-REF-01.wim**, and then select **Next**.
General	1. In **Name**, type **Windows 8.1 Reference Image**. 1. In **Version**, type **1.00**. 1. In **Comments**, type **Windows 8.1 captured image of reference computer (WDG-REF-01) used to deploy to target computers**, and then select **Next**.
Summary	1. Review the information in the **Details** box that you provided while completing the previous wizard pages. 2. Select **Next**.
Progress	The progress for importing the operating system image is displayed.
Completion	Select **Close**.

6. In the preview pane, select **Windows 8.1 Reference Image**.

7. In the preview pane, select the **Details** tab.

   The list of operating system partitions captured in the .wim is displayed. The image index that contains Windows 8.1 is the image index you will specify later during the Create MDT Task Sequence Wizard.

8. Record the image index that contains Windows 8.1.

> **Tip**
>
> For the purposes of this example, image index 2 should have the Windows 8.1 operating system.

Step 5-2: Create an MDT Task Sequence to Deploy the Captured Image

After the image is captured, create a task sequence to deploy the captured image of the reference computer (WDG-REF-01) to the target computer (WDG-CLI-01). Most of the packages needed for this task sequence were created earlier in the process. However, you must create a new MDT Custom Settings package that has the proper configuration settings for the target computer and creates an operating system image of the captured image of the reference computer.

To create a task sequence template to deploy the captured image to the target computer

1. Select **Start**, point to **All Programs**, and then point to **Microsoft System Center 2012**. Point to **Configuration Manager**, and then select **Configuration Manager Console**.

2. In the Configuration Manager console, in the navigation pane, select **Software Library**.

3. In the Software Library workspace, go to Overview/Operating Systems/Task Sequences.

4. On the Ribbon, on the **Home** tab, in the **Task Sequences** group, select **Create MDT Task Sequence**.

   The Create MDT Task Sequence Wizard starts.

5. Complete the Create MDT Task Sequence Wizard using the following information. Accept the default values unless otherwise specified.

Expand table

On this wizard page	Do this
Choose Template	Select **Client Task Sequence**, and then select **Next**.
Choose Template: General	1. In **Task sequence name**, type **Windows 8.1 Target Deployment**. 2. In **Task sequence comments**, type **Task sequence for deploying captured reference computer image to the target computer (WDG-CLI-01)**, and then select **Next**.
Choose Template: Details	a. Select **Join a domain**. b. In **Domain**, type **mdt2013.corp.woodgrovebank.com**. c. In **Account**, select **Set**, and then complete the **Windows User Account** dialog box by performing the following steps:

On this wizard page	Do this
	i. In **User name**, type **MDT2013\Administrator**. ii. In **Password** and **Confirm password**, type **P@ssw0rd**. iii. Select **OK**. d. In **User name**, type **Woodgrove Bank Employee**. e. In **Organization name**, type **Woodgrove Bank**. f. In **Product key**, type ***product_key*** (where *product_key* is the product key for Windows 8.1). g. Select **Next**.
Choose Template: Capture Settings	Select **Next**.
Boot Image	1. In **Specify an existing boot image package**, select **Browse**. 2. In **Select a Package** dialog box, select **Windows PE Custom**, and then select **OK**. 3. Select **Next**.
MDT Package	1. In **Specify an existing Microsoft Deployment Toolkit Files package**, select **Browse**. 2. In the **Select a Package** dialog box, select **MDT Files**, and then select **OK**. 3. Select **Next**.
OS Image	1. Select **Specify an existing OS image**. 2. In **Specify an existing OS image**, select **Browse**. 3. In the **Select a Package** dialog box, select **Windows 8.1 Reference Image**, and then select **OK**. 4. Select **Next**.
OS Image: OS Image Index	1. In **The selected operating system image (WIM) file contains multiple images. Specify which image you would like to deploy**, select ***image_index*** (where *image_index* is the image index of the image that contains Windows 8.1, which was identified in the section Step 5-1: Import the Captured .wim File into Configuration Manager; for the purposes of this guide, select **2**). 2. Select **Next**.
Deployment Method	Select **Next**.
Client Package	1. In **Specify an existing ConfigMgr client package**, select **Browse**. 2. In the **Select a Package** dialog box, select **Microsoft Configuration Manager Client Upgrade**, and then select **OK**. 3. Select **Next**.

On this wizard page	Do this
USMT Package	1. In **Specify an existing USMT package**, select **Browse**. 2. In the **Select a Package** dialog box, select **USMT**, and then select **OK**. 3. Select **Next**.
Settings Package	1. Select **Create a new settings package**. 2. In **Package source folder to be created**, type **\\WDG-MDT-01\Packages$\CustomSettings_Target**, and then select **Next**.
Settings Package: Settings Details	1. In **Name**, type **MDT Target Computer Custom Settings**. 2. In **Version**, type **1.00**. 3. In **Comments**, type **Configuration settings for MDT deployment process (such as CustomSettings.ini) for the target computer**, and then select **Next**.
Sysprep Package	Select **Next**.
Summary	1. Review the information in the **Details** box that you provided while completing the previous wizard pages. 2. Select **Next**.
Progress	The progress for creating the task sequence is displayed.
Confirmation	Select **Finish**.

The list of task sequences is displayed. The task sequence that you just created (Windows 8.1 Target Deployment) is listed in the list of task sequences.

Step 5-3: Select the Distribution Points for the New Packages and Images

The Create MDT Task Sequence Wizard creates a number of packages and images. After these packages and images are created, select the distribution points from which the packages and images will be copied and available to target computers.

> **Note**
>
> In this sample, there is only one distribution point (WDG-MDT-01). However, most production networks have multiple distribution points. When performing this step in a production environment, select the appropriate distribution points for the network.

To select the distribution points for software distribution packages

1. Select **Start**, point to **All Programs**, and then point to **Microsoft System Center 2012**. Point to **Configuration Manager**, and then select **Configuration Manager Console**.

2. In the Configuration Manager console, in the navigation pane, select **Software Library**.

3. In the Software Library workspace, go to Overview/Operating Systems/Task Sequences.

4. In the preview pane, select **Windows 8.1 Target Deployment**.

5. On the Ribbon, on the **Home** tab, in the **Deployment** group, select **Distribute Content**.

   The Distribute Content Wizard starts.

6. Complete the Distribute Content Wizard using the following information. Accept default values unless otherwise specified.

Expand table

On this wizard page	Do this
General	Select **Next**.
Content	Select **Next**.
General: Content	Select **Next**.
General: Content Destination	1. Select **Add**, and then select **Distribution Point**. The **Add Distribution Points** dialog box appears. 2. In the **Add Distribution Points** dialog box, select **\\WDGMDT01.mdt2013.corp.woodgrovebank.com**, and then select **OK**. \\WDGMDT01.mdt2013.corp.woodgrovebank.com appears in the **Content destination** list. 3. Select **Next**.
Summary	1. Review the information in the **Details** box that you provided while completing the previous wizard pages. 2. Select **Next**.
Progress	The progress for distributing the software is displayed.
Completion	Select **Close**.

7. Close all open windows and dialog boxes.

Step 5-4: Customize the MDT Configuration Files

When the task sequence for the target computer has been created, customize the MDT configuration files that provide the configuration settings for deploying Windows 8.1 to the target computer—specifically, CustomSettings.ini.

When the CustomSettings.ini file has been customized, save the updated files to the source folder for the MDT Custom Settings package created earlier in the process (E:\Packages$\CustomSettings_Target).

To customize the MDT configuration files for the target computer

1. In Windows Explorer, go to E:\Packages$\CustomSettings_Target folder, and then double-click **CustomSettings.ini**.

2. Open Notepad, and then add the following lines to the CustomSettings.ini file:

   ini

   ```
   EventService=http://WDG-MDT-01:9800
   ```

   This setting will configure monitoring of the target computer deployment.

   > **Note**
   >
   > Make any other changes that are required by your environment.

   Example of the edited CustomSettings.ini File:

   ini

   ```
   [Settings]
   Priority=Default
   Properties=MyCustomProperty

   [Default]
   OSInstall=Y
   SkipCapture=YES
   SkipAdminPassword=NO
   SkipProductKey=YES
   EventService=http://WDG-MDT-01:9800
   ```

3. Save the file, and then close Notepad.

Step 5-5: Update the Distribution Points for the Custom Settings Package

When the source folder has been updated for the MDT Target Computer Custom Settings package in Configuration Manager, update the distribution points for the MDT Target Computer Custom Settings package. Updating the distribution points copies the updated version of the CustomSettings.ini file to the deployment shares specified in the package.

To update the distribution points for the Custom Settings package

1. Select **Start**, point to **All Programs**, and then point to **Microsoft System Center 2012**. Point to **Configuration Manager**, and then select **Configuration Manager Console**.

2. In the Configuration Manager console, in the navigation pane, select **Software Library**.

3. In the Software Library workspace, go to Overview/Application Management/Packages.

4. In the preview pane, select **MDT Target Computer Custom Settings**.

5. On the Ribbon, on the **Home** tab, in the **Deployment** group, select **Update Distribution Points**.

   The **Configuration Manager** dialog box opens, notifying you that you are going to update the package on all distribution points.

6. In the **Configuration Manager** dialog box, select **OK**.

7. Close all open windows and dialog boxes.

Step 5-6: Customize the Task Sequence for the Target Computer

For most deployments, the Windows 8.1 Target Deployment task sequence created earlier in the process performs all the necessary steps without modification. In this sample, modify the task sequence template to set the password for the local Administrator account to a known value. (By default, the task sequence sets the

password for the local Administrator account to a random value.) The task sequence may require further customization depending on the environment.

To customize the Windows 8.1 Target Deployment task sequence

1. Select **Start**, point to **All Programs**, and then point to **Microsoft System Center 2012**. Point to **Configuration Manager**, and then select **Configuration Manager Console**.
2. In the Configuration Manager console, in the navigation pane, select **Software Library**.
3. In the Software Library workspace, go to Overview/Operating Systems/Task Sequences.
4. In the preview pane, select **Windows 8.1 Reference Deployment**.
5. On the Ribbon, on the **Home** tab, in the **Task Sequence** group, select **Edit**.

   The **Windows 8.1 Reference Deployment Task Sequence Editor** dialog box opens.
6. In the **Windows 8.1 Reference Deployment Task Sequence Editor** dialog box, go to PostInstall/Apply Windows Settings.
7. On the **Properties** tab, select **Enable the account and specify the local administrator password**.
8. On the **Properties** tab, in **Password** and **Confirm Password**, type **P@ssw0rd**, and then select **Apply**.
9. Make any additional modifications to the task sequence that the environment requires, and then select **OK**.
10. Close all open windows and dialog boxes.

Step 6: Deploy the Captured Image of the Reference Computer to the Target Computer

When you have captured the image of the reference computer and created and configured the task sequence, deploy the captured image. Configure MDT to provide all the necessary configuration settings to deploy to the target computer. After initiating the deployment process, the image of the reference computer running Windows 8.1 is

automatically deployed to the target computer and configured with the settings defined.

Deploy the captured image by:

- Adding the target computer to the Configuration Manager site database as described in Step 6-1: Add the Target Computer to the Configuration Manager Site Database
- Creating a computer collection that includes the target computer as described in Step 6-2: Create a Computer Collection That Includes the Target Computer
- Deploying the task sequence created earlier in the process as described in Step 6-3: Deploy the Target Computer Task Sequence
- Starting the target computer with the task sequence bootable media as described in Step 6-4: Start the Target Computer with the Task Sequence Bootable Media

Step 6-1: Add the Target Computer to the Configuration Manager Site Database

To deploy an operating system without stand-alone media to a new computer that Configuration Manager does not currently manage, add the new computer to the Configuration Manager site database prior to initiating the operating system deployment process. Configuration Manager can automatically discover computers on the network that have a Windows operating system installed; however, if the computer has no operating system installed, use the Import Computer Information Wizard to import the new computer information.

To add the target computer to the Configuration Manager site database

1. Select **Start**, point to **All Programs**, and then point to **Microsoft System Center 2012**. Point to **Configuration Manager**, and then select **Configuration Manager Console**.
2. In the Configuration Manager console, in the navigation pane, select **Assets and Compliance**.
3. In the Assets and Compliance workspace, go to Overview/Devices.
4. On the Ribbon, on the **Home** tab, in the **Create** group, select **Import Computer Information**.

The Import Computer Information Wizard starts.

5. Complete the Import Computer Information Wizard using the following information. Accept the default values unless otherwise specified.

Expand table

On this wizard page	Do this
Select Source	Select **Import single computer**, and then select **Next**.
Select Source: Single Computer	1. In **Computer Name**, type **WDG-CLI-01**. 2. In **MAC address**, type ***mac_address*** (where *mac_address* is the MAC address of the primary network adapter for the target computer, WDG-CLI-01). 3. Select **Next**.
Select Source: Data Preview	Select **Next**.
Select Source: Choose Target Collection	Select **Next**.
Summary	1. Review the information in the **Details** box that you provided while completing the previous wizard pages. 2. Select **Next**.
Progress	The progress for importing the computer is displayed.
Confirmation	Select **Close**.

For more information on adding a new computer to the Configuration Manager site database, see the section, "To import computer information for a single computer," in the section, "How to Deploy Operating Systems in Configuration Manager," in the Configuration manager Documentation Library, which is installed with Configuration Manager.

Step 6-2: Create a Computer Collection That Includes the Target Computer

In the Configuration Manager console, create a collection that includes the target computer (WDG-CLI-01). You use this computer collection later when advertising the task sequence created earlier in the process.

To create a computer collection that includes the target computer

1. Select **Start**, point to **All Programs**, and then point to **Microsoft System Center 2012**. Point to **Configuration Manager**, and then select **Configuration Manager Console**.

2. In the Configuration Manager console, in the navigation pane, select **Assets and Compliance**.

3. In the Assets and Compliance workspace, go to Overview/Device Collections.

4. On the Ribbon, on the **Home** tab, in the **Create** group, select **Create Device Collection**.

   The Create Device Collection Wizard starts.

5. Complete the Create Device Collection Wizard using the following information. Accept the default values unless otherwise specified.

Expand table

On this wizard page	Do this
On this wizard page	**Do this**
General	a. In **Name**, type **Microsoft Deployment - Batch 01**. b. In **Comment**, type **Computers that are to be included in the first batch of computers deployed**. c. In **Limited Collection**, select **Browse**. The **Select Collection** dialog box appears. Complete the dialog box by performing the following steps: i. In the **Select Collection** dialog box, in **Name**, select **All Systems**. ii. Select **OK**. d. Select **Next**.
Membership Rules	a. Select **Add Rule**, and then select **Direct Rule**. The Create Direct Membership Rule Wizard starts. b. Complete the Create Direct Membership Rule Wizard by performing the following steps: i. On the **Welcome** page, select **Next**. ii. On the **Search for Resources** page, in **Resource class**, select **System Resource**; in **Attribute name**, select **Name**; in **Value**, type **WDG-CLI-01**; and then select **Next**.

On this wizard page	Do this
	iii. On the **Select Resources** page, select **WDG-CLI-01**, and then select **Next**. **Note:** The process for adding the target computer (WDG-CLI-01) to All Systems can take a few minutes to complete. If WDG-CLI-01 does not appear in the list, repeat steps b and c until WDGCLI01 appears. iv. On the **Summary** page, select **Next**. v. On the **Completion** page, select **Close**. c. Select **Next**.
Summary	1. Review the information in the **Details** box that you provided while completing the previous wizard pages. 2. Select **Next**.
Progress	The progress for creating the device collection is displayed.
Completion	Select **Close**.

For more information, see the section, "How to Create Collections in Configuration Manager," in the Configuration manager Documentation Library, which is installed with Configuration Manager.

Step 6-3: Deploy the Target Computer Task Sequence

In the Configuration Manager console, deploy the task sequence created earlier in the process for the target computers. Deploy the task sequence to the collection of target computers created earlier in the process.

To deploy the task sequence

1. Select **Start**, point to **All Programs**, and then point to **Microsoft System Center 2012**. Point to **Configuration Manager**, and then select **Configuration Manager Console**.

2. In the Configuration Manager console, in the navigation pane, select **Software Library**.

3. In the Software Library workspace, go to Overview/Operating Systems/Task Sequences.

4. In the preview pane, select **Windows 8.1 Target Deployment**.

5. On the Ribbon, on the **Home** tab, in the **Deployment** group, select **Deploy**.

The Deploy Software Wizard starts.

6. Complete the Deploy Software Wizard using the following information. Accept the default values unless otherwise specified.

Expand table

On this wizard page	Do this
General	1. In **Collection**, select **Browse**. 2. In the **Browse Collection** dialog box, select **Microsoft Deployment - Batch 01**, and then select **OK**. 3. In **Comment**, type **Deploy Windows 8.1 to the first batch of target computers**. 4. Select **Next**.
Deployment Settings	1. In **Purpose**, select **Available**. 2. Select the **Make available to boot media and PXE** check box. 3. Select **Next**.
Deployment Settings: Schedule	Select **Next**.
Deployment Settings: User Experience	Select **Next**.
Deployment Settings: Alerts	Select **Next**.
Deployment Settings: Distribution Points	Select **Next**.
Summary	1. Review the information in the **Details** box that you provided while completing the previous wizard pages. 2. Select **Next**.
Progress	The progress for deploying the task sequence is displayed.
Completion	Select **Close**.

For more information, see the section, "How to Deploy a Task Sequence," in the Configuration manager Documentation Library, which is installed with Configuration Manager.

Step 6-4: Start the Target Computer with the Task Sequence Bootable Media

Start the target computer (WDG-CLI-01) with the task sequence bootable media created earlier in the process. This medium starts Windows PE on the reference computer and initiates the MDT process. At the end of the MDT process, Windows 8.1 is deployed on the target computer.

> **Note**
>
> You can also initiate the MDT process by starting the target computer from Windows Deployment Services.

To start the target computer with the task sequence bootable media

1. Start WDG-CLI-01 with the task sequence bootable media created earlier in the process.

   Windows PE starts, and then the Task Sequence Wizard starts.

2. Complete the Task Sequence Wizard using the following information. Accept the default values unless otherwise specified.

Expand table

On this wizard page	Do this
Welcome to the Task Sequence Wizard	In **Password**, type **P@ssw0rd**, and then select **Next**.
Select a Task Sequence	In the list box, select **Windows 8.1 Target Deployment**, and then select **Next**.

To monitor the reference computer deployment process using the Deployment Workbench

1. On WDG-MDT-01, select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.

2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/MDT Deployment Share (C:\DeploymentShare$)/Monitoring.

3. In the details pane, view the deployment process for **WDG-CLI-01**.

4. In the Actions pane, periodically select **Refresh**.

   The status of the deployment process is updated in the details pane. Continue to monitor the deployment process until the process is complete.

5. In the details pane, select **WDG-CLI-01**.

6. In the Actions pane, select **Properties**.

   The **WDG-CLI-01 Properties** dialog box is displayed.

7. In the **WDG-CLI-01 Properties** dialog box, on the **Identity** tab, view the monitoring information provided about the deployment process as described in the following table:

Expand table

Information	Description
ID	Unique identifier for the computer being deployed.
Computer Name	The name of the computer being deployed.
Deployment status	The current status of the computer being deployed; the status can be one of the following: - **Running**. The task sequence is healthy and running. - **Failed**. The task sequence failed, and the deployment process was unsuccessful. - **Completed**. The task sequence has finished. - **Unresponsive**. The task sequence has not updated its status in the past four hours and is assumed to be nonresponsive.
Step	The current task sequence step being run.
Progress	The overall progress of the task sequence. The progress bar indicates how many task sequence steps have been run out of the total number of task sequence steps.
Start	The time the deployment process started.
End	The time the deployment process ended.
Elapsed	The length of time the deployment process has been running or took to run if the deployment process has finished.
Errors	The number of errors encountered during the deployment process.
Warnings	The number of warnings encountered during the deployment process.

Information	Description
Remote Desktop	This button allows you to establish a remote desktop connection with the computer being deployed using the Windows Remote Desktop feature. This method assumes that: - The target operating system is running and has remote desktop support enabled - **mstsc.exe** is in the path **Note:** This button is always visible but may not be able to establish a remote desktop session if the monitored computer is running Windows PE, has not completed installation of the target operating system, or does not have the Remote Desktop feature enabled.
VM Connection	This button allows you to establish a remote desktop connection to a VM running in Hyper-V. This method assumes that: - The deployment is being performed to a VM running on Hyper-V - **vmconnect.exe** is located in the %ProgramFiles%\Hyper-V folder **Note:** This button appears when ZTIGather.wsf detects that Hyper-V integration components are running on the monitored computer. Otherwise, this button will not be visible.
DaRT Remote Control	This button allows you to establish a remote control session using the remote viewer feature in the Diagnostics and Recovery Toolkit (DaRT). This method assumes that: - DaRT has been deployed to the target computer and is currently running - **DartRemoteViewer.exe** is located in the %ProgramFiles%\Microsoft DaRT 7\v7 folder **Note:** This button appears when ZTIGather.wsf detects that DaRT is running on the monitored computer. Otherwise, this button will not be visible.
Automatically refresh this information every 10 seconds	Check box that controls whether the information in the dialog box is automatically refreshed. If the check box is: - Selected, the information is refreshed every 10 seconds - Cleared, the information is not automatically refreshed and must be manually refreshed using the **Refresh Now** button
Refresh Now	This button immediately refreshes the information displayed in the dialog box.

8. In the **WDG-REF-01 Properties** dialog box, select **OK**.

9. Close the Deployment Workbench.

To monitor the reference computer deployment process using the Get-MDTMonitorData cmdlet

1. On WDG-MDT-01, select **Start**, point to **Administrative Tools**, and then select **Windows PowerShell Modules**.The Windows PowerShell Modules command prompt opens.

2. Create a PowerShell drive that uses the MDT PowerShell provider by running the New-PSDrive cmdlet as shown in the following example:

```
PowerShell

New-PSDrive -Name DS001 -PSProvider mdtprovider -Root
d:\DeploymentShare$
```

3. View the MDT monitoring process by running the **Get-MDTMonitorData** cmdlet as shown in the following example:

```
PowerShell

Get-MDTMonitorData -Path DS001:
```

   This command returns the monitoring data collected by the MDT monitoring service running on the same computer that hosts the deployment share as shown in the following example output:

```
PowerShell

Name               : WDG-REF-01
PercentComplete    : 96
Settings           :
Warnings           : 0
Errors             : 0
DeploymentStatus   : 1
StartTime          : 6/7/2012 6:45:39 PM
EndTime            :
ID                 : 1
UniqueID           : 94a0830e-f2bb-421c-b1e0-6f86f9eb9fa1
CurrentStep        : 130
TotalSteps         : 134
StepName           : Gather
LastTime           : 6/7/2012 8:46:32 PM
DartIP             :
DartPort           :
DartTicket         :
VMHost             : XYL-DC-02
VMName             : WDG-REF-01
ComputerIdentities : {}
```

```
Name                : WDG-CLI-01
PercentComplete     : 26
Settings            :
Warnings            : 0
Errors              : 0
DeploymentStatus    : 1
StartTime           : 6/7/2012 3:07:13 AM
EndTime             :
ID                  : 2
UniqueID            : 94a0830e-f2bb-421c-b1e0-6f86f9eb9fa1
CurrentStep         : 49
TotalSteps          : 134
StepName            : Capture Network Settings using MDT
LastTime            : 6/7/2012 3:08:32 AM
DartIP              :
DartPort            :
DartTicket          :
VMHost              :
VMName              :
ComputerIdentities : {}
```

4. Close the Windows PowerShell console.

   If any problems occur during the deployment, consult the MDT document *Troubleshooting Reference*. When successfully completed, the target computer is running a Windows 8.1 operating system configured like the reference computer.

Feedback

Was this page helpful? Yes No

Provide product feedback

Quick Start Guide for User-Driven Installation

Article • 02/12/2024

Microsoft Deployment Toolkit (MDT) 2013 provides technology for deploying Windows operating systems, and Microsoft Office. This quick start guide helps you quickly evaluate MDT 2013 by providing condensed, step-by-step instructions for using it to install the Windows 8.1 operating system and Microsoft Office Professional Plus 2010 with User-Driven Installation (UDI) and Microsoft System Center 2012 R2 Configuration Manager. This quick start guide demonstrates how to perform the MDT New Computer deployment scenario, which covers the deployment of Windows 8.1 to a new computer. This scenario assumes that there is no user data or profile to preserve.

> ⓘ **Note**
>
> In this document, *Windows* applies to the Windows 8.1, Windows 8, Windows 7, Windows Server® 2012 R2, Windows Server 2012, and Windows Server 2008 R2 operating systems unless otherwise noted. MDT does not support ARM processor-based versions of Windows. Similarly, *MDT* refers to MDT 2013 unless otherwise stated.

After using this guide to evaluate MDT, review the rest of the MDT guidance to learn more about the technology's advanced features.

> ⓘ **Note**
>
> The infrastructure setup described here is for evaluation purposes and not intended for a production system.

Prerequisites

UDI installations using System Center 2012 R2 Configuration Manager have the following prerequisites.

Required Software

To complete this guide, the following software is required:

- Windows Server 2008 R2
- Microsoft SQL Server® 2008 R2
- SQL Server 2008 R2 Service Pack 1 (SP1)
- SQL Server 2008 R2 SP1 Cumulative Update 6 (CU6)
- Windows 8.1
- System Center 2012 R2 Configuration Manager
- Office Professional Plus 2010 volume license, 32-bit version
- Microsoft .NET Framework version 3.5 with SP1
- Windows PowerShell™ version 2.0
- Windows Preinstallation Environment (Windows PE), which is included in Configuration Manager
- Networking services, including Domain Name System (DNS) and Dynamic Host Configuration Protocol (DHCP)
- Active Directory® Domain Services (AD DS)

> ⓘ **Note**
>
> The Task Sequencer used in MDT deployments requires that the Create Global Object right be assigned to credentials used to access and run the Deployment Workbench and the deployment process. This right is normally available to accounts with Administrator-level permissions (unless explicitly removed). Also, the Specialized Security - Limited Functionality (SSLF) security profile removes the Create Global Object right and should not be applied to computers deployed using MDT.

Computer Configuration

To complete this guide, set up the computers listed in Table 1. These computers can be either physical computers or virtual machines (VMs) with the system resources designated.

Table 1. Computers Used in This Guide

Expand table

Computer	Description and system resources
WDG-MDT-01	This computer runs the MDT infrastructure and Configuration Manager. The computer runs Windows Server 2008 R2 with the following networking services installed: - AD DS - DNS Server - DHCP Server - Windows Deployment Services The system resources of the computer are as follows: - Quad-core processor running at 2.66 gigahertz (GHz) or faster - 4 gigabytes (GB) or more of physical memory - A disk partition that has 40 GB or more of available disk space; it will become the drive C partition - One CD-ROM or DVD-ROM drive that will be assigned the drive letter D - A disk partition that has 40 GB or more of available disk space; it will become partition E.
WDG-REF-01	This is the reference computer, which runs no current operating system. The system resources of the computer are as follows: - Processor running at 1.4 GHz or faster - 1 GB or more of physical memory - 16 GB or more of available disk space
WDG-CLI-01	This is the target computer, which runs no current operating system. The system resources of the computer are as follows: - Processor running at 1.4 GHz or faster - 1 GB or more of physical memory - 16 GB or more of available disk space

The resources listed in Table 1 reflect the system resources recommended to perform the steps in this guide. For information on the minimum system resource requirements for:

- Windows Server 2008 R2, see Installing Windows Server 2008 R2
- SQL Server 2008 R2, see Hardware and Software Requirements for Installing SQL Server 2008 R2

> **Note**

This guide assumes that MDT is being evaluated on 64-bit (x64) physical or virtual computers. If evaluating MDT on 32-bit (x86) platforms, download and install the x86 editions of MDT and the components that this guide describes.

Step 1: Prepare the Prerequisite Infrastructure

For purposes of this guide, all the prerequisite infrastructure services run on the computer named *WDG-MDT-01*. Install the prerequisite software, server roles, and services on this computer before installing MDT.

> **Note**
>
> This section assumes that you are creating a new Configuration Manager infrastructure for MDT. If you are using an existing Configuration Manager infrastructure, review the steps in this section and substitute existing resource names for the resources created in this section (such as the computer name and shared network folders). After reviewing this section, proceed to **Step 2: Prepare the MDT Environment**

Prepare the prerequisite infrastructure before installing MDT by:

- Installing Windows Server 2008 R2 as described in Step 1-1: Install Windows Server 2008 R2
- Creating the required folders and network shares as described in Step 1-2: Create the Required Folders and Network Shares
- Obtaining the software required to perform the steps in this guide as described in Step 1-3: Obtain the Required Software
- Installing the AD DS server role as described in Step 1-4: Install the AD DS Server Role
- Installing the DHCP Server server role as described in Step 1-5: Install the DHCP Server Server Role
- Installing the Web Services (IIS) server role as described in Step 1-6: Install the Web Services (IIS) Server Role
- Adding the required Windows Server 2008 R2 features as described in Step 1-7: Add the Required Windows Server 2008 R2 Features

- Creating the user and service accounts required to perform the steps in this guide as described in Step 1-8: Create the Required User and Service Accounts
- Installing SQL Server 2008 R2 for Configuration Manager to use as described in Step 1-9: Install SQL Server 2008 R2
- Adding the site server to the Administrators security group as described in Step 1-10: Add the Site Server to the Administrators Security Group
- Installing Configuration Manager as described in Step 1-11: Install Configuration Manager
- Configuring the Network Access Account that Configuration Manager clients use to access Configuration Manager distribution points as described in Step 1-12: Configure the Network Access Account
- Configuring the Configuration Manager site boundaries and boundary groups as described in Step 1-13: Configure the Configuration Manager Site Boundaries and Boundary Groups
- Configuring the publishing of site information in AD DS and DNS as described in Step 1-14: Configure the Publishing of Site Information in AD DS and DNS
- Configuring discovery of users in AD DS as described in Step 1-15: Configure Discovery of Active Directory Users

Step 1-1: Install Windows Server 2008 R2

Use the information in 2 to install Windows Server 2008 R2. Accept default values unless otherwise specified.

Table 2. Information for Installing Windows Server 2008 R2

Expand table

When prompted for	Provide these values
Where do you want to install Windows?	**Disk 0 Unallocated Space**
Password	Any strong password
Computer name	**WDG-MDT-01**

When prompted for	Provide these values
Format for volumes C and E	**NTFS**
TCP/IP configuration	Configure with a static IP address configuration, with the other TCP/IP configuration options as appropriate for the environment

Step 1-2: Create the Required Folders and Network Shares

The MDT deployment process requires additional folders that are used as the source for files or to store files created during the MDT deployment process. Some of these folders need to be shared so that they can be accessed from other computers.

To create the required folders and shares

1. Create the folders and shares listed in Table 3 with the permissions specified for each share

Table 3. Folders That the MDT Deployment Process Requires

Expand table

Create this folder	With this share name	With these share permissions
E:\Source$	Source$	**Administrators**: Co-owner **Everyone**: Read
E:\Images$	Images$	**Administrators**: Co-owner **Everyone**: Read
E:\Capture$	Capture$	**Administrators**: Co-owner **Everyone**: Read
E:\Packages$	Packages$	**Administrators**: Co-owner **Everyone**: Read

2. Create the following folders:
    - E:\CMDownloads

- E:\Source$\CustomSettings
- E:\Source$\Drivers
- E:Source$Windows_8-1
- E:Source$MDT_2013
- E:Source$SQL2008R2
- E:Source$SQL2008R2SP1
- E:Source$SQL2008R2CU6
- E:Source$OfficeProPlus2010
- E:Source$ConfigMgr
- E:Packages$Drivers

3. Copy the device drivers for the reference computer (WDG-REF-01) and the target computer (WDG-CLI-01) to E:\Source$\Drivers.

> **Note**
>
> The processes in this guide assume that the reference computer and target computer have the same devices and do not require different devices drivers.

Step 1-3: Obtain the Required Software

Besides Windows Server 2008 R2, Windows 8.1, and System Center 2012 R2 Configuration Manager, certain software is required to evaluate MDT based on the processes in this guide. Table 4 lists the software required to perform deployments using MDT, where to obtain the software, and where to place the software on WDG-MDT-01.

Table 4. Additional Software Required for Deployment Using MDT

Expand table

Obtain this software	Place in this folder
MDT 2013	E:\Source$\MDT_2013

Obtain this software	Place in this folder
Windows 8.1 distribution files from the product media	E:\Source$\Windows_8-1
Device drivers required for the reference and target computers (WDG-REF-01 and WDG-CLI-01)	E:\Source$\Drivers
SQL Server 2008 R2 from the product media	E:\Source$\SQL2008
SQL Server 2008 R2 SP1, available at https://www.microsoft.com/download/details.aspx?id=26113	E:\Source$\SQL2008R2SP1
SQL Server 2008 R2 SP1 CU6, available at https://support.microsoft.com/kb/2679367	E:\Source$\SQL2008R2SP1CU6
System Center 2012 R2 Configuration Manager from the product media	E:\Source$\ConfigMgr
Office Professional Plus 2010 32-bit Volume Licensing version from the product media	E:\ Source$\OfficeProPlus2010

Step 1-4: Install the AD DS Server Role

AD DS is required to provide authentication and act as a repository for configuration values for the Microsoft products and technologies that MDT uses, such as Microsoft SQL Server and Configuration Manager.

To install AD DS, run the DCPROMO Wizard to configure the computer as a domain controller. Install AD DS using the information in Table 5, accepting any defaults unless otherwise specified.

Table 5. Information for Installing AD DS

Expand table

When prompted	Do this
For the domain type	Create a new domain in a new forest.
For the fully qualified domain name	Type **mdt2013.corp.woodgrovebank.com**.
For the forest functional level	Select **Windows Server 2008 R2.**
To install the DNS Server service as part of the domain controller installation process	Select **Yes**.

Step 1-5: Install the DHCP Server Server Role

The DHCP Server server role is required to provide automatic IP configuration for the target computers. Install DHCP Server using the information in Table 6, accepting any defaults unless otherwise specified.

> **Note**
>
> If you are using a virtualized environment, disable any DHCP configuration that the computer-virtualization software provides. Ensure that the DHCP Server service running WDG-MDT-01 is the only provider of IP configuration using DHCP.

Table 6. Information for Installing the DHCP Server Server Role

Expand table

On this wizard page	Do this
Authorize DHCP server in Active Directory	Authorize WDG-MDT-01 to provide client IP configuration.
DHCP scopes	Create an appropriate scope that can be used to automatically configure TCP/IP for WDG-REF-01 and WDG-CLI-01.
DHCPv6 stateless mode configuration	Disable DHCPv6 stateless mode for this server.

Step 1-6: Install the Web Services (IIS) Server Role

Install the Web Services (IIS) server role with the role services listed in Table 7, which are required for SQL Server 2008 R2 and Configuration Manager. Unless otherwise specified, use the default values.

Table 7. Information for Installing the Web Services (IIS) Server Role

Expand table

Role service	Status
Web Server	Installed
Common HTTP Features	Installed
Static Content	Installed
Default Document	Installed
Directory Browsing	Installed
HTTP Errors	Installed
HTTP Redirection	Installed
WebDAV Publishing	Installed
Application Development	Installed
ASP.NET	Installed
.NET Extensibility	Installed
ASP	Not installed
CGI	Not installed
ISAPI Extensions	Installed
ISAPI Filters	Installed
Server Side Includes	Not installed
Health and Diagnostics	Installed
HTTP Logging	Installed
Logging Tools	Installed
Request Monitor	Installed
Tracing	Installed
Custom Logging	Not installed
ODBC Logging	Not installed
Security	Installed
Basic Authentication	Not installed
Windows Authentication	Installed

Role service	Status
Digest Authentication	Not installed
Client Certificate Mapping Authentication	Not installed
IIS Client Certificate Mapping Authentication	Not installed
URL Authorization	Not installed
Request Filtering	Installed
IP and Domain Restriction	Not installed
Performance	Installed
Static Content Compression	Installed
Dynamic Content Compression	Not installed
Management Tools	Installed
IIS Management Console	Installed
IIS Management Scripts and Tools	Not installed
Management Service	Not installed
IIS 6 Management Compatibility	Installed
IIS 6 Metabase Compatibility	Installed
IIS 6 WMI Compatibility	Installed
IIS 6 Scripting Tools	Not installed
IIS 6 Management Console	Not installed
FTP Publishing Service	Not installed
FTP Server	Not installed
FTP Management Console	Not installed
IIS Hostable Web Core	Not installed

Step 1-7: Add the Required Windows Server 2008 R2 Features

In addition to installing the required Windows Server 2008 R2 server roles, add the following required features in Server Manager in the **Features Summary** section:

- Background Intelligent Transfer Service
- Remote Differential Compression

Step 1-8: Create the Required User and Service Accounts

Configuration Manager and SQL Server 2008 R2 require user accounts during the installation process. Table 8 lists the information needed for creating these accounts.

Table 8. Information for Creating the Required Accounts

Expand table

Create this account	With these settings
SQL Server Agent service account	1. In **First name**, type **SQL Agent**. 2. In **Last name**, type **Service Account**. 3. In **User logon name**, type **SQLAgent**. 4. In **Password** and **Confirm password**, type **P@ssw0rd**. 5. Clear the **User must change password at next logon** check box. 6. Select the **Password never expires** check box. 7. Make the account a member of the Domain Admins security group. 8. In **Description**, type **Service account used to run SQL Server 2008 R2 Agent service**.
SQL Server Database Engine service account	1. In **First name**, type **SQL DB Engine**. 2. In **Last name**, type **Service Account**. 3. In **User logon name**, type **SQLDBEngine**. 4. In **Password** and **Confirm password**, type **P@ssw0rd**. 5. Clear the **User must change password at next logon** check box. 6. Select the **Password never expires** check box. 7. Make the account a member of the Domain Admins security group. 8. In **Description**, type **Service account used to run SQL Server 2008 R2 database engine**.
SQL Server Reporting Services service account	1. In **First name**, type **SQL Reporting**. 2. **In Last name**, type **Service Account**. 3. In **User logon name**, type **SQLReport**. 4. In **Password** and **Confirm password**, type **P@ssw0rd**. 5. Clear the **User must change password at next logon** check box. 6. Select the **Password never expires** check box. 7. Make the account a member of the Domain Admins

Create this account	With these settings
	security group. 8. In **Description**, type **Service account used to run SQL Server 2008 R2 reporting services**.
Configuration Manager Client Network Access account	1. In **First name**, type **CM 2012**. 2. In **Last name**, type **Client Network Access**. 3. In **User logon** name, type **CMNetAccess**. 4. In **Password** and **Confirm password**, type **P@ssw0rd**. 5. Clear the **User must change password at next logon** check box. 6. Select the **Password never expires** check box. 7. In **Description**, type **Service account used as the network access account for Configuration Manager Client**.

Step 1-9: Install SQL Server 2008 R2

Before installing Configuration Manager, install SQL Server 2008 R2 SP1 and CU6.

> **Note**
>
> To enable all SQL Server 2008 R2 features, install the Web Services (IIS) server role before installing SQL Server 2008 R2.

To install SQL Server 2008 R2

1. Start the SQL Server Installation Center.
2. In the SQL Server Installation Center, in the navigation pane, select **Installation**.
3. In the preview pane, select **New installation or add features to an existing installation**.

   SQL Server 2008 R2 Setup Wizard starts.
4. Install SQL Server 2008 R2 using the information in Table 9, accepting the defaults unless otherwise specified.

Table 9. Information for Installing SQL Server 2008 R2

Expand table

On this wizard page	Do this
Setup Support Rules	Select **OK**.
Product Key	Select **Next**.
License Terms	Select the **I accept the license terms** check box, and then select **Next**.
Setup Support Files	Select **Install**.
Setup Support Rules	Ensure that no critical results exist for the rules, and then select **Next**.
Setup Role	Select **SQL Server Feature Installation**, and then select **Next**.
Feature Selection	1. Select the **Database Engine Services** check box. 2. Select the **Reporting Services** check box. 3. Select the **Full-Text Search** check box. 4. Select the **Management Tools - Complete** check box. 5. Select **Next**.
Installation Rules	Select **Next**.
Instance Configuration	Select **Next**.
Disk Space Requirements	Select **Next**.
Server Configuration	1. For **SQL Server Agent**, in **Account Name**, type **MDT2013\SQLAgent**, in **Password**, type **P@ssw0rd**. 2. For **SQL Server Database Engine**, in **Account Name**, type **MDT2013\SQLDBEngine**, in **Password**, type **P@ssw0rd**. 3. For **SQL Server Reporting Services**, in **Account Name**, type **MDT2013\SQLReport**, in **Password**, type **P@ssw0rd**. 4. Select **Next**.
Database Engine Configuration	Select **Add Current User**, and then select **Next**.
Reporting Services Configuration	Select **Next**.
Error Reporting	Select **Next**.
Installation Configuration Rules	Select **Next**.
Ready to Install	Select **Install**.
Complete	Select **Close**.

5. Close the SQL Server Installation Center.

To install SQL Server 2008 R2 SP1

6. In Windows Explorer, go to E:\Source$\SQL2008R2SP1, and double-click **SQLServer2008R2SP1-KB2528583-x64-ENU.exe**.

   The **Extracting Files** dialog box displays the file-extraction process. When the process is complete, the SQL Server 2008 R2 Service Pack 1 Update Setup Wizard starts.

7. Install SQL Server 2008 R2 SP1 using the information in Table 10, accepting the defaults unless otherwise specified.

Table 10. Information for Installing SQL Server 2008 R2 SP1

Expand table

On this wizard page	Do this
SQL Server 2008 R2 update	Select **Next**.
License Terms	Select the **I accept the license terms** check box, and then select **Next**.
Select Features	Select **Next**.
Check Files In Use	Select **Next**.
Ready to update	Select **Update**.
Update Progress	The progress is displayed on the wizard page as the update is performed and finishes.
Complete	Select **Close**.

To install SQL Server 2008 R2 SP1 CU6

8. In Windows Explorer, go to E:\Source$\SQL2008R2SP1CU6, and double-click **446622_intl_x64_zip.exe**.

   The **Microsoft Self-Extractor** dialog box appears.

9. In the **Microsoft Self-Extractor** dialog box, select **Continue**.

10. In the **Microsoft Self-Extractor** dialog box, in **Select the folder where you want to unzip the files to**, type **E:\Source$\SQL2008R2SP1CU6**, and then select **OK**.

    > **Note**
    >
    > You can select the ellipsis (...) to browse for the E:\Source$\SQL2008R2SP1CU6 folder.

    The extraction process is displayed. When the process is complete, the completion status is displayed.

11. In the **Microsoft Self-Extractor** dialog box, select **OK**.

12. In Windows Explorer, go to E:\Source$\SQL2008R2SP1CU6, and double-click **SQLServer2008R2- KB2679367-x64.exe**.

    The **Extracting Files** dialog box displays the file-extraction process. When the process is complete, the SQL Server 2008 R2 Service Pack 1 CU6 Update Setup Wizard starts.

13. Install SQL Server 2008 R2 SP1 CU6 using the information in Table 11, accepting the defaults unless otherwise specified.

Table 11. Information for Installing SQL Server 2008 R2 SP1 CU6

Expand table

On this wizard page	Do this
SQL Server 2008 R2 update	Select **Next**.
License Terms	Select the **I accept the license terms** check box, and then select **Next**.
Select Features	Select **Next**.
Check Files In Use	Select **Next**.
Ready to update	Select **Update**.
Update Progress	The progress is displayed on the wizard page as the update is performed and finishes.

On this wizard page	Do this
Complete	Select **Close**.

The **Install a SQL Server 2008 R2 update** dialog box appears prompting you to restart the computer to complete the setup.

14. In the **Install a SQL Server 2008 R2 update** dialog box, select **OK**.
15. Restart the computer.
16. After installing SQL Server 2008 R2 SP1 CU6, the SQL Server build number should be 10.51.2811.0.

> **Tip**
>
> You can verify the SQL Server build number by viewing the SQL Server updates applied in the Programs and Features Control Panel item by selecting **View installed updates**.

Step 1-10: Add the Site Server to the Administrators Security Group

When all computers are in the same forest, manually add the site server computer account to the local Administrators group on each computer. Complete this step before configuring the computer as a site system.

To add the site server to the Administrators security group

1. Select **Start**, point to **Administrative Tools**, and then select **Active Directory Users and Computers**.
2. In the Active Directory Users and Computers console tree, go to mdt2013.corp.woodgrovebank.com/Builtin.
3. In the preview pane, right-click **Administrators**, and then select **Properties**.
4. In the **Administrators Properties** dialog box, select the **Members** tab, and then select **Add**.
5. In the **Select Users, Contacts, Computers, or Groups** dialog box, select **Object Types**.

6. In the **Object Types** dialog box, in **Object types**, select **Computers**, and then select **OK**.

7. In the **Select Users, Contacts, Computers, or Groups** dialog box, in **Enter the object names to select**, type **WDG-MDT-01**. Select **Check Names**, and then select **OK**.

8. Close any open windows.

Step 1-11: Install Configuration Manager

When the other products and technologies have been installed, install Configuration Manager. Before doing so, however, extend the Active Directory schema so that computers can locate the distribution points, service locator points, and other server roles. Also, you can extend the schema after you have installed Configuration Manager. For more information about how to extend the Active Directory schema for Configuration Manager, see the section, "Extend the Active Directory Schema," in the Configuration Manager Documentation Library, which is installed with Configuration Manager.

After extending the Active Directory schema, install Configuration Manager. The configuration of WDG-MDT-01 supports Configuration Manager for this sample. The configuration of computers in the production network may vary. To find out more about the prerequisites for installing Configuration Manager, see Supported Configurations for Configuration Manager.

To install Configuration Manager

1. Start the System Center 2012 R2 Configuration Manager Setup splash screen.

2. On the System Center 2012 R2 Configuration Manager Setup splash screen, select the **Install** link.

   The Microsoft System Center 2012 R2 Configuration Manager Setup Wizard starts.

3. Complete the Microsoft System Center 2012 R2 Configuration Manager Setup Wizard using the information in Table 12. Accept the defaults unless otherwise specified.

Table 12. Information for Installing Configuration Manager

Expand table

On this wizard page	Do this
Before You Begin	Select **Next**.
Getting Started	Select **Next**.
Product Key	In **Enter your 25-character product key**, type ***product_key*** (where *product_key* is your product key for Configuration Manager).
Microsoft Software License Terms	Select the **I accept these license terms** check box, and then select **Next**.
Update Prerequisite Components	In **Download and use the latest updates. Updates will be saved to the following location**, type **E:\CMDownloads**, and then select **Next**.
Server Language Selection	Select **Next**.
Client Language Selection	Select **Next**.
Site and Installation Settings	1. In **Site code**, type **NYC**. 2. In **Site name**, type **New York City Site**. 3. Select **Next**.
Primary Site Installation	1. Select **Install the primary site as a stand-alone site**. 2. Select **Next**. The **Configuration Manager** dialog box appears, confirming that you want to install this site as a stand-alone site. 3. In the **Configuration Manager** dialog box, select **Yes**.
Database Information	Select **Next**.
SMS Provider Settings	Select **Next**.
Client Computer Communication Settings	Select **Configure the communication method on each site system role**, and then select **Next**.
Site System Roles	Select **Next**.
Customer Experience Improvement Program Configuration	Select the appropriate participation in the Customer Experience Improvement program for your organization, and then select **Next**.
Settings Summary	Select **Next**.
Prerequisite Check	Select **Begin Install**.

On this wizard page	Do this
Install	Monitor the installation process until it is complete, and then select **Close**.

4. Close all open windows and dialog boxes.

   When the wizard is complete, Configuration Manager is installed.

Step 1-12: Configure the Network Access Account

The Configuration Manager client needs an account to provide credentials when accessing the Configuration Manager distribution points, MDT deployment shares, and shared folders. This account is called the *Network Access account*. The CMNetAccess account was created earlier in the process to use as the Network Access account.

To configure the Network Access account

1. Select **Start**, point to **All Programs**, and then point to **Microsoft System Center 2012**. Point to **Configuration Manager**, and then select **Configuration Manager Console**.

2. In the Configuration Manager console, in the navigation pane, select **Administration**.

3. In the Administration workspace, go to Overview/Site Configuration/Sites.

4. In the preview pane, select **NYC - New York City Site**.

5. On the Ribbon, select **Settings**, select **Configure Site Components**, and then select **Software Distribution**.

6. In the **Software Distribution Properties** dialog box, select the **Network Access Account** tab.

7. In **Network Access Account**, select **Specify the account that accessed network locations**, select **Set**, and then select **New Account**.

   The **Windows User Account** dialog box appears.

8. Complete the **Windows User Account** dialog box using the information in Table 13, and then select **OK**.

Table 13. Information Required to Complete the Windows User Account Dialog Box

Expand table

For this	Do this
User name	Type **MDT2013\CMNetAccess**.
Password	Type **P@ssw0rd**.
Confirm password	Type **P@ssw0rd**.

9. In the **Software Distribution Properties** dialog box, select **OK**.
10. Close any open windows.

Step 1-13: Configure the Configuration Manager Site Boundaries and Boundary Groups

The Configuration Manager client needs to know the boundaries for the site. Unless the site boundaries are specified, the client assumes that the computer running Configuration Manager is in a remote site. Add a site boundary based on the IP subnet that WDG-MDT-01, WDG-REF-01, and WDG-CLI-01 use. Then, add the site boundary to a site boundary group.

To create a Configuration Manager site boundary

1. Select **Start**, point to **All Programs**, and then point to **Microsoft System Center 2012**. Point to **Configuration Manager**, and then select **Configuration Manger Console**.
2. In the Configuration Manager console, in the navigation pane, select **Administration**.
3. In the Administration workspace, go to Overview/Hierarchy Configuration/Boundaries.
4. On the Ribbon, select **Create Boundary**.

   The **Create Boundary** dialog box opens.
5. Complete the **Create Boundary** dialog box using the information in Table 14, and then select **OK**.

   > ⓘ Note

For this sample, the site boundary is specified by network address. However, you can also specify site boundaries using an AD DS site name or an IP address range.

Table 14. Information Required to Complete the Create Boundary Dialog Box

Expand table

For this	Do this
Description	Type **IP Subnet Boundary**.
Type	Select **IP subnet**.
Network	Type ***network_address*** (where *network_address* is the network address of the subnet where the computers are installed).
Subnet mask	Type ***subnet_mask*** (where *subnet_mask* is the subnet mask of the subnet where the computers are installed).

To add the Configuration Manager site boundary to a site boundary group

6. In the Configuration Manager console, in the navigation pane, select **Administration**.
7. In the Administration workspace, go to Overview/Hierarchy Configuration/Boundary Groups.
8. On the Ribbon, select **Create Boundary Group**.

   The **Create Boundary Group** dialog box opens.
9. Complete the **General** tab of the **Create Boundary Group** dialog box using the information in Table 15.

Table ARABIC 15. Information Required to Complete the General Tab of the Create Boundary Group Dialog Box

Expand table

For this	Do this
Name	Type **New York City Boundary Group**.
Description	Type **Boundary group for the site boundaries at the New York City site**.
Boundaries	1. Select **Add**. The **Add Boundaries** dialog box appears. 2. In the **Add Boundaries** dialog box, select ***site_boundary*** (where *site_boundary* is the site boundary you created earlier in the process), and then select **OK**. The site boundary appears in the list of boundaries.

10. Complete the **References** tab of the **Create Boundary Group** dialog box using the information in Table 16, and then select **OK**.

Table 16. Information Required to Complete the References Tab of the Create Boundary Group Dialog Box

Expand table

For this	Do this
Site assignment	Select the **Use this boundary group for site assignment** check box.
Content location	1. Select **Add**. The **Add Site Systems** dialog box appears. 2. In the **Add Site Systems** dialog box, select **\\WDG-MDT-01.mdt2013.corp.woodgrovebank.com**, and then select **OK**. The site system server appears in the list of site system servers.

11. Close any open windows.

Step 1-14: Configure the Publishing of Site Information in AD DS and DNS

The Configuration Manager client needs to locate the various Configuration Manager server roles. Modify the site properties to publish the site information in AD DS and in DNS.

To configure the publishing of site information in AD DS and in DNS

1. Select **Start**, point to **All Programs**, and then point to **Microsoft System Center 2012**. Point to **Configuration Manager**, and then select **Configuration Manager Console**.

2. In the Configuration Manager console, in the navigation pane, select **Administration**.

3. In the Administration workspace, go to Overview/Site Configuration/Sites.

4. In the preview pane, select **NYC - New York City Site**.

5. On the Ribbon, select **Properties**.

6. In the **New York City Site Properties** dialog box, on the **Publishing** tab, verify that the **mdt2013.corp.woodgrovebank.com** Active Directory forest is listed, and then select **Cancel**.

7. Close any open windows.

Step 1-15: Configure Discovery of Active Directory Users

In some instances, software will be deployed to user collections that Configuration Manager discovers. Configuration Manager can discovery user accounts stored in AD DS using the Active Directory User Discovery method.

To configure discovery of Active Directory users

1. Select **Start**, point to **All Programs**, and then point to **Microsoft System Center 2012**. Point to **Configuration Manager**, and then select **Configuration Manager Console**.

2. In the Configuration Manager console, in the navigation pane, select **Administration**.

3. In the Administration workspace, go to Overview/Hierarchy/Discovery Methods.

4. In the preview pane, select **Active Directory User Discovery**.

5. On the Ribbon, on the **Home** tab, select **Properties**.

   The **Active Directory User Discovery Properties** dialog box appears.

6. In the **Active Directory User Discovery Properties** dialog box, on the **General** tab, perform the following steps:

   a. Select the **Enable Active Directory User Discovery** check box.

b. In **Active Directory containers**, select **New**.

   The **New Active Directory Container** dialog box appears.

c. In the **New Active Directory Container** dialog box, in **Path**, select **Browse**.

   The **Select New Container** dialog box appears.

d. In the **Select New Container** dialog box, select **mdt2013**, and then select **OK**.

   In the **New Active Directory Container** dialog box, the Lightweight Directory Access Protocol (LDAP) path is displayed in the **Path** box.

e. In the **New Active Directory Container** dialog box, select OK.

   The LDAP path appears in the **Active Directory containers** list box.

7. In the **Active Directory User Discovery Properties** dialog box, select **OK**.

   The **Configuration Manager** dialog box appears, querying whether you want to perform the discovery as soon as possible.

8. In the **Configuration Manager** dialog box, select **Yes**.

9. In the Configuration Manager console, in the navigation pane, select **Assets and Compliance**.

10. In the Assets and Compliance workspace, go to Overview/Users.

    The list of users discovered in AD DS is displayed in the preview pane.

11. Close any open windows.

Step 2: Prepare the MDT Environment

The first step in the deployment process is to prepare the MDT environment. When this step is complete, you can create the reference computer and deploy a captured image of it to the target computer (WDG-CLI-01) using Configuration Manager integration with MDT.

Prepare the MDT environment by:

- Installing MDT as described in Step 2-1: Install MDT
- Enabling Configuration Manager console integration by running the Configure ConfigMgr Integration wizard as described in Step 2-2: Enable Configuration Manager Console Integration

Step 2-1: Install MDT

To install MDT, complete the following steps:

1. In Windows Explorer, go to E:\Source$\MDT_2013.
2. Double-click **MicrosoftDeploymentToolkit2013_x64.msi** (for 64-bit operating systems) or **MicrosoftDeploymentToolkit2013_x86.msi** (for 32-bit operating systems), and then select **Install**.

   The Microsoft Deployment Toolkit 2013 Setup Wizard starts.
3. Complete the Microsoft Deployment Toolkit 2013 Setup Wizard using the information in Table 17. Accept the default values unless otherwise specified.

Table 17. Information for Completing the Microsoft Deployment Toolkit 2013 Setup Wizard

Expand table

On this wizard page	Do this
Welcome to the Microsoft Deployment Toolkit 2013 Setup Wizard	Select **Next**.
End-User License Agreement	Select **I accept the terms in the License Agreement**, and then select **Next**.
Custom Setup	Select **Next**.
Ready to install Microsoft Deployment Toolkit 2013	Select **Install**.
Installing Microsoft Deployment Toolkit 2013	The progress for installing MDT is displayed.
Completed the Microsoft Deployment Toolkit 2013 Setup Wizard	Select **Finish**.

The Microsoft Deployment Toolkit 2013 Setup Wizard finishes, and MDT is installed on WDG-MDT-01.

Step 2-2: Enable Configuration Manager Console Integration

Before you can use the Configuration Manager integration features of MDT, run the Configure ConfigMgr Integration wizard. This wizard copies the appropriate integration files to the folder in which Configuration Manager is installed. The wizard also adds Windows Management Instrumentation (WMI) classes for the new MDT custom actions. The classes are added by compiling a new Managed Object Format (.mof) file that contains the new class definitions.

To enable Configuration Manager console integration

> **Note**
>
> Ensure that the Configuration Manager console is closed while performing these steps.

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Configure ConfigMgr Integration**.

   The Configure ConfigMgr Integration Wizard starts.

2. Complete the Configure ConfigMgr Integration Wizard using the information in Table 18. Accept the default values unless otherwise specified.

Table 18. Information for Completing the Configure ConfigMgr Integration Wizard

Expand table

On this wizard page	Do this
Options	1. Verify that the **Install the MDT console extensions for ConfigMgr 2012** check box is selected. 2. Verify that the **Add the MDT task sequence actions to a ConfigMgr server** check box is selected. 3. In **Site server name**, verify that the value is **WDG-MDT-01.mdt2013.corp.woodgrovebank.com**. 4. In **Site code**, verify that the value is **NYC**. 5. Select **Next**.
Confirmation	Select **Finish**.

The Configure ConfigMgr Integration Wizard finishes, and MDT is integrated with Configuration Manager.

Step 3: Create and Configure a Task Sequence to Create a Reference Computer

When you have prepared the MDT environment, create the reference computer. The reference computer is the template for deploying new images to the target computers. Configure this computer (WDG-REF-01) exactly as you will configure the target computers. You will then capture an image of the reference computer and deploy the image to the target computers.

Create the reference computer, WDG-REF-01, by:

- Creating an MDT task sequence to deploy Windows 8.1 to the reference computer as described in Step 3-1: Create an MDT Task Sequence for Deploying the Reference Computer
- Selecting the distribution points for the new packages and images that the Create MDT Task Sequence Wizard creates as described in Step 3-2: Select the Distribution Points for the New Packages and Images
- Adding the necessary device drivers to a new drive package and to the appropriate boot images as described in Step 3-3: Add the Necessary Device Drivers
- Enable monitoring of the MDT deployment process as described in Step 3-4: Enable MDT Deployment Process Monitoring
- Configuring the MDT configuration files for the reference computer—specifically, the CustomSettings.ini file—as described in Step 3-5: Customize the MDT Configuration Files for the Reference Computer
- Updating the Configuration Manager distribution points for the Custom Settings Files package as described in Step 3-6: Update the Distribution Points for the Custom Settings Files Package
- Customizing the task sequence for the reference computer as described in Step 3-7: Customize the Task Sequence for the Reference Computer

Step 3-1: Create an MDT Task Sequence for Deploying the Reference Computer

Use the Create MDT Task Sequence Wizard in the Configuration Manager console to create task sequences in Configuration Manager that are integrated with MDT. MDT includes the Standard Client Task Sequence template, which you can use to deploy the reference computer.

The Create MDT Task Sequence Wizard substitutes the packages and images selected for the placeholders in the task sequence templates. After completing the wizard, the new task sequence references the appropriate packages and images.

> **Note**
>
> Always use the Create MDT Task Sequence Wizard to create task sequences based on the MDT task sequence templates. Although you can manually import the task sequence templates, Microsoft does not recommend this process.

To create a task sequence for deploying the reference computer

1. Select **Start**, point to **All Programs**, and then point to **Microsoft System Center 2012**. Point to **Configuration Manager**, and then select **Configuration Manager Console**.
2. In the Configuration Manager console, in the navigation pane, select **Software Library**.
3. In the Software Library workspace, go to Overview/Operating Systems/Task Sequences.
4. On the Ribbon, on the **Home** tab, in the **Task Sequences** group, select **Create MDT Task Sequence**.

   The Create MDT Task Sequence Wizard starts.
5. Complete the Create MDT Task Sequence Wizard using the information in Table 19. Accept the default values unless otherwise specified.

Table 19. Information for Completing the Create MDT Task Sequence Wizard

Expand table

On this wizard page	Do this
Choose Template	Select **Client Task Sequence**, and then select **Next**.
Choose Template: General	1. In **Task sequence name**, type **Windows 8.1 Reference Deployment**. 2. In **Task sequence comments**, type **Task sequence for deploying Windows 8.1 to the reference computer (WDG-REF-01)**, and then select **Next**.

On this wizard page	Do this
Choose Template: Details	1. Select **Join a workgroup**. 2. In **Workgroup**, type **WORKGROUP**. 3. In **User name**, type **Woodgrove Bank Employee**. 4. In **Organization name**, type **Woodgrove Bank**. 5. In **Product key**, type ***product_key*** (where *product_key* is the product key for Windows 8.1). 6. Select **Next**.
Choose Template: Capture Settings	a. Select **This task sequence may be used to capture and image**. b. In **Capture destination**, type **\\WDG-MDT-01\Capture$\WDG-REF-01.wim**. c. In **Capture account**, select **Set**. d. Complete the **Windows User Account** dialog box by performing the following steps: i. In **User name**, type **MDT2013\Administrator**. ii. In **Password** and **Confirm password**, type **P@ssw0rd**. e. Select **OK**. f. Select **Next**.
Boot Image	1. Select **Create a new boot image package**. 2. In **Package source folder to be created**, type **\\WDG-MDT-01\Packages$\WINPE_Custom**, and then select **Next**.
Boot Image: General Settings	1. In **Name**, type **Windows PE Custom**. 2. In **Version**, type **1.00**. 3. In **Comments**, type **Customized version of Windows PE to be used in deployment of reference and target computers**, and then select **Next**.
Boot Image: Options	Under **Platform**, select **x64**, and then select **Next**.
Boot Image: Components	Select **Next**.
Boot Image: Customization	Select **Next**.
MDT Package	1. Select **Create a new Microsoft Deployment Toolkit Files package**. 2. In **Package source folder to be created**, type **\\WDG-MDT-01\Packages$\MDT_Files**, and then select **Next**.
MDT Package: MDT Details	1. In **Name**, type **MDT Files**. 2. In **Version**, type **1.00**. 3. In **Comments**, type **Provides access to MDT files during Configuration Manager deployment process**, and then select **Next**.

On this wizard page	Do this
OS Image	1. Select **Create a new OS install package**. 2. In **OS installation folder location**, type **\\WDG-MDT-01\Source$\Windows_8-1**. 3. In **Package source folder to be created**, type **\\WDG-MDT-01\Packages$\Windows_8-1**, and then select **Next**.
OS Image: Image Details	1. In **Name**, type **Windows 8.1**. 2. In **Version**, type **1.00**. 3. In **Comments**, type **Windows 8.1 package used to deploy to reference computers**, and then select **Next**.
Deployment Method	Select **Next**.
Client Package	Select **Create a new ConfigMgr client package**, and then select **Next**.
USMT Package	1. Select **Create a new USMT package**. 2. In **Package source folder to be created**, type **\\WDG-MDT-01\Packages$\USMT**, and then select **Next**.
USMT Package: USMT Details	1. In **Name**, type **USMT**. 2. In **Version**, type **1.00**. 3. In **Comments**, type **USMT files used to capture and restore user state migration information**, and then select **Next**.
Settings Package	1. Select **Create a new settings package**. 2. In **Package source folder to be created**, type **\\WDG-MDT-01\Packages$\CustomSettings_Reference**, and then select **Next**.
Settings Package: Settings Details	1. In **Name**, type **MDT Reference Computer Custom Settings**. 2. In **Version**, type **1.00**. 3. In **Comments**, type **Configuration settings for MDT deployment process (such as CustomSettings.ini) for the reference computer**, and then select **Next**.
Sysprep Package	Select **Next**.
Summary	Review the information in the **Details** box that you provided while completing the previous wizard pages, and then select **Next**.
Progress	The progress for creating the task sequence is displayed.
Confirmation	Select **Finish**.

The new task sequence appears in the preview pane.

Step 3-2: Select the Distribution Points for the New Packages and Images

The Create MDT Task Sequence Wizard creates a number of packages and images. After these packages and images are created, select the distribution points from which the packages and images will be copied and available to target computers.

> **Note**
>
> In this sample, there is only one distribution point (WDG-MDT-01). However, most production networks have multiple distribution points. When performing this step in a production environment, select the appropriate distribution points for the network.

To select the distribution points for software distribution packages

1. Select **Start**, point to **All Programs**, and then point to **Microsoft System Center 2012**. Point to **Configuration Manager**, and then select **Configuration Manager Console**.
2. In the Configuration Manager console, in the navigation pane, select **Software Library**.
3. In the Software Library workspace, go to Overview/Operating Systems/Task Sequences.
4. In the preview pane, select **Windows 8.1 Reference Deployment**.
5. On the Ribbon, on the **Home** tab, in the **Deployment** group, select **Distribute Content**.

   The Distribute Content Wizard starts.
6. Complete the Distribute Content Wizard using the information in 20. Accept default values unless otherwise specified.

Table 20. Information for Completing the Distribute Content Wizard

Expand table

On this wizard page	Do this
General	Select **Next**.
General: Content	Select **Next**.
General: Content Destination	1. Select **Add**, and then select **Distribution Point**. The **Add Distribution Points** dialog box appears. 2. In the **Add Distribution Points** dialog box, select **\\WDGMDT01.mdt2013.corp.woodgrovebank.com**, and then select **OK**. \\WDGMDT01.mdt2013.corp.woodgrovebank.com appears in the **Content destination** list. 3. Select **Next**.
Summary	Review the information in the **Details** box that you provided while completing the previous wizard pages, and then select **Next**.
Progress	The progress for distributing the software is displayed.
Completion	Select **Close**.

7. Close all open windows and dialog boxes.

Step 3-3: Add the Necessary Device Drivers

When the MDT task sequence has been created, add any device drivers required for the reference computer (WDG-REF-01) to the Windows PE boot image and to the Windows 8.1 image. Add the device drivers in the Drivers node in the Configuration Manager console. Create a package that contains the device drivers, and inject the drivers into the custom Windows PE image created earlier in the process.

After creating the package that contains the device drivers, select the distribution point to which the package will be deployed.

To add the necessary device drivers

1. Select **Start**, point to **All Programs**, and then point to **Microsoft System Center 2012**. Point to **Configuration Manager**, and then select **Configuration Manager Console**.

2. In the Configuration Manager console, in the navigation pane, select **Software Library**.

3. In the Software Library workspace, go to Overview/Operating Systems/Drivers.

4. On the Ribbon, on the **Home** tab, in the **Create** group, select **Import Driver**.

   The Import New Driver Wizard starts.

5. Complete the Import New Driver Wizard using the information in Table 21. Accept the default values unless otherwise specified.

Table 21. Information for Completing the Import New Driver Wizard

Expand table

On this wizard page	Do this
Locate Driver	**In Source folder**, type **\\WDG-MDT-01\Source$\Drivers**, and then select **Next**.
Locate Driver: Driver Details	Select **Next**.
Locate Driver: Add Driver to Package	a. Select **New Package**. b. Complete the **New Driver Package** dialog box by performing the following steps: i. In **Name**, type ***device_driver_name* Package** (where *device_driver_name* is a descriptive name for the device drivers). ii. In **Comment**, type **Device drivers that are necessary for the reference and target computers**. c. In **Driver package source**, type **\\WDG-MDT-01\Packages$\Drivers**, and then select **OK**. d. Select **Next**.
Locate Driver: Add Driver to Boot Images	1. In the list of images, select the **Windows PE Custom** check box. 2. Select the **Update distribution points when finished** check box, and then select **Next**.
Summary	Review the information in the **Details** box that you provided while completing the previous wizard pages, and then select **Next**.
Progress	The progress for importing the device drivers is displayed.
Confirmation	Select **Close**.

To select the distribution points for the driver package

6. Select **Start**, point to **All Programs**, and then point to **Microsoft System Center 2012**. Point to **Configuration Manager**, and then select **Configuration Manager Console**.

7. In the Configuration Manager console, in the navigation pane, select **Software Library**.

8. In the Software Library workspace, go to Overview/Operating Systems/Driver Packages.

9. In the preview pane, select ***device_driver_name* Package** (where *device_driver_name* is a descriptive name for the device drivers).

10. On the Ribbon, on the **Home** tab, in the **Deployment** group, select **Distribute Content**.

    The Distribute Content Wizard starts.

11. Complete the Distribute Content Wizard using the information in Table 22. Accept the default values unless otherwise specified.

Table 22. Information for Completing the Distribute Content Wizard

Expand table

On this wizard page	Do this
General	Select **Next**.
General: Content	Select **Next**.
General: Content Destination	1. Select **Add**, and then select **Distribution Point**. The **Add Distribution Points** dialog box appears. 2. In the **Add Distribution Points** dialog box, select **\\WDG-MDT-01.mdt2013.corp.woodgrovebank.com**, and then select **OK**. \\WDGMDT01.mdt2013.corp.woodgrovebank.com appears in the **Content destination** list. 3. Select **Next**.
Summary	Review the information in the **Details** box that you provided while completing the previous wizard pages, and then select **Next**.
Progress	The progress for distributing the software is displayed.

On this wizard page	Do this
Completion	Select **Close**.

12. Close all open windows and dialog boxes.

Step 3-4: Enable MDT Deployment Process Monitoring

Prior to deploying the reference computer (WDG-REF-01) with the task sequence bootable media, enable MDT monitoring of the ZTI deployment process. You enable monitoring on the **Monitoring** tab on the deployment share **Properties** dialog box. Later in the process, you will monitor the ZTI deployment process using the Deployment Workbench or the **Get-MDTMonitorData** cmdlet.

To enable MDT monitoring of the ZTI deployment process

1. Select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
2. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares.
3. In the Actions pane, **select New Deployment Shares**.

   The New Deployment Share Wizard starts.
4. Complete the New Deployment Share Wizard using the information in Table 23.

Table 23. Information for Completing the New Deployment Share Wizard

Expand table

On this wizard page	Do this
Path	**In Deployment share path**, type **C:\DeploymentShare$**, and then select **Next**.
Share	Select **Next**.
Descriptive Name	Select **Next**.
Options	Select **Next**.

On this wizard page	Do this
Summary	Select **Next**.
Progress	The progress for creating the deployment share is displayed.
Confirmation	Select Finish.

The New Deployment Share Wizard finishes, and the new deployment share—MDT Deployment Share (C:\DeploymentShare$)—appears in the details pane.

5. In the details pane, select **MDT Deployment Share (C:\DeploymentShare$)**.
6. In the Actions pane, select Properties.

   The **MDT Deployment Share (C:\DeploymentShare$) Properties** dialog box opens.
7. In the **MDT Deployment Share (C:\DeploymentShare$) Properties** dialog box, on the **Monitoring** tab, select the **Enable monitoring for this deployment share** check box, and then select **Apply**.
8. In the **MDT Deployment Share (C:\DeploymentShare$) Properties** dialog box, on the **Rules** tab, notice that the **EventService** property has been added to the CustomSettings.ini file, and then select **OK**.

   The **EventService** property is as follows:

   ```ini
   EventService=http://WDG-MDT-01:9800
   ```
9. Close all open windows and dialog boxes.

Step 3-5: Customize the MDT Configuration Files for the Reference Computer

When the MDT task sequence has been created, customize the MDT configuration files that provide the configuration settings for deploying Windows 8.1 to the target computer. Specifically, customize the CustomSettings.ini file.

When the CustomSettings.ini file customization is finished, save the updated files to the source folder for the MDT Reference Computer Custom Settings package created earlier in the process (E:\Packages$\CustomSettings_Reference). Then, add the **DoCapture** and

EventService properties and corresponding values to the CustomSettings.ini file so that the MDT deployment process captures an image of the reference computer (WDG-REF-01) after deploying Windows 8.1.

To customize the MDT configuration files for the reference computer

1. In Windows Explorer, go to E:\Packages$\CustomSettings_Reference, and then double-click **CustomSettings.ini**.

2. Open Microsoft Notepad, and then add the following lines to the end of the CustomSettings.ini file, as shown in Listing 1:

```ini
DoCapture=YES
EventService=http://WDG-MDT-01:9800
```

> **Note**
>
> Ensure that you remove any additional settings other than those shown in Listing 1.

Listing 1. CustomSettings.ini File After Adding the DoCapture Property

```ini
[Settings]
Priority=Default
Properties=MyCustomProperty

[Default]
OSInstall=Y
SkipCapture=YES
SkipAdminPassword=NO
SkipProductKey=YES
DoCapture=YES
EventService=http://WDG-MDT-01:9800
```

3. Save the file, and then exit Notepad.

Step 3-6: Update the Distribution Points for the Custom Settings Files Package

When the source folder has been updated for the MDT Reference Computer Custom Settings package in Configuration Manager, update the distribution points for the MDT Reference Computer Custom Settings Files package. Updating the distribution points copies the updated version of the CustomSettings.ini file to the deployment shares specified in the package.

To update the distribution points for the Custom Settings package

1. Select **Start**, point to **All Programs**, and then point to **Microsoft System Center 2012**. Point to **Configuration Manager**, and then select **Configuration Manager Console**.

2. In the Configuration Manager console, in the navigation pane, select **Software Library**.

3. In the Software Library workspace, go to Overview/Application Management/Packages.

4. In the preview pane, select **MDT Reference Computer Custom Settings**.

5. On the Ribbon, on the **Home** tab, in the **Deployment** group, select **Update Distribution Points**.

   The **Configuration Manager** dialog box opens, notifying you that you are going to update the package on all distribution points.

6. In the **Configuration Manager** dialog box, select **OK**.

7. Close all open windows and dialog boxes.

   Configuration Manager starts updating the distribution points with the latest versions of the CustomSettings.ini file. This process could take several minutes. Check the status of the package until the **Last Update** value of the package status has been updated to a recent date and time.

Step 3-7: Customize the Task Sequence for the Reference Computer

For most deployments, the **Windows 8.1 Reference Deployment** task sequence created earlier in the process performs all the necessary steps without modification. In this sample, modify the task sequence to set the password for the local Administrator account to a known value. By default, the task sequence sets the password for the local Administrator account to a random value. Further customization of the task sequence may be required depending on the environment.

To customize the Windows 8.1 Reference Deployment task sequence

1. Select **Start**, point to **All Programs**, and then point to **Microsoft System Center 2012**. Point to **Configuration Manager**, and then select **Configuration Manager Console**.

2. In the Configuration Manager console, in the navigation pane, select **Software Library**.

3. In the Software Library workspace, go to Overview/Operating Systems/Task Sequences.

4. In the preview pane, select **Windows 8.1 Reference Deployment**.

5. On the Ribbon, on the **Home** tab, in the **Task Sequence** group, select **Edit**.

   The **Windows 8.1 Reference Deployment Task Sequence Editor** dialog box opens.

6. In the **Windows 8.1 Reference Deployment Task Sequence Editor** dialog box, go to PostInstall/Apply Windows Settings.

7. On the **Properties** tab, select **Enable the account and specify the local administrator password**.

8. On the **Properties** tab, in **Password** and **Confirm Password**, type **P@ssw0rd**, and then select **Apply**.

9. Make any additional modifications to the task sequence that the environment requires, and then select **OK**.

10. Close all open windows and dialog boxes.

Step 4: Deploy Windows 8.1 and Capture an Image of the Reference Computer

When you have created the task sequence to deploy Windows 8.1 to the reference computer and captured an image of the reference computer, start the task sequence. Create the operating system capture by using the Task Sequence Media Wizard in the Configuration Manager console.

Deploy Windows 8.1 and capture an image of the reference computer by:

- Adding the reference computer to the Configuration Manager site database as described in Step 4-1: Add the Reference Computer to the Configuration Manager Site Database

- Creating a collection that contains the reference computer you just added as described in Step 4-2: Create a Collection That Contains the Reference Computer
- Deploy the reference computer task sequence as described in Step 4-3: Deploy the Reference Computer Task Sequence
- Using the Task Sequence Media Wizard to create a task sequence bootable media disk as described in Step 4-4: Create the Task Sequence Bootable Media
- Starting the reference computer with the task sequence bootable media disk as described in Step 4-5: Start the Reference Computer with the Task Sequence Bootable Media

Step 4-1: Add the Reference Computer to the Configuration Manager Site Database

To deploy an operating system without stand-alone media to a new computer that Configuration Manager does not currently manage, add the new computer to the Configuration Manager site database prior to initiating the operating system deployment process. Configuration Manager can automatically discover computers on the network that have a Windows operating system installed; however, if the computer has no operating system installed, use the Import Computer Information Wizard to import the new computer information.

To add the reference computer to the Configuration Manager site database

1. Select **Start**, point to **All Programs**, and then point to **Microsoft System Center 2012**. Point to **Configuration Manager**, and then select **Configuration Manager Console**.
2. In the Configuration Manager console, in the navigation pane, select **Assets and Compliance**.
3. In the Assets and Compliance workspace, go to Overview/Devices.
4. On the Ribbon, on the **Home** tab, in the **Create** group, select **Import Computer Information**.

   The Import Computer Information Wizard starts.
5. Complete the Import Computer Information Wizard using the information in 24. Accept the default values unless otherwise specified.

Table 24. Information for Completing Import Computer Information Wizard

Expand table

On this wizard page	Do this
Select Source	Select **Import single computer**, and then select **Next**.
Select Source: Single Computer	1. In **Computer Name**, type **WDG-REF-01**. 2. In **MAC address**, type ***mac_address*** (where *mac_address* is the media access control [MAC] address of the primary network adapter for the reference computer, WDG-REF-01). 3. Select **Next**.
Select Source: Data Preview	Select **Next**.
Select Source: Choose Target Collection	Select **Next**.
Summary	Review the information in the **Details** box that you provided while completing the previous wizard pages, and then select **Next**.
Progress	The progress for importing the computer is displayed.
Confirmation	Select **Close**.

For more information on adding a new computer to the Configuration Manager site database, see the section, "To import computer information for a single computer," in the section, "How to Deploy Operating Systems in Configuration Manager," in the Configuration Manager Documentation Library, which is installed with Configuration Manager.

Step 4-2: Create a Collection That Contains the Reference Computer

In the Configuration Manager console, create a collection that includes the reference computer (WDG-REF-01). This computer collection is used later when advertising the task sequence created earlier in the process.

To create a collection that includes the reference computer

1. Select **Start**, point to **All Programs**, and then point to **Microsoft System Center 2012**. Point to **Configuration Manager**, and then select **Configuration Manager**

Console.

2. In the Configuration Manager console, in the navigation pane, select **Assets and Compliance**.
3. In the Assets and Compliance workspace, go to Overview/Device Collections.
4. On the Ribbon, on the **Home** tab, in the **Create** group, select **Create**, and then select **Create Device Collection**.

   The Create Device Collection Wizard starts.
5. Complete the Create Device Collection Wizard using the information in Table 25. Accept the default values unless otherwise specified.

Table 25. Information for Completing the Create Device Collection Wizard

Expand table

On this wizard page	Do this
General	a. In **Name**, type **Microsoft Deployment - Reference Computer**. b. In **Comment**, type **Computer that is to be the reference computer for the target computers to be deployed**. c. In **Limited Collection**, select **Browse**. The **Select Collection** dialog box appears. Complete the dialog box by performing the following steps: i. In **Name**, select **All Systems**. ii. Select **OK**. d. Select **Next**.
Membership Rules	a. Select **Add Rule**, and then select **Direct Rule**. The Create Direct Membership Rule Wizard starts. b. Complete the Create Direct Membership Rule Wizard by performing the following steps: i. On the **Welcome** page, select **Next**. ii. On the **Search for Resources** page, in **Resource class**, select **System Resource**; in **Attribute name**, select **Name**; in **Value**, type **WDG-REF-01**; and then select **Next**. iii. On the **Select Resources** page, select **WDG-REF-01**, and then select **Next**.

On this wizard page	Do this
	iv. On the **Summary** page, select **Next**. v. On the **Progress** page, view the progress for creating the new membership rule. vi. On the **Completion** page, select **Close**. c. Select **Next**.
Summary	Review the information in the **Details** box that you provided while completing the previous wizard pages, and then select **Next**.
Progress	The progress for creating the device collection is displayed.
Completion	Select **Close**.

For more information, see the section, "How to Create Collections in Configuration Manager," in the Configuration Manager Documentation Library, which is installed with Configuration Manager.

Step 4-3: Deploy the Reference Computer Task Sequence

In the Configuration Manager console, deploy the task sequence created earlier in the process to the device collection that includes the reference computer created earlier in the process.

To deploy the task sequence

1. Select **Start**, point to **All Programs**, and then point to **Microsoft System Center 2012**. Point to **Configuration Manager**, and then select **Configuration Manager Console**.

2. In the Configuration Manager console, in the navigation pane, select **Software Library**.

3. In the Software Library workspace, go to Overview/Operating Systems/Task Sequences.

4. In the preview pane, select **Windows 8.1 Reference Deployment**.

5. On the Ribbon, on the **Home** tab, in the **Deployment** group, select **Deploy**.

   The Deploy Software Wizard starts.

6. Complete the Deploy Software Wizard using the information in Table 26. Accept the default values unless otherwise specified.

Table ARABIC 26. Information for Completing the Deploy Software Wizard

Expand table

On this wizard page	Do this
General	1. In **Collection**, select **Browse**. 2. In the **Browse Collection** dialog box, select **Microsoft Deployment - Reference Computer**, and then select **OK**. 3. In **Comment**, type **Deploy Windows 8.1 to the reference computer and then capture an image of the reference computer**. 4. Select **Next**.
Deployment Settings	1. In **Purpose**, select **Available**. 2. Select the **Make available to boot media and PXE** check box. 3. Select **Next**.
Deployment Settings: Schedule	Select **Next**.
Deployment Settings: User Experience	Select **Next**.
Deployment Settings: Alerts	Select **Next**.
Deployment Settings: Distribution Points	Select **Next**.
Summary	Review the information in the **Details** box that you provided while completing the previous wizard pages, and then select **Next**.
Progress	The progress for deploying the task sequence is displayed.
Completion	Select **Close**.

For more information, see the section, "How to Deploy a Task Sequence," in the Configuration Manager Documentation Library, which is installed with Configuration Manager.

Step 4-4: Create the Task Sequence Bootable Media

To initiate the MDT process, provide a method for starting the computer with Windows PE and the necessary software by creating the task sequence bootable media disk. Use

the Task Sequence Media Wizard in the Configuration Manager console to create bootable media for storage on a USB flash drive, CD, or DVD.

To create a task sequence bootable media disk

1. Select **Start**, point to **All Programs**, and then point to **Microsoft System Center 2012**. Point to **Configuration Manager**, and then select **Configuration Manager Console**.

2. In the Configuration Manager console, in the navigation pane, select **Software Library**.

3. In the Software Library workspace, go to Overview/Operating Systems/Task Sequences.

4. On the Ribbon, on the **Home** tab, in the **Create** group, select **Create Task Sequence Media**.

   The Create Task Sequence Media Wizard starts.

5. Complete the Create Task Sequence Media Wizard using the information in Table 27. Accept the default values unless otherwise specified.

Table 27. Information for Completing the Create Task Sequence Media Wizard

Expand table

On this wizard page	Do this
Select Media Type	1. Select **Bootable media**. 2. Clear the **Allow unattended operating system deployment** check box. 3. Select **Next**.
Select Media Type: Media Management	Select **Site-based media**, and then select **Next**.
Select Media Type: Media Type	In **Media file**, type **\\WDG-MDT-01\Capture$\CM2012_TS_Boot_Media.iso**, and then select **Next**.
Select Media Type: Security	In **Password** and **Confirm password**, type **P@ssw0rd**, and then select **Next**.
Select Media Type: Boot Image	1. In **Boot image**, select **Browse**. 2. In the **Select a Boot Image** dialog box, select **Windows PE Custom**, and then select **OK**.

On this wizard page	Do this
	3. In **Distribution point**, select **\\WDG-MDT-01.mdt2013.corp.woodgrovebank.com**, and then select **OK**. 4. In **Management point**, select **\\WDG-MDT-01.mdt2013.corp.woodgrovebank.com**, and then select **OK**. 5. Select **Next**.
Select Media Type: Customization	Select **Next**.
Summary	Review the information in the **Details** box that you provided while completing the previous wizard pages, and then select **Next**.
Progress	The progress for creating the task sequence media is displayed.
Completion	Select **Close**.

The wizard creates the CM2012_TS_Boot_Media.iso file in the WDG-MDT-01Capture$ shared folder.

6. If WDG-REF-01 is a physical computer, create a CD or DVD of the International Organization for Standardization (ISO) file. If WDG-REF-01 is a VM, start the VM directly from the ISO file.

   For more information on creating the task sequence bootable media disk, see the section, "How to Create Bootable Media," in the Configuration Manager Documentation Library, which is installed with Configuration Manager.

Step 4-5: Start the Reference Computer with the Task Sequence Bootable Media

Start the reference computer (WDG-REF-01) with the task sequence bootable media disk created earlier in the process. This medium starts Windows PE on the reference computer and initiates the MDT process. At the end of the MDT process, Windows 8.1 is deployed on the reference computer and an image of the reference computer is saved to \WDG-MDT-01\Capture$\WDG-REF-01.wim.

> ⓘ **Note**
>
> You can also initiate the MDT process by starting the target computer from Windows Deployment Services.

To start the reference computer with the task sequence bootable media

1. Start WDG-REF-01 with the task sequence bootable media created earlier in the process.

   Windows PE starts, and then the Task Sequence Wizard starts.

2. Complete the Task Sequence Wizard using the information in Table 28. Accept the default values unless otherwise specified.

Table 28. Information for Completing the Task Sequence Wizard

Expand table

On this wizard page	Do this
Welcome to the Task Sequence Wizard	In **Password**, type **P@ssw0rd**, and then select **Next**.
Select a Task Sequence	In the list box, select **Windows 8.1 Reference Deployment**, and then select **Next**.

To monitor the reference computer deployment process using the Deployment Workbench

3. On WDG-MDT-01, select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.

4. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/MDT Deployment Share (C:\DeploymentShare$)/Monitoring.

5. In the details pane, view the deployment process for WDG-REF-01.

6. In the Actions pane, periodically select **Refresh**.

   The status of the deployment process is updated in the details pane. Continue to monitor the deployment process until the process is complete.

7. In the details pane, select **WDG-REF-01**.

8. In the Actions pane, select **Properties**.

   The **WDG-REF-01 Properties** dialog box is displayed.

9. In the **WDG-REF-01 Properties** dialog box, on the **Identity** tab, view the monitoring information provided about the deployment process as described in

Table 29. Monitoring Information About the Deployment Process

Expand table

Information	Description
ID	Unique identifier for the computer being deployed.
Computer Name	The name of the computer being deployed.
Deployment status	The current status of the computer being deployed; the status can be one of the following: - **Running**. The task sequence is healthy and running. - **Failed**. The task sequence failed, and the deployment process was unsuccessful. - **Completed**. The task sequence has finished. - **Unresponsive**. The task sequence has not updated its status in the past four hours and is assumed to be nonresponsive.
Step	The current task sequence step being run.
Progress	The overall progress of the task sequence. The progress bar indicates how many task sequence steps have been run out of the total number of task sequence steps.
Start	The time the deployment process started.
End	The time the deployment process ended.
Elapsed	The length of time the deployment process has been running or took to run if the deployment process has finished.
Errors	The number of errors encountered during the deployment process.
Warnings	The number of warnings encountered during the deployment process.
Remote Desktop	This button allows you to establish a remote desktop connection with the computer being deployed using the Windows Remote Desktop feature. This method assumes that: - The target operating system is running and has remote desktop support enabled - **mstsc.exe** is in the path **Note:** This button is always visible but may not be able to establish a remote desktop session if the monitored computer is running Windows PE, has not completed installation of

Information	Description
	the target operating system, or does not have the Remote Desktop feature enabled.
VM Connection	This button allows you to establish a remote desktop connection to a VM running in HyperV®. This method assumes that: - The deployment is being performed to a VM running on Hyper-V - **vmconnect.exe** is located in the %ProgramFiles%\Hyper-V folder **Note:** This button appears when ZTIGather.wsf detects that Hyper-V integration components are running on the monitored computer. Otherwise, this button will not be visible.
DaRT Remote Control	This button allows you to establish a remote control session using the remote viewer feature in the Diagnostics and Recovery Toolkit (DaRT). This method assumes that: - DaRT has been deployed to the target computer and is currently running - **DartRemoteViewer.exe** is located in the %ProgramFiles%\Microsoft DaRT 7\v7 folder **Note:** This button appears when ZTIGather.wsf detects that DaRT is running on the monitored computer. Otherwise, this button will not be visible.
Automatically refresh this information every 10 seconds	Check box that controls whether the information in the dialog box is automatically refreshed. If the check box is: - Selected, the information is refreshed every 10 seconds - Cleared, the information is not automatically refreshed and must be manually refreshed using the **Refresh Now** button
Refresh Now	This button immediately refreshes the information displayed in the dialog box.

10. In the **WDG-REF-01 Properties** dialog box, select **OK**.

11. Close the Deployment Workbench.

    To monitor the reference computer deployment process using the Get-MDTMonitorData cmdlet

12. On WDG-MDT-01, select **Start**, point to **Administrative Tools**, and then select **Windows PowerShell Modules**.

    The Windows PowerShell Modules command prompt opens.

13. Create a Windows PowerShell drive that uses the MDT PowerShell provider by running the New-PSDrive cmdlet, as shown in the following example:

```
PowerShell

New-PSDrive -Name DS001 -PSProvider mdtprovider -Root
d:\DeploymentShare$
```

14. View the MDT monitoring process by running the **Get-MDTMonitorData** cmdlet, as shown in the following example:

```
PowerShell

Get-MDTMonitorData -Path DS001:
```

    This command returns the monitoring data collected by the MDT monitoring service running on the same computer that hosts the deployment share, as shown in the following example output:

```
PowerShell

Name               : WDG-REF-01
PercentComplete    : 96
Settings           :
Warnings           : 0
Errors             : 0
DeploymentStatus   : 1
StartTime          : 6/7/2012 6:45:39 PM
EndTime            :
ID                 : 1
UniqueID           : 94a0830e-f2bb-421c-b1e0-6f86f9eb9fa1
CurrentStep        : 130
TotalSteps         : 134
StepName           : Gather
LastTime           : 6/7/2012 8:46:32 PM
DartIP             :
DartPort           :
DartTicket         :
VMHost             : XYL-DC-02
VMName             : WDG-REF-01
ComputerIdentities : {}
```

15. Close the Windows PowerShell console.

    If any problems occur during the deployment, consult the MDT document *Troubleshooting Reference*. When completed, a captured image of the reference computer should exist in \\WDG-MDT-01\Capture$\WDG-REF-01.wim.

Step 5: Create and Configure a Task Sequence to Deploy the Target Computer

After the task sequence to deploy the reference computer (WDG-REF-01) finishes, a captured image of the reference computer is stored in \\WDG-MDT-01\Capture$\WDG-REF-01.wim. Now, create a task sequence that will deploy the captured image of the reference computer to the target computer (WDG-CLI-01). When this step is complete, you can deploy the captured image of the reference computer to the target computer.

Create and configure a task sequence to deploy the target computer by:

- Importing the .wim file captured in the previous step into Configuration Manager using the Add Operating System Image Wizard as described in Step 5-1: Import the Captured .wim File into Configuration Manager
- Using the Create MDT Task Sequence Wizard to create an MDT task sequence template to deploy the captured image of the reference computer to the target computer as described in Step 5-2: Create an MDT Task Sequence to Deploy the Captured Image
- Selecting the distribution points for the new packages and images that the Create MDT Task Sequence Wizard creates as described in Step 5-3: Select the Distribution Points for the New Packages and Images
- Customizing the MDT configuration files for the target computer—specifically, the CustomSettings.ini file—as described in Step 5-4: Customize the MDT Configuration Files
- Updating the Configuration Manager distribution points for the Custom Settings package as described in Step 5-5: Update the Distribution Points for the Custom Settings Package
- Customizing the task sequence for the target computer as described in Step 5-6: Customize the Task Sequence for the Target Computer
- Configuring unattended installation of Office Professional Plus 2010 as described in Step 5-7: Configure an Unattended Installation of Office Professional Plus 2010
- Creating a Configuration Manager application to deploy Office Professional Plus 2010 as described in Step 5-8: Create an Office Professional Plus 2010 Application
- Distributing the Office Professional Plus 2010 application to the distributions points as described in Step 5-9: Distribute the Office Professional Plus 2010 Application

- Making the Office Professional Plus 2010 application available to all users as described in Step 5-10: Make the Office Professional Plus 2010 Application Available to All Users
- Customizing the UDI Wizard configuration file as described in Step 5-11: Customize the UDI Wizard Configuration File for the Target Computer
- Creating a new custom wizard page to collect additional deployment information as described in Step 5-13: Create a New Custom Wizard Page
- Adding controls to the new custom wizard page as described in Step 5-14: Add Controls to New Custom Wizard Page
- Updating the MDT files package that contains the updated UDI Wizard configuration file as described in Step 5-15: Update the Distribution Points for the MDT Files Package

Step 5-1: Import the Captured .wim File into Configuration Manager

After the image of the reference computer (WDG-REF-01) is captured in to the .wim file, import the captured .wim file into Configuration Manager. Import the captured .wim file into the Operating System Images node using the Add Operating System Image Wizard.

The captured WIM file contains two images, one for each partition on the reference computer. Identify which of the images has the captured Windows 8.1 operating system using the image description containing Windows 8.1. You use the image index when you create the task sequence for deploying the captured image to the target computer.

To import the captured .wim file into Configuration Manager

1. Select **Start**, point to **All Programs**, and then point to **Microsoft System Center 2012**. Point to **Configuration Manager**, and then select **Configuration Manager Console**.
2. In the Configuration Manager console, in the navigation pane, select **Software Library**.
3. In the Software Library workspace, go to Overview/Operating Systems/Operating System Images.
4. On the Ribbon, in the **Create** group, select **Add Operating System Image**.

   The Add Operating System Image Wizard starts.

5. Complete the Add Operating System Image Wizard using the information in Table 30. Accept the default values unless otherwise specified.

Table 30. Information for Completing the Add Operating System Image Wizard

Expand table

On this wizard page	Do this
Data Source	In Path, type **\\WDG-MDT-01\Capture$\WDG-REF-01.wim**, and then select **Next**.
General	1. In **Name**, type **Windows 8.1 Reference Image**. 2. In **Version**, type **1.00**. 3. In **Comments**, type **Windows 8.1 captured image of reference computer (WDG-REF-01) used to deploy to target computers**, and then select **Next**.
Summary	Review the information in the **Details** box that you provided while completing the previous wizard pages, and then select **Next**.
Progress	The progress for importing the operating system image is displayed.
Completion	Select **Close**.

6. In the preview pane, select **Windows 8.1 Reference Image**.

7. In the preview pane, select the **Details** tab.

   The list of operating system partitions captured in the .wim file is displayed. The image index that contains Windows 8.1 is the image index you will specify later during the Create MDT Task Sequence Wizard.

8. Record the image index that contains Windows 8.1.

> **Tip**
>
> For the purposes of this example, image index 2 should have the Windows 8.1 operating system.

Step 5-2: Create an MDT Task Sequence to Deploy the Captured Image

After the image is captured, create a task sequence to deploy the captured image of the reference computer (WDG-REF-01) to the target computer (WDG-CLI-01). Most of the packages needed for this task sequence were created earlier in the process. However, you must create a new MDT Custom Settings package that has the proper configuration settings for the target computer and creates an operating system image of the captured image of the reference computer.

To create a task sequence template to deploy the captured image to the target computer

1. Select **Start**, point to **All Programs**, and then point to **Microsoft System Center 2012**. Point to **Configuration Manager**, and then select **Configuration Manager Console**.

2. In the Configuration Manager console, in the navigation pane, select **Software Library**.

3. In the Software Library workspace, go to Overview/Operating Systems/Task Sequences.

4. On the Ribbon, on the **Home** tab, in the **Task Sequences** group, select **Create MDT Task Sequence**.

   The Create MDT Task Sequence Wizard starts.

5. Complete the Create MDT Task Sequence Wizard using the information in Table 31. Accept the default values unless otherwise specified.

Table 31. Information for Completing the Create MDT Task Sequence Wizard

Expand table

On this wizard page	Do this
Choose Template	Select **Client Task Sequence**, and then select **Next**.
Choose Template: General	1. In **Task sequence name**, type **UDI - Windows 8.1 Target Deployment**. 2. In **Task sequence comments**, type **Task sequence for deploying captured reference computer image to the target computer (WDG-CLI-01) using UDI**, and then select **Next**.
Choose Template:	1. In **Use name**, type **Woodgrove Bank Employee**. 2. In **Organization name**, type **Woodgrove Bank**.

On this wizard page	Do this
Details	3. Select **Next**.
Choose Template: Capture Settings	Select **Next**.
Boot Image	1. In **Specify an existing boot image package**, select **Browse**. 2. In **Select a Package** dialog box, select **Windows PE Custom**, and then select **OK**. 3. Select **Next**.
MDT Package	1. In **Specify an existing Microsoft Deployment Toolkit Files package**, select **Browse**. 2. In the **Select a Package** dialog box, select **MDT Files**, and then select **OK**. 3. Select **Next**.
OS Image	1. Select **Specify an existing OS image**. 2. In **Specify an existing OS image**, select **Browse**. 3. In the **Select a Package** dialog box, select **Windows 8.1 Reference Image**, and then select **OK**. 4. Select **Next**.
OS Image: OS Image Index	1. In **The selected operating system image (WIM) file contains multiple images. Specify which image you would like to deploy**, select ***image_index*** (where *image_index* is the image index of the image that contains Windows 8.1, which was identified in the Step 5-1: Import the Captured .wim File into Configuration Manager; for the purposes of this guide, select 2). 2. Select **Next**.
Deployment Method	Select **Perform a "User-Driven Installation"**, and then select **Next**.
Client Package	1. In **Specify an existing ConfigMgr client package**, select **Browse**. 2. In the **Select a Package** dialog box, select **Microsoft Configuration Manager Client Upgrade**, and then select **OK**. 3. Select **Next**.
USMT Package	1. In **Specify an existing USMT package**, select **Browse**. 2. In the **Select a Package** dialog box, select **USMT**, and then select **OK**. 3. Select **Next**.
Settings Package	1. Select **Create a new settings package**. 2. In **Package source folder to be created**, type **\\WDG-MDT-01\Packages$\UDICustomSettings_Target**, and then select **Next**.

On this wizard page	Do this
Settings Package: Settings Details	1. In **Name**, type **UDI Target Computer Custom Settings**. 2. In **Version**, type **1.00**. 3. In **Comments**, type **Configuration settings for MDT deployment process using UDI (such as CustomSettings.ini) for the target computer**, and then select **Next**.
Sysprep Package	Select **Next**.
Summary	Review the information in the **Details** box that you provided while completing the previous wizard pages, and then select **Next**.
Progress	The progress for creating the task sequence is displayed.
Confirmation	Select **Finish**.

The list of task sequences is displayed. The task sequence that you just created (**UDI - Windows 8.1 Target Deployment**) is listed in the list of task sequences.

Step 5-3: Select the Distribution Points for the New Packages and Images

Running the Create MDT Task Sequence Wizard to create the task sequence for the target generates a new software-distribution package and a new image. When the package and image are created, select the distribution points from which the package and image will be copied and available to target computers.

> ⓘ **Note**
>
> In this sample, there is only one distribution point (WDG-MDT-01). However, most production networks will have multiple distribution points. When performing this step in a production environment, select the appropriate distribution points for the network.

Select the distribution points for the software-distribution package (for the new target computer custom settings package called *MDT 2013 Target Computer Custom Settings*) and the operating system image package (for the new captured .wim file of the reference computer called *Windows 8.1 Reference Image*).

To select the distribution points for the software-distribution package

1. Select **Start**, point to **All Programs**, and then point to **Microsoft System Center 2012**. Point to **Configuration Manager**, and then select **Configuration Manager Console**.

2. In the Configuration Manager console, in the navigation pane, select **Software Library**.

3. In the Software Library workspace, go to Overview/Operating Systems/Task Sequences.

4. In the preview pane, select **UDI - Windows 8.1 Target Deployment**.

   On the Ribbon, on the **Home** tab, in the **Deployment** group, select **Distribute Content**.

   The Distribute Content Wizard starts.

5. Complete the Distribute Content Wizard using the information in Table 32. Accept default values unless otherwise specified.

Table 32. Information for Completing the Distribute Content Wizard

Expand table

On this wizard page	Do this
General	Select **Next**.
Content	Select **Next**.
General: Content Destination	1. Select **Add**, and then select **Distribution Point**. The **Add Distribution Points** dialog box appears. 2. In the **Add Distribution Points** dialog box, select **\\WDGMDT01.mdt2013.corp.woodgrovebank.com**, and then select **OK**. \\WDGMDT01.mdt2013.corp.woodgrovebank.com appears in the **Content destination** list. 3. Select **Next**.
Summary	Review the information in the **Details** box that you provided while completing the previous wizard pages, and then select **Next**.
Progress	The progress for distributing the software is displayed.
Completion	Select **Close**.

Step 5-4: Customize the MDT Configuration Files

When the task sequence for the target computer has been created, customize the MDT configuration files that provide the configuration settings for deploying Windows 8.1 to the target computer—specifically, CustomSettings.ini.

When the CustomSettings.ini file has been customized, save the updated files to the source folder for the MDT Custom Settings package created earlier in the process (E:\Packages$\CustomSettings_Target).

To customize the MDT configuration files for the target computer

1. In Windows Explorer, go to E:\Packages$\CustomSettings_Target folder, and then double-click **CustomSettings.ini**.

2. Open Notepad, and then add the following line to the CustomSettings.ini file that the environment requires, as shown in Listing 2:

   This setting configures monitoring of the target computer deployment.

   > ⓘ **Note**
   >
   > Make any other changes that your environment requires.

   Listing 2. Default CustomSettings.ini File

   ini

   ```ini
   [Settings]
   Priority=Default
   Properties=MyCustomProperty

   [Default]
   OSInstall=Y
   SkipCapture=YES
   SkipAdminPassword=NO
   SkipProductKey=YES
   EventService=http://WDG-MDT-01:9800
   ```

3. Save the file, and then close Notepad.

Step 5-5: Update the Distribution Points for the Custom Settings Package

When the source folder has been updated for the MDT Target Computer Custom Settings package in Configuration Manager, update the distribution points for the MDT Target Computer Custom Settings package. Updating the distribution points copies the updated version of the CustomSettings.ini file to the deployment shares specified in the package.

To update the distribution points for the Custom Settings package

1. Select **Start**, point to **All Programs**, and then point to **Microsoft System Center 2012**. Point to **Configuration Manager**, and then select **Configuration Manager Console**.

2. In the Configuration Manager console, in the navigation pane, select **Software Library**.

3. In the Software Library workspace, go to Overview/Application Management/Packages.

4. In the preview pane, select **MDT Target Computer Custom Settings**.

5. On the Ribbon, on the **Home** tab, in the **Deployment** group, select **Update Distribution Points**.

   The **Configuration Manager** dialog box opens, notifying you that you are going to update the package on all distribution points.

6. In the **Configuration Manager** dialog box, select **OK**.

7. Close all open windows and dialog boxes.

Step 5-6: Customize the Task Sequence for the Target Computer

For most deployments, the Windows 8.1 Target Deployment task sequence created earlier in the process performs all the necessary steps without modification. In this sample, modify the task sequence template to set the password for the local Administrator account to a known value. (By default, the task sequence sets the password for the local Administrator account to a random value.) The task sequence may require further customization depending on the environment.

To customize the Windows 8.1 Target Deployment task sequence

1. Select **Start**, point to **All Programs**, and then point to **Microsoft System Center 2012**. Point to **Configuration Manager**, and then select **Configuration Manager**

Console.

2. In the Configuration Manager console, in the navigation pane, select **Software Library**.
3. In the Software Library workspace, go to Overview/Operating Systems/Task Sequences.
4. In the preview pane, select **UDI - Windows 8.1 Target Deployment**.
5. On the Ribbon, on the **Home** tab, in the **Task Sequence** group, select **Edit**.

   The **Windows 8.1 Reference Deployment Task Sequence Editor** dialog box opens.
6. In the **Windows 8.1 Reference Deployment Task Sequence Editor** dialog box, go to PostInstall/Apply Windows Settings.
7. On the **Properties** tab, select **Enable the account and specify the local administrator password**.
8. On the **Properties** tab, in **Password** and **Confirm Password**, type **P@ssw0rd**, and then select **Apply**.
9. Make any additional modifications to the task sequence that the environment requires, and then select **OK**.
10. Close all open windows and dialog boxes.

Step 5-7: Configure an Unattended Installation of Office Professional Plus 2010

Configuration Manager distributes the files and folders used to deploy Office Professional Plus 2010 but does not provide the method for performing an unattended installation after distribution. Instead, the unattended installation must be configured using methods provided in Office Professional Plus 2010. You can configure unattended (silent) installation of Office Professional Plus 2010 using one of the following methods:

- Create an Office Customization Tool (OCT) Setup customization file (.msp file).
- Modify the Config.xml file.

  For more information about each of these methods, see Customize Setup before installing Office 2010.

For the purposes of this guide, the unattended installation of Office Professional Plus 2010 will be done by creating an OCT Setup customization file (.msp file). You will save the OCT Setup customization file in the Updates folder, which is automatically scanned by the Office Professional Plus 2010 Setup Wizard.

To configure an unattended installation of Office Professional Plus 2010

1. At a command prompt, type the following command, and then press ENTER.

```
Windows Command Prompt

e:
```

2. At a command prompt, type the following command, and then press ENTER.

```
Windows Command Prompt

cd \Source$\OfficeProPlus2010\
```

3. At a command prompt, type the following command, and then press ENTER.

```
Windows Command Prompt

setup /admin
```

   The OCT starts, and the **Select Product** dialog box opens.

4. In the OCT, in the **Select Product** dialog box, select **OK**.

   The OCT loads the appropriate information, and then displays the settings that can be customized in the .msp file.

5. In the OCT, in the navigation pane, go to Setup/Install location and organization name.

6. In the preview pane, in **Organization name**, type **Woodgrove Bank**.

7. In the OCT, in the navigation pane, go to Setup/Licensing and user interface.

8. In the preview pane, select the **I accept the terms in the License Agreement** check box.

9. In the preview pane, in **Display level**, select **None**.

10. From the **File** menu, select **Save As**.

The **Save As** dialog box opens.

11. In the **Save As** dialog box, type **E:\Source$\OfficeProPlus2010\Updates\OPP2010_Unattend**, and then select **Save**.

    The OPP2010_Unattend.msp file is saved.

12. Close all open windows and dialog boxes.

Step 5-8: Create an Office Professional Plus 2010 Application

One of the advantages to performing MDT deployments using UDI is the ability for the user to select the applications to install at deployment time. You can add any number of applications to Configuration Manager, and then select the applications when running the UDI Wizard, as described in Step 6-4: Start the Target Computer with the Task Sequence Bootable Media.

You can configure the applications that appear in the UDI Wizard using the UDI Wizard Designer, as described in Step 5-11: Customize the UDI Wizard Configuration File for the Target Computer.

To create an Office Professional Plus 2010 application

1. Select **Start**, point to **All Programs**, and then point to **Microsoft System Center 2012**. Point to **Configuration Manager**, and then select **Configuration Manager Console**.

2. In the Configuration Manager console, in the navigation pane, select **Software Library**.

3. In the Software Library workspace, go to Overview/Application Management/Applications.

4. On the Ribbon, on the **Home** tab, in the **Create** group, select **Create Application**.

   The Create Application Wizard starts.

5. Complete the Create Application Wizard using the information in Table 33. Accept default values unless otherwise specified.

Table 3. Information for Completing the Create Application Wizard

Expand table

On this wizard page	Do this
General	Select **Manually specify the application information**, and then select **Next**.
General: General	1. In **Name**, type **Microsoft Office Professional Plus 2010 - x86**. 2. In **Administrator comments**, type **32-bit version of Microsoft Office Professional Plus 2010**. 3. Select the **Allow this application to be installed from Install Application task sequence action instead of deploying it manually** check box. 4. Select **Next**.
General: Application Catalog	1. In **Localized description**, type **32-bit version of Microsoft Office Professional Plus 2010 for use by Woodgrove Bank Employees**. 2. In **Keywords**, **type Office Professional Plus 2010**. 3. Select **Next**.
General: Deployment Type s	a. Select **Add**. The Create Deployment Type Wizard Starts. b. In the **Create Deployment Type Wizard**, on the **General** page, select **Manually specify the deployment type information** and then select **Next**. c. On the **General: General Information** page, perform the following steps, and then select **Next**: i. In **Name**, type **Microsoft Office Professional Plus 2010 - x32 (Windows Installer)**. ii. In **Administrator comments**, type **Deploy Microsoft Office Professional Plus 2010 using native Windows Installer**. d. On the **General: Content** page, perform the following steps, and then select **Next**: i. In **Content location**, type **\\WDGMDT01\Source$\OfficeProPlus2010**. ii. In **Installation program**, type **setup.exe**. iii. In **Uninstall program**, type **setup.exe /uninstall PROPLUS**. e. On the **General: Detection Method** page, perform the following steps, and then select **Next**: i. Select Add **Clause**, The **Detection Rule** dialog box appears. ii. In the **Detection Rule** dialog box, in **Setting Type**, select **Windows Installer**. iii. In **Product code**, select **Browse** The **Open** dialog box appears. iv. In the **Open dialog** box, in **File name**, type **\\WDGMDT01\Source$\OfficeProPlus2010\ProPlus.WW\ProPlusWW.msi**,

On this wizard page	Do this
	and then select **Open**. The product code for Office Professional Plus 2010 appears in the **Product code** box. v. In the **Detection Rule** dialog box, select **OK**. f. On the **General: User Experience** page, perform the following steps, and then select **Next**: i. In **Installation behavior**, select **Install for system**. ii. In **Logon requirement**, select **Whether or not a user is logged on**. iii. In **Installation program visibility**, select **Normal**. iv. In **Estimated installation time**, type **120**. g. On the **Requirements** page, select **Next**. h. On the **Dependencies** page, select **Next**. i. On the **Summary** page, select **Next**. j. On the **Completion** page, select **Close**. The Create Application Wizard starts. k. Select **Next**.
Summary	Review the information in the **Details** box that you provided while completing the previous wizard pages, and then select **Next**.
Progress	The progress for creating the application is displayed.
Completion	Select **Close**.

The Office Professional Plus 2010 - x86 application appears in the preview pane.

Step 5-9: Distribute the Office Professional Plus 2010 Application

After you have created the Office Professional Plus 2010 application, you need to distribute the application to the distribution points. Doing so allows installation of the application from the distribution points. For the purposes of this guide, there is only one distribution point (WDG-MDT-01). In typical Configuration Manager deployments, there are usually multiple distribution points.

To distribute the Office Professional Plus 2010 application

1. Select **Start**, point to **All Programs**, and then point to **Microsoft System Center 2012**. Point to **Configuration Manager**, and then select **Configuration Manager Console**.

2. In the Configuration Manager console, in the navigation pane, select **Software Library**.
3. In the Software Library workspace, go to Overview/Application Management/Applications.
4. In the preview pane, select **Microsoft Office Professional Plus 2012 - x86**.
5. On the Ribbon, on the **Home** tab, in the **Deployment** group, select **Distribute Content**.

   The Distribute Content Wizard starts.
6. Complete the Distribute Content Wizard using the information in Table 34. Accept the default values unless otherwise specified.

Table 34. Information for Completing the Distribute Content Wizard

Expand table

On this wizard page	Do this
General	Select **Next**.
General: Content	Select **Next**.
General: Content Destination	1. Select **Add**, and then select **Distribution Point**. The **Add Distribution** Points dialog box appears. 2. In the **Add Distribution Points** dialog box, select **\\WDGMDT01.mdt2013.corp.woodgrovebank.com**, and then select **OK**. \\WDGMDT01.mdt2013.corp.woodgrovebank.com appears in the Content destination list. 3. Select **Next**.
Summary	Review the information in the **Details** box that you provided while completing the previous wizard pages, and then select **Next**.
Progress	The progress for distributing the application is displayed.
Completion	Select **Close**.

7. Close all open windows and dialog boxes.

Step 5-10: Make the Office Professional Plus 2010 Application Available to All Users

After you have created the Office Professional Plus 2010 application, you need to distribute the application to the distribution points. Doing so allows installation of the application from the distribution points. For the purposes of this guide, there is only one distribution point (WDG-MDT-01). In typical Configuration Manager deployments, there are usually multiple distribution points.

To make the Office Professional Plus 2010 application available to all users

1. Select **Start**, point to **All Programs**, and then point to **Microsoft System Center 2012**. Point to **Configuration Manager**, and then select **Configuration Manager Console**.
2. In the Configuration Manager console, in the navigation pane, select **Software Library**.
3. In the Software Library workspace, go to Overview/Application Management/Applications.
4. In the preview pane, select **Microsoft Office Professional Plus 2010 - x86**.
5. On the Ribbon, on the **Home** tab, in the **Deployment** group, select **Deploy**.

   The Deploy Software Wizard starts.
6. Complete the Deploy Software Wizard using the information in Table 35. Accept the default values unless otherwise specified.

Table 35. Information for Completing the Deploy Software Wizard

Expand table

On this wizard page	Do this
General	1. In **Collection**, select **Browse**. The **Select Collection** dialog box appears. 2. In the **Select Collection** dialog box, select **All Users**, and then select **OK**. 3. In **Comments**, type **Make Microsoft Office Professional Plus 2010 available for deployment to all users**. 4. Select **Next**.

On this wizard page	Do this
Content	Select **Next**.
Deployment Settings	Select **Next**.
Scheduling	Select **Next**.
Alerts	Select **Next**.
Summary	Review the information in the **Details** box that you provided while completing the previous wizard pages, and then select **Next**.
Progress	The progress for deploying the application is displayed.
Completion	Select **Close**.

7. Close all open windows and dialog boxes.

Step 5-11: Customize the UDI Wizard Configuration File for the Target Computer

The User-Driven Installation task sequence template includes a task sequence step that runs the UDI Wizard. When a task sequence step runs the UDI Wizard, the step also references an XML file that determines the configuration of the UDI Wizard. The UDIWizard_Config.xml file in the Scripts folder controls the behavior of the UDI Wizard. Customize the UDIWizard_Config.xml file using the UDI Wizard Designer.

The UDI Wizard Designer includes predefined stage groups for the UDI Wizard listed in Table 36. You can add or remove the wizard pages that appear in the UDI Wizard and the sequence of each wizard page for each stage group.

Table 36. Predefined Stage Groups for Each Supported MDT Deployment Scenario

Expand table

Stage group	Description
New Computer	Use this stage group as the basis for your deployment when a new installation of a Windows operating system is deployed to a new computer and no user state is migrated.

Stage group	Description
Refresh	Use this stage group as the basis for your deployment when a computer is refreshed, including computers that must be re-imaged for image standardization or to address a problem.
Replace	Use this stage group as the basis for your deployment when one computer replaces another computer. The existing user state migration data is saved from the original computer. Then, a new installation of Windows is deployed to a new computer. Finally, the user state data is restored to the new computer.

To customize the UDI Wizard configuration file for the reference computer

1. Select **Start**, point to **All Programs**, point to **Microsoft Deployment Toolkit**, and then select **UDI Wizard Designer**.

   The UDI Wizard Designer starts.

2. On the Ribbon, on the **Home** tab, in the **File Menu** group, select **Open**.

3. In the **Open** dialog box, in **File name**, type **\\WDG-MDT-01\Packages$\MDT_Files\Scripts\UDIWizard_Config.xml**, and then select **Open**.

   > ⓘ **Note**
   >
   > This opens the copy of the UDIWizard_Config.xml file that resides in the MDT Package folder you created when you ran the Create Microsoft Deployment Task Sequence Wizard earlier in the process.

4. In the Page Library, select **Install Programs**.

5. On the Ribbon, on the **Home** tab, in the **Edit Settings** group, select **Configuration Manager**.

   The **Site Settings** dialog box appears.

6. In the **Site Settings dialog** box, perform the following steps, and then select **OK**:

   a. In **Site Server Name**, type **WDG-MDT-01**.

   b. In **Site Code**, type **NYC**.

   c. Select **Validate Site**.

   d. In **Application Collection**, type **All Users**.

> **Note**
>
> The Configuration Manager collection you type here must match the Configuration Manager collection to which you deployed your applications. In this guide, you selected the All Users collection in Step 5-10: Make the Office Professional Plus 2010 Application Available to All Users.

7. In the preview pane, on the **Flow** tab, expand **StageGroup: New Computer**.

   The list of wizard pages used in the StageGroup: New Computer flow is displayed.

> **Note**
>
> Make note of the sequence of the wizard pages in the StageGroup: New Computer flow in the UDI Wizard Designer. You will see the same sequence of wizard pages when you run the UDI Wizard in Step 6-4: Start the Target Computer with the Task Sequence Bootable Media.

8. Configure the **StageGroup: New Computer** flow using the information for each page listed in Table 37. Accept the default values unless otherwise specified.

Table 37. Information for Configuring UDI Wizard Designer Pages

Expand table

Wizard page	Select the Configure tab and do the following
BitLocker	a. Under **BitLocker Mode**, expand **BitLocker Mode**. In **BitLocker Checkbox**, clear the **Initially check this check box** check box. b. Under **BitLocker Mode**, select **Unlocked** for each of the following configuration options: • **BitLocker Checkbox** • **BitLocker Mode Radio Buttons** • **PIN Text Box** The status for each configuration option changes to **Locked**, which prevents users from changing these options in the UDI Wizard.

Wizard page	Select the Configure tab and do the following
Volume	a. Under **Image Combo Box**, expand **Image Combo Behavior**, under **Image Combo Box Values**, right-click **Windows 8.1 RTM (x86)**, and then select Select an **Operating System Image**. The **Select an Operating System Image** dialog box appears. b. Complete the **Select an Operating System Image** dialog box by performing the following steps, and then select **OK**: i. In **Select an Operating System Image/Installer to add**, select ***image_index*** (where *image_index* is the image index of the image that contains Windows 8.1, which was identified in Step 5-1: Import the Captured .wim File into Configuration Manager; for the purposes of this guide, select **2**). ii. In **Display Name**, type **Windows 8.1 Reference Image - x64**. c. Under **Image Combo Box**, expand **Image Combo Behavior**; under **Image Combo Box Values**, right-click **Windows 8.1 RTM (x86)**, and then select **Remove Item**. The **Delete Item Confirmation** dialog box appears. d. In the **Delete Item Confirmation** dialog box, select **Yes**. e. Under **User Data and Settings**, expand **User Data Combo Behavior**, and then select the **Format: Clean all the data on the target volume during install** check box. f. Under **User Data Combo Behavior**, select **Unlocked** for each of the following configuration options: • **Format Drive** • **Windows Directory** The status for each configuration option changes to **Locked**, which prevents users from changing these options in the UDI Wizard.
New Computer Details	1. Under **Network Details**, expand **Network Details**; in **Domain or Workgroup Radio Buttons**, select **Domain**. 2. Under **Domain or Workgroup Radio Buttons**, select **Unlocked**. The status changes to **Locked**, which prevents users from changing this option in the UDI Wizard. 3. Under **Network Details**, expand **Domains and OUs**, and then select **Add Domain**. The **Create or Edit Domain Information** dialog box appears. 4. In the **Create or Edit Domain Information** dialog box, in **Domain Name** type **mdt2013.corp.woodgrovebank.com**. 5. In the **Create or Edit Domain Information** dialog box, in **Friendly Name**, type **Woodgrove Bank Active Directory Domain**, and then select **OK**.

Wizard page	Select the Configure tab and do the following
Install Programs	a. Under **Software and Groups**, right-click any blank area, and then select **Add Software Group**. The **Add/Edit a Software Group** dialog box appears. b. In the **Add/Edit a Software Group** dialog box, in **Name**, type **Woodgrove Bank Applications**, and then select **OK**. c. Under **Software and Groups**, select **Woodgrove Bank Applications**. d. On the Ribbon, on the **Home** tab, in the **General Software Item Settings** group, select **Add**, and then select **Add Software to Group**. The Add Software to Group Wizard starts. e. Complete the Add Software to Group Wizard by performing the following steps: i. On the **What type of software item do you want to add** page, select **I want to add an Application**, and then select **Next**. ii. On the **Search Configuration Manager for the Software Item to Add** page, in **Display Name**, type **Microsoft Office Professional Plus 2010 - x86**. iii. On the **Search Configuration Manager for the Software Item to Add** page, select **Select**. The **Search Applications** dialog box appears. iv. In the **Search Applications** dialog box, select **Search**, select **Microsoft Office Professional Plus 2010 - X86**, and then select **OK**. v. On the **Search Configuration Manager for the Software Item to Add page**, select **Finish**. **Microsoft Office Professional Plus 2010 - x86** appears underneath the **Woodgrove Bank Applications** software group. f. Under **Software and Groups**, select **General Software**. g. On the Ribbon, on the **Home** tab, in the **General Software Item Settings** group, select **Add**, and then select **Remove Item**. The **Delete the Selected Item** dialog box appears. h. In the **Delete the Selected Item** dialog box, select **Yes**. i. Under **Software and Groups**, select the check box for **Woodgrove Bank Applications**. The group and Microsoft Office Professional Plus 2010 - x86 are selected.

9. On the Ribbon, on the **Home** tab, select **Save**.

   The **File Save** dialog box appears.

10. In the **File Save** dialog box, select **OK**.

11. Leave the UDI Wizard Designer open for the next step.

Step 5-13: Create a New Custom Wizard Page

You can create custom wizard pages that allow you to collect deployment information in addition to the information collected on other UDI Wizard pages. You create custom wizard pages based on the **Build Your Own Page** wizard page type. After you create the custom wizard page, you can add controls to it and configure the task sequence variables the controls set.

For this guide, Woodgrove Bank wants to allow users to enter their name and the department in which they work. Woodgrove Bank is departmentalized by geographic location. This information will be used to configure the registered user name and organization in Windows. In this step, you add a new custom wizard page to the New Computer stage group.

To create a new custom wizard page

1. On the Ribbon, on the **Home** tab, in the **Page Library** group, select **Add Page**.The **Add New Page** dialog box appears.
2. In the **Add New Page** dialog box, in the **Page Type** column, select **Build Your Own Page**.
3. In **Display Name**, type **User Information**.
4. In **Page Name**, type **UserInformationPage**, and then select **OK**.

   The **User Information** page appears in the Page Library.
5. In the details pane, select the **Flow** tab.
6. On the **Flow** tab, expand the New Computer stage group.

   The list of wizard pages in the New Computer stage group is displayed.
7. In the Page Library, drag the **User Information** page to a point immediately before the **BitLocker** page in the New Computer stage group on the **Flow** tab.
8. On the Ribbon, on the **Home** tab, select **Save**.

   The **File Save** dialog box appears.
9. In the **File Save** dialog box, select **OK**.
10. Leave the UDI Wizard Designer open for the next step.

Step 5-14: Add Controls to New Custom Wizard Page

After the new UDI custom wizard page has been added to the New Computer stage group, the appropriate controls need to be added to the new custom wizard page. The controls are added to the custom wizard page from the Build Your Own Page toolbox, which is displayed when you view the custom wizard page on the **Configure** tab in the UDI Wizard Designer.

Table 38 lists the types of controls to your custom wizard page, which is illustrated in Figure 1.

Table 38. Types of Controls in the UDI Build Your Own Page Toolbox

Expand table

Control type	Description
Checkbox	This control allows you to select or clear a configuration option and behaves as a traditional user interface (UI) check box. This control has a corresponding label that you can use to describe the purpose of the check box. The state of this control is True when the check box is selected and False when the check box is cleared. The state of the check box is stored in the task sequence variable configured for this control. For more information on this control, see "Checkbox Control" in the MDT document, *Toolkit Reference*.
Combobox	This control allows you to select an item from a list of items and behaves as a traditional UI drop-down list. This control allows you to add or remove items from the list and provide a corresponding value that will be set in the task sequence variable configured for this control. For more information on this control, see "Combobox Control" in the MDT document, *Toolkit Reference*.
Line	This control allows you to add a horizontal line to divide one portion of the custom wizard page from another. This control does not collect any configuration values but rather is used to visually enhance the UI. For more information on this control, see "Line Control" in the MDT document, *Toolkit Reference*.
Label	This control allows you to add descriptive, read-only text to the wizard page. This control does not collect any configuration values but rather is used to visually enhance the UI. For more information on this control, see "Label Control" in the MDT document, *Toolkit Reference*.
Radio	This control allows you to select one configuration option from a group of two or more options. As with traditional radio buttons, two or more of these controls can be grouped together, and then the user can select one of the options in the radio button group. A unique value is assigned to each option. The value assigned to the

Control type	Description
	selected option control is saved in the task sequence variable configured for this control. For more information on this control, see "Radio Control" in the MDT document, *Toolkit Reference*.
Bitmap	This control allows you to add a bitmap graphic (.bmp file) to the custom wizard page. This control does not collect any configuration values but rather is used to visually enhance the UI. The path to the .bmp file is relative to the location of the UDI Wizard (OSDSetupWizard.exe). For more information on this control, see "Bitmap Control" in the MDT document, *Toolkit Reference*.
Textbox	This control allows you to enter text on the custom wizard page. The text typed into this control is saved in the task sequence variable configured for this control. For more information on this control, see "Textbox Control" in the MDT document, *Toolkit Reference*.

You can add any combination of these controls to your custom wizard page based on the information you want to collect. In addition, you can use the **Show Gridlines** check box to show or hide gridlines that can be used to assist in visually designing the custom wizard page.

For the purposes of this example, you will create a custom wizard page as illustrated in Figure 1.

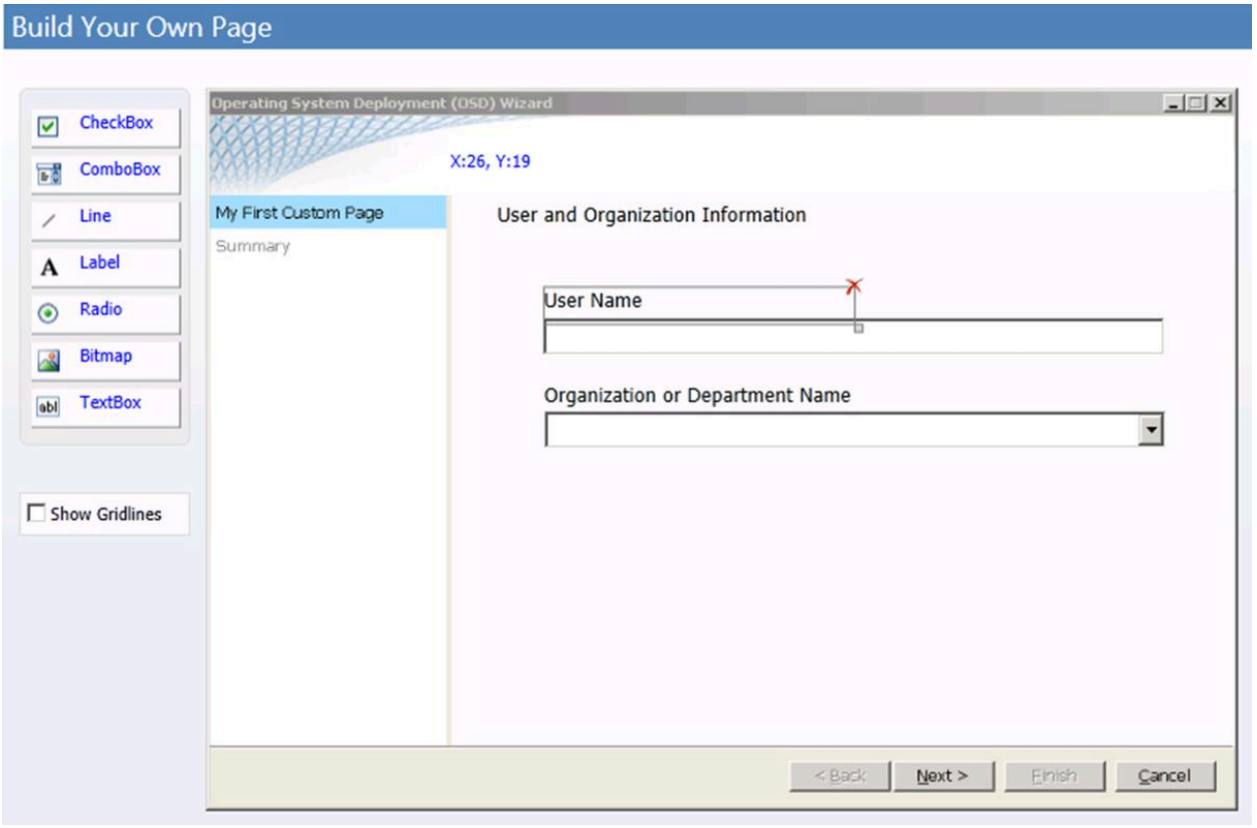


Figure 1. Custom wizard page to be created

Figure 1. Custom wizard page to be created

To add controls to the new custom wizard page

1. In the Page Library, select **User Information** page.
2. In the details pane, select the **Configure** tab.

   The Build Your Own Page toolbox and empty wizard page are displayed.
3. In the Build Your Own Page toolbox, drag the **Label** control to the empty wizard page at approximately the following coordinates:

   - *x* = 30
   - *y* = 5

     The label control is placed on the wizard page and named *label1*.
4. On the custom wizard page, select **label1** (the label control added in step 3).

   This control acts as a heading for the custom wizard page and describes the purpose of the page.
5. Configure the layout properties of **label1** on the **Layout** tab using the information in Table 39. Accept the default values unless otherwise stated.

Table 39. label1 Layout Properties

Expand table

Property	Value
Label	User and organization information
X	30
Y	5

6. In the Build Your Own Page toolbox, drag the **Label** control to the empty wizard page at approximately the following coordinates:

   - *x* = 60
   - *y* = 60

     The label control is placed on the wizard page and named *label2*.
7. On the custom wizard page, select **label2** (the control added in the previous step).

This control acts as a label for the text box used to enter the user name.

8. Configure the layout properties of **label2** on the **Layout** tab using the information in Table 40. Accept the default values unless otherwise stated.

Table 40. lable2 Layout Properties

Expand table

Property	Value
Label	User name
X	60
Y	60

9. In the Build Your Own Page toolbox, select and drag the **Textbox** control to the empty wizard page at approximately the following coordinates:

   - *x* = 60
   - *y* = 80

   The **Textbox** control is placed on the wizard page and named *text1*.

10. On the custom wizard page, select **text1** (the control added in the previous step).

    This control is the text box used to enter the user name.

11. Configure the layout properties of **text1** on the **Layout** tab using the information in Table 41. Accept the default values unless otherwise stated.

Table 41. text1 Layout Properties

Expand table

Property	Value
X	60
Y	80
Width	400

12. Configure the settings properties of **text1** on the **Settings** tab using the information in Table 42. Accept the default values unless otherwise stated.

Table 42. text1 Settings Properties

Expand table

Property	Value
Task sequence variable name	**FullName**
Friendly display name visible in summary page	Registered user name

13. In the Build Your Own Page toolbox, drag the **Label** control to the empty wizard page at approximately the following coordinates:

    - *x* = 60
    - *y* = 60

    The **Label** control is placed on the wizard page and named *label3*.

14. On the custom wizard page, select **label3** (the control added in the previous step).

    This control acts as a label for the combo box used to select the organization or department name for the user.

15. Configure the layout properties of **lable3** on the **Layout** tab using the information in Table 43. Accept the default values unless otherwise stated.

Table 43. lable3 Layout Properties

Expand table

Property	Value
Label	Organization or department name
X	60
Y	121

16. In the Build Your Own Page toolbox, drag the **Combobox** control to the empty wizard page at approximately the following coordinates:

    - *x* = 60

- *y* = 140

The **Combobox** control is placed on the wizard page and named *combo1*.

17. On the custom wizard page, select **combo1** (the control added in the previous step).

    This control is the combo box used to select the organization name.

18. Configure the layout properties of **combo1** on the **Layout** tab using the information in Table 44. Accept the default values unless otherwise stated.

Table 44. combo1 Layout Properties

Expand table

Property	Value
X	60
Y	80
Width	400

19. Add data items to the layout properties of **combo1** on the **Layout** tab using the information in Table 45. Accept the default values unless otherwise stated.

Table 45. combo1 Data Items

Expand table

Value	Display Value
Woodgrove Bank - New York City	Woodgrove Bank - New York City
Woodgrove Bank - Dallas	Woodgrove Bank - Dallas
Woodgrove Bank - Chicago	Woodgrove Bank - Chicago
Woodgrove Bank - Seattle	Woodgrove Bank - Seattle

20. Configure the settings properties of **combo1** on the **Settings** tab using the information in Table 46. Accept the default values unless otherwise stated.

Table 46. combo1 Settings Properties

Expand table

Property	Value
Task sequence variable name	**OrgName**
Friendly display name visible in summary page	Registered organization name

21. On the Ribbon, on the **Home** tab, select **Save**.

    The **File Save** dialog box appears.

22. In the **File Save** dialog box, select **OK**.

23. Close the UDI Wizard Designer.

Step 5-15: Update the Distribution Points for the MDT Files Package

After the UDI Wizard configuration file, UDIWizard_Config.xml, has been updated for the MDT Files package in Configuration Manager, update the distribution points for the MDT Files package. Updating the distribution points copies the updated version of the UDIWizard_Config.xml file to the deployment shares specified in the package.

To update the distribution points for the MDT Files package

1. Select **Start**, point to **All Programs**, and then point to **Microsoft System Center 2012**. Point to **Configuration Manager**, and then select **Configuration Manager Console**.

2. In the Configuration Manager console, in the navigation pane, select **Software Library**.

3. In the Software Library workspace, go to Overview/Application Management/Packages.

4. In the preview pane, select **MDT Files**.

5. On the Ribbon, on the **Home** tab, in the **Deployment** group, select **Update Distribution Points**.

   The **Configuration Manager** dialog box opens, notifying you that you are going to update the package on all distribution points.

6. In the **Configuration Manager** dialog box, select **OK**.

7. Close all open windows and dialog boxes.

   Configuration Manager starts updating the distribution points with the latest versions of the UDIWizard_Config.xml file. This process could take several minutes. Check the status of the package until the **Last Update** value of the package status has been updated to a recent date and time.

Step 6: Deploy the Captured Image of the Reference Computer to the Target Computer

When you have captured the image of the reference computer and created and configured the task sequence, deploy the captured image. Configure MDT to provide all the necessary configuration settings to deploy to the target computer. After initiating the deployment process, the image of the reference computer running Windows 8.1 is automatically deployed to the target computer and configured with the settings defined.

Deploy the captured image by:

- Adding the target computer to the Configuration Manager site database as described in Step 6-1: Add the Target Computer to the Configuration Manager Site Database
- Creating a computer collection that includes the target computer as described in Step 6-2: Create a Computer Collection That Includes the Target Computer
- Deploy the task sequence created earlier in the process as described in Step 6-3: Deploy the Target Computer Task Sequence
- Starting the target computer with the task sequence bootable media as described in Step 6-4: Start the Target Computer with the Task Sequence Bootable Media

Step 6-1: Add the Target Computer to the Configuration Manager Site Database

To deploy an operating system without stand-alone media to a new computer that Configuration Manager does not currently manage, add the new computer to the Configuration Manager site database prior to initiating the operating system deployment process. Configuration Manager can automatically discover computers on the network that have a Windows operating system installed; however, if the computer has no operating system installed, use the Import Computer Information Wizard to import the new computer information.

To add the target computer to the Configuration Manager site database

1. Select **Start**, point to **All Programs**, and then point to **Microsoft System Center 2012**. Point to **Configuration Manager**, and then select **Configuration Manager Console**.

2. In the Configuration Manager console, in the navigation pane, select **Assets and Compliance**.

3. In the Assets and Compliance workspace, go to Overview/Devices.

4. On the Ribbon, on the **Home** tab, in the **Create** group, select **Import Computer Information**.

   The Import Computer Information Wizard starts.

5. Complete the Import Computer Information Wizard using the information in Table 47. Accept the default values unless otherwise specified.

Table 47. Information for Completing Import Computer Information Wizard

Expand table

On this wizard page	Do this
Select Source	Select **Import single computer**, and then select **Next**.
Select Source: Single Computer	1. In **Computer Name**, type **WDG-CLI-01**. 2. In *MAC address*, type ***mac_address*** (where *mac_address* is the MAC address of the primary network adapter for the target computer, WDG-CLI-01). 3. Select **Next**.
Select Source: Data Preview	Select **Next**.
Select Source: Choose Target Collection	Select **Next**.
Summary	Review the information in the **Details** box that you provided while completing the previous wizard pages, and then select **Next**.
Progress	The progress for importing the computer is displayed.
Confirmation	Select **Close**.

For more information on adding a new computer to the Configuration Manager site database, see the section, "To import computer information for a single computer," in the section, "How to Deploy Operating Systems in Configuration Manager," in the Configuration Manager Documentation Library, which is installed with Configuration Manager.

Step 6-2: Create a Computer Collection That Includes the Target Computer

In the Configuration Manager console, create a collection that includes the target computer (WDG-CLI-01). You use this computer collection later when advertising the task sequence created earlier in the process.

To create a computer collection that includes the target computer

1. Select **Start**, point to **All Programs**, and then point to **Microsoft System Center 2012**. Point to **Configuration Manager**, and then select **Configuration Manager Console**.
2. In the Configuration Manager console, in the navigation pane, select **Assets and Compliance**.
3. In the Assets and Compliance workspace, go to Overview/Device Collections.
4. On the Ribbon, on the **Home** tab, in the **Create** group, select **Create Device Collection**.

   The Create Device Collection Wizard starts.
5. Complete the Create Device Collection Wizard using the information in Table 48. Accept the default values unless otherwise specified.

Table 48. Information for Completing the Create Device Collection Wizard

Expand table

On this wizard page	Do this
General	a. In **Name**, type **Microsoft Deployment - Batch 01**. b. In **Comment**, type **Computers that are to be included in the first batch of computers deployed**.

On this wizard page	Do this
	c. In **Limited Collection**, select **Browse**. The **Browse Collections** dialog box appears. Complete the dialog box by performing the following steps: i. In the **Browse Collection** dialog box, in **Name**, select **All Systems**. ii. Select **OK**. d. Select **Next**.
Membership Rules	a. Select **Add Rule**, and then select **Direct Rule**. The Create Direct Membership Rule Wizard starts. b. Complete the Create Direct Membership Rule Wizard by performing the following steps: i. On the **Welcome** page, select **Next**. ii. On the **Search for Resources** page, in **Resource class**, select **System Resource**; in **Attribute name**, select **Name**; in **Value**, type **WDG-CLI-01**; and then select **Next**. iii. On the **Select Resources** page, select **WDG-CLI-01**, and then select **Next**. **Note:** The process for adding the target computer (WDG-CLI-01) to All Systems can take a few minutes to finish. If WDG-CLI-01 does not appear in the list, repeat steps b and c until WDGCLI01 appears. iv. On the **Summary** page, select **Next**. v. On the **Completion** page, select **Close**. c. Select **Next**.
Summary	Review the information in the **Details** box that you provided while completing the previous wizard pages, and then select **Next**.
Progress	The progress for creating the device collection is displayed.
Completion	Select **Close**.

For more information, see the section, "How to Create Collections in Configuration Manager," in the Configuration Manager Documentation Library, which is installed with Configuration Manager.

Step 6-3: Deploy the Target Computer Task Sequence

In the Configuration Manager console, deploy the task sequence created earlier in the process for the target computers. Deploy the task sequence to the collection of target computers created earlier in the process.

To deploy the task sequence

1. Select **Start**, point to **All Programs**, and then point to **Microsoft System Center 2012**. Point to **Configuration Manager**, and then select **Configuration Manager Console**.
2. In the Configuration Manager console, in the navigation pane, select **Software Library**.
3. In the Software Library workspace, go to Overview/Operating Systems/Task Sequences.
4. In the preview pane, select **UDI - Windows 8.1 Target Deployment**.
5. On the Ribbon, on the **Home** tab, in the **Deployment** group, select **Deploy**.

   The Deploy Software Wizard starts.
6. Complete the Deploy Software Wizard using the information in Table 49. Accept the default values unless otherwise specified.

Table 49. Information for Completing the Deploy Software Wizard

Expand table

On this wizard page	Do this
General	1. In **Collection**, select **Browse**. 2. In the **Browse Collection** dialog box, select **Microsoft Deployment - Batch 01**, and then select **OK**. 3. In **Comment**, type **Deploy Windows 8.1 to the first batch of target computers using UDI**. 4. Select **Next**.
Deployment Settings	1. In **Purpose**, select **Available**. 2. Select the **Make available to boot media and PXE** check box. 3. Select **Next**.
Deployment Settings: Schedule	Select **Next**.
Deployment Settings: User Experience	Select **Next**.
Deployment Settings: Alerts	Select **Next**.

On this wizard page	Do this
Deployment Settings: Distribution Points	Select **Next**.
Summary	Review the information in the **Details** box that you provided while completing the previous wizard pages, and then select **Next**.
Progress	The progress for creating the deploying the task sequence is displayed.
Completion	Select **Close**.

For more information, see the section, "How to Deploy a Task Sequence," in the Configuration Manager Documentation Library, which is installed with Configuration Manager.

Step 6-4: Start the Target Computer with the Task Sequence Bootable Media

Start the target computer (WDG-CLI-01) with the task sequence bootable media created earlier in the process. This medium starts Windows PE on the reference computer and initiates the MDT process. At the end of the MDT process, Windows 8.1 is deployed on the target computer.

> ⓘ **Note**
>
> You can also initiate the MDT process by starting the target computer from Windows Deployment Services.

To start the target computer with the task sequence bootable media

1. Start WDG-CLI-01 with the task sequence bootable media created earlier in the process.

   Windows PE starts, and then the Task Sequence Wizard starts.

2. Complete the Task Sequence Wizard using the information in Table 50. Accept the default values unless otherwise specified.

Table 50. Information for Completing the Task Sequence Wizard

Expand table

On this wizard page	Do this
Welcome to the Task Sequence Wizard	In **Password**, type **P@ssw0rd**, and then select **Next**.
Select a Task Sequence	In the list box, select **UDI - Windows 8.1 Target Deployment**, and then select **Next**.

At the appropriate task sequence step, the UDI Deployment Wizard starts.

3. Complete the UDI Deployment Wizard using the information in Table 51. Accept the default values unless otherwise specified.

Table 51. Information for UDI Deployment Wizard

Expand table

On this wizard page	Do this
Welcome	Select **Next**.
User Information	1. In **User Name,** type **Woodgrove Bank Chicago Employee**. 2. In **Organization or Department Name**, select **Woodgrove Bank - Chicago**. 3. Select **Next**.
BitLocker	Select **Next**.
Volume	Select **Next**.
Select Target	Select **Next**.
Deployment Readiness	1. Review the configuration checks, and ensure that the status for all checks are set to **Success**. 2. Select *Next*.
New Computer Details	1. In **Computer name**, type **WDG-CLI-01**. **Note:** In unknown computer scenarios, users could change the computer name to the appropriate value. 2. In **User name**, type **MDT2013\Administrator**. 3. In **Password** and **Confirm password**, type **P@ssw0rd**. 4. Select **Next**.
Administrator Password	1. In **Administrator password** and **Confirm password**, type **P@ssw0rd**. 2. Select **Next**.

On this wizard page	Do this
User Device Affinity	Select the **Set primary user** check box, and then select **Next**.
Language	Select **Next**.
Install Programs	Verify that the **Microsoft Office Professional Plus 2010 - x86** check box is selected, and then select **Next**.
Summary	Review the information that you provided while completing the previous wizard pages, and then select **Finish**.

To monitor the reference computer deployment process using the Deployment Workbench

4. On WDG-MDT-01, select **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then select **Deployment Workbench**.
5. In the Deployment Workbench console tree, go to Deployment Workbench/Deployment Shares/MDT Deployment Share (C:\DeploymentShare$)/Monitoring.
6. In the details pane, view the deployment process for WDG-REF-01.
7. In the Actions pane, periodically select **Refresh**.

   The status of the deployment process is updated in the details pane. Continue to monitor the deployment process until the process is complete.
8. In the details pane, select **WDG-REF-01**.
9. In the Actions pane, select **Properties**.

   The **WDG-REF-01 Properties** dialog box is displayed.
10. In the **WDG-REF-01 Properties** dialog box, on the **Identity** tab, view the monitoring information provided about the deployment process as described in Table 52.

Table 52. Monitoring Information About the Deployment Process

Expand table

Information	Description
ID	Unique identifier for the computer being deployed.
Computer Name	The name of the computer being deployed.
Deployment status	The current status of the computer being deployed; the status can be one of the following: - **Running**. The task sequence is healthy and running. - **Failed**. The task sequence failed, and the deployment process was unsuccessful. - **Completed**. The task sequence has finished. - **Unresponsive**. The task sequence has not updated its status in the past four hours and is assumed to be nonresponsive.
Step	The current task sequence step being run.
Progress	The overall progress of the task sequence. The progress bar indicates how many task sequence steps have been run out of the total number of task sequence steps.
Start	The time the deployment process started.
End	The time the deployment process ended.
Elapsed	The length of time the deployment process has been running or took to run if the deployment process has finished.
Errors	The number of errors encountered during the deployment process.
Warnings	The number of warnings encountered during the deployment process.
Remote Desktop	This button allows you to establish a remote desktop connection with the computer being deployed using the Windows Remote Desktop feature. This method assumes that: - The target operating system is running and has remote desktop support enabled - **mstsc.exe** is in the path **Note:** This button is always visible but may not be able to establish a remote desktop session if the monitored computer is running Windows PE, has not completed installation of the target operating system, or does not have the Remote Desktop feature enabled.
VM Connection	This button allows you to establish a remote desktop connection to a VM running in HyperV. This method assumes that: - The deployment is being performed to a VM running on Hyper-V - **vmconnect.exe** is located in the %ProgramFiles%\Hyper-V folder

Information	Description
	Note: This button appears when ZTIGather.wsf detects that Hyper-V integration components are running on the monitored computer. Otherwise, this button will not be visible.
DaRT Remote Control	This button allows you to establish a remote control session using the remote viewer feature in DaRT. This method assumes that: - DaRT has been deployed to the target computer and is currently running - **DartRemoteViewer.exe** is located in the %ProgramFiles%\Microsoft DaRT 7\v7 folder **Note:** This button appears when ZTIGather.wsf detects that DaRT is running on the monitored computer. Otherwise, this button will not be visible.
Automatically refresh this information every 10 seconds	Check box that controls whether the information in the dialog box is automatically refreshed. If the check box is: - Selected, the information is refreshed every 10 seconds - Cleared, the information is not automatically refreshed and must be manually refreshed using the **Refresh** Now button
Refresh Now	This button immediately refreshes the information displayed in the dialog box.

11. In the **WDG-REF-01 Properties** dialog box, select **OK**.

12. Close the Deployment Workbench.

    To monitor the reference computer deployment process using the Get-MDTMonitorData cmdlet

13. On WDG-MDT-01, select **Start**, point to **Administrative Tools**, and then select **Windows PowerShell Modules**.

    The Windows PowerShell Modules command prompt opens.

14. Create a Windows PowerShell drive that uses the MDT PowerShell provider by running the New-PSDrive cmdlet as shown in the following example:

```
PowerShell
```

```
New-PSDrive -Name DS001 -PSProvider mdtprovider -Root
d:\DeploymentShare$
```

15. View the MDT monitoring process by running the **Get-MDTMonitorData** cmdlet, as shown in the following example:

```
PowerShell

Get-MDTMonitorData -Path DS001:
```

This command returns the monitoring data collected by the MDT monitoring service running on the same computer that hosts the deployment share as shown in the following example output:

```
PowerShell

Name               : WDG-REF-01
PercentComplete    : 96
Settings           :
Warnings           : 0
Errors             : 0
DeploymentStatus   : 1
StartTime          : 6/7/2012 6:45:39 PM
EndTime            :
ID                 : 1
UniqueID           : 94a0830e-f2bb-421c-b1e0-6f86f9eb9fa1
CurrentStep        : 130
TotalSteps         : 134
StepName           : Gather
LastTime           : 6/7/2012 8:46:32 PM
DartIP             :
DartPort           :
DartTicket         :
VMHost             : XYL-DC-02
VMName             : WDG-REF-01
ComputerIdentities : {}

Name               : WDG-CLI-01
PercentComplete    : 26
Settings           :
Warnings           : 0
Errors             : 0
DeploymentStatus   : 1
StartTime          : 6/7/2012 3:07:13 AM
EndTime            :
ID                 : 2
UniqueID           : 94a0830e-f2bb-421c-b1e0-6f86f9eb9fa1
CurrentStep        : 49
TotalSteps         : 134
StepName           : Capture Network Settings using MDT
LastTime           : 6/7/2012 3:08:32 AM
DartIP             :
DartPort           :
DartTicket         :
```

```
VMHost             :
VMName             :
ComputerIdentities : {}
```

16. Close the Windows PowerShell console.

    If any problems occur during the deployment, consult the MDT document *Troubleshooting Reference*. When successfully completed, the target computer is running a Windows 8.1 operating system configured like the reference computer.

    At the completion of the deployment process, Windows 8.1 starts for the first time, and the **Welcome** tab in the **Deployment Complete** dialog box is displayed. The **Welcome** tab displays helpful information about the deployment and provides contact information in the event issues with the deployment occur.

    Review the information on the **Deployment Summary** and **Applications Installed** tabs to verify that Windows 8.1 and Office Professional Plus 2010 were installed correctly. When you have finished reviewing these tables, select **Start Windows** to log on to Windows 8.1 for the first time.

> **Note**
>
> Configuration Manager applications are not displayed on the **Applications Installed** tab. Instead, they are detected after the user logs on to the target computer for the first time.

Feedback

Was this page helpful? Yes No

Provide product feedback

Toolkit Reference for the Microsoft Deployment Toolkit

Article • 02/12/2024

This reference is part of Microsoft® Deployment Toolkit (MDT) 2013 and provides configuration settings that you can use in the deployment process. Review the MDT 2013 documents Microsoft Deployment Toolkit Samples Guide and Using the Microsoft Deployment Toolkit for help in customizing configuration settings for the deployment environment.

> **Note**
>
> In this document, *Windows* applies to the Windows 8.1, Windows 8, Windows 7, Windows Server® 2012 R2, Windows Server 2012, and Windows Server 2008 R2 operating systems unless otherwise noted. MDT does not support ARM processor-based versions of Windows. Similarly, *MDT* refers to MDT 2013 unless otherwise stated.

Microsoft® Deployment Toolkit (MDT) 2013 reference articles:

- Task Sequence Steps.
- Properties.
- Scripts.
- Support Files.
- Utilities.
- MDT Windows PowerShell Cmdlets.
- Tables and Views in the MDT DB.
- Windows 7 Feature Dependency Reference.
- UDI Reference.

Feedback

Was this page helpful? Yes No

Provide product feedback

Task Sequence Steps

Article • 02/12/2024

Task sequences are created by the Task Sequence Editor and consist of a combined series of steps that are designed to complete an action. Task sequences can operate across a computer restart and can be configured to automate tasks on a computer without requiring user intervention. In addition, you can add task sequence steps to a task sequence group, which helps keep similar task sequence steps together for better organization and error control.

Each task sequence step performs a specific task, such as validating that the target computer is capable of receiving the deployment image, storing user data in a safe location, deploying an image to a target computer, and restoring saved user data. These task sequence steps accomplish their tasks by using utilities and scripts provided with MDT or by the deployment team. Use this reference to help determine the correct task sequence groups and task sequence steps to configure the deployment process and the valid properties and options to use.

The following information is provided for each task sequence group and step:

- **Name**. The name of the task sequence group or step
- **Description**. A description of the purpose of the task sequence group or step and any pertinent information regarding its customization
- **Properties**. Indicates the valid configuration properties that you can specify for the task sequence group or step that define how the task is performed
- **Options**. Indicates the valid configuration options that you can specify for the task sequence group or step that define if and when the task is performed and what is considered a successful exit code from the task

  For more information about the Task Sequence Editor, see Operating System Deployment: Task Sequence Editor.

Common Properties and Options for Task Sequence Step Types

Each task sequence group and step has configurable settings on the **Properties** and **Options** tabs that are common to all task sequence groups and steps. These common settings are briefly described in the following sections.

Common Properties

Table 1 shows the settings that are available on the **Properties** tab of each task sequence step. For more information about the **Properties** tab for a particular task sequence step, see the topic that corresponds to the step later in this reference.

> **Note**
>
> The task sequence step types listed here are those that are available in the Deployment Workbench. Additional task sequence step types might be available when configuring task sequences using Microsoft System Center 2012 R2 Configuration Manager.

Table 1. Settings Available on the Properties Tab

Expand table

Name	Description	Group	Step
Type	A read-only value that indicates the task sequence group or step type. The type will be set to one of these values: - Apply Network Settings - Authorize DHCP - Capture Network Settings - Configure ADDS - Configure DHCP - Configure DNS - Enable BitLocker - Format and Partition Disk - Gather - Group - Inject Drivers - Install Application	-	-

Name	Description	Group	Step
	- Install Operating System - Install Roles and Features - Install Updates Offline - Recover From Domain Join Failure - Restart computer - Run Command Line - Validate		
Name	A user-defined name that should allow easy identification and differentiation from other task sequence steps.	-	-
Description	A user-defined description that should make the task sequence step requirements and tasks easily understandable.	-	-

Common Options

Table 2 shows the settings that are available on the Options tab of a task sequence step. For more information about the Options tab, see Task Sequence Options Tab.

Table 2. Settings Available on the Options Tab

Expand table

Name	Description	Group	Step
Disable this step	Select this option to disable this task sequence step.	-	-
Success codes	Exit codes of the utility associated with this task sequence step that indicate that the step has finished successfully.		-
Continue on error	Select this option to allow the Task Sequencer to process additional task sequence steps if a failure occurs.	-	-
Conditional statements	One or more conditions that limit the running of this task sequence group or step. These conditional are based on the following: - File properties	-	-

Name	Description	Group	Step
	- Folder properties Operating system version: - Is a certain architecture - Is a certain version - Query Windows Management Instrumentation (WMI) Registry setting: - Exists - Does not exist - Equals - Does not equal - Greater than - Greater than or equals - Less than - Less than or equals - Installed software Task sequence variable: - Exists - Equals - Does not equal - Greater than - Greater than or equals - Less than - Less than or equals These conditions can be grouped using **IF** statements that test		

Name	Description	Group	Step
	all conditions, any condition, or no condition that evaluates as True.		

> **Note**
>
> Additional conditional statements might be available when using Configuration Manager to configure task sequence steps.

Specific Properties and Settings for Task Sequence Step Types

Some properties and parameters of each task sequence step type are unique to that type. Each type with unique properties and settings is shown in the following sections with its unique task sequence step properties and settings.

Apply Network Settings

This task sequence step configures the network adapter on the target computer. For more information about what script accomplishes this task and which properties are used, see ZTINICConfig.wsf.

The unique properties and settings for the **Apply Network Settings** task sequence step type are:

Properties

Expand table

Name	Value
Type	Apply Network Settings

Settings

Expand table

Name	Value
Name	The name to be assigned to the network connection.
Obtain an IP address automatically	When selected, Dynamic Host Configuration Protocol (DHCP) is used to obtain the required Internet Protocol (IP) configuration settings for the network connection. This is the default selection.
Use the following IP address	When selected, you can provide one or more IP address and subnet mask combinations in addition to gateways that will be assigned to the network connection.
Obtain a Domain Name System (DNS) server automatically	When selected, DHCP is used to obtain the required IP configuration settings for the network connection. This is the default selection.
Use the following DNS servers	When selected, you can provide one or more DNS server IP addresses that will be assigned to the network connection.
DNS Suffix	The DNS suffix that will be applied to all network connections that use TCP/IP.
Register this connection's address in DNS	Specifies that the computer will attempt dynamic registration of the IP addresses (through DNS) of this connection with the full computer name of this computer.
Use this connection's DNS suffix in DNS registration	Specifies whether DNS dynamic update is used to register the IP addresses and the connection-specific domain name of this connection.
WINS server addresses	You can provide one or more Windows Internet Naming Service (WINS) server IP addresses that will be assigned to the network connection.
Enable LMHOSTS lookup	Specifies whether a local area network (LAN) Manager Hosts (LMHOSTS) file for network basic input/output system (NetBIOS) name resolution is used.
Default	Specifies whether this network connection obtains the setting to enable or disable NetBIOS over TCP/IP (NetBT) from a DHCP server. This is the default selection.
Enable NetBIOS over TCP/IP	Specifies that this network connection uses NetBT and WINS.
Disable NetBIOS over TCP/IP	Specifies that this network connection does not use NetBT and WINS.

Authorize DHCP

This task sequence step authorizes the target computer as a DHCP server. For more information about which script accomplishes this task and which properties you use, see ZTIAuthorizeDHCP.wsf.

The unique properties and settings for the **Authorize DHCP** task sequence step type are:

Properties

Expand table

Name	Description
Type	Set this read-only type to **Authorize DHCP Server**.

Settings

Expand table

Name	Description
Name	Description
Account	A user account that is a member of the Enterprise Admins group, to be used when authorizing DHCP for the target computer.

Capture Network Settings

This task sequence step gathers the network adapter settings from the target computer. For more information about which script accomplishes this task and which properties you use, see ZTINICConfig.wsf.

The unique properties and settings for the **Capture Network Settings** task sequence step type are:

Properties

Expand table

Name	Description
Name	Description
Type	Set this read-only type to **Capture Network Settings**.

Settings

Expand table

Name	Description
None	None

Configure ADDS

This task sequence step configures the target computer as an Active Directory® Domain Services (AD DS) domain controller. For more information about the settings listed in the following tables and which this task sequence step can configure, see the Microsoft Help and Support article, How to use unattended mode to install and remove Active Directory Domain Services on Windows Server 2008-based domain controllers.

The unique properties and settings for the **Configure ADDS** task sequence step type are:

Properties

Expand table

Name	Description
Type	Set this read-only type to **Configure ADDS**.

Settings

Expand table

Name	Description
Create	Specifies the configuration set that will be used to configure the target computer. The configuration sets are: - **New domain controller replica**. Creates an additional domain controller in an existing AD DS domain - **New read-only domain controller (RODC) replica**. Creates an RODC - **New domain in existing forest**. Creates a domain in an existing AD DS forest

Name	Description
	- **New domain tree in existing forest**. Creates a new tree in an existing AD DS forest - **New forest**. Creates a new AD DS forest
Domain DNS name	The DNS name of the new or existing domain.
Domain NetBIOS name	The NetBIOS name of the new child domain, child domain tree, or forest that pre-AD DS clients use to access the domain. This name must be unique on the network.
DNS name	The DNS name of the child domain or domain tree.
Replication source domain controller	The name of the domain controller from which to source AD DS on new replica or backup domain controller upgrade installations. If no value is supplied, the closest domain controller from the domain being replicated will be selected by default.
Account	The account to be used to perform the configuration.
Recovery (safe mode) password	The password for the offline Administrator account that is used in AD DS Repair mode.
Install DNS if not already present	When selected, DNS will be installed if it has not already been installed.
Make this domain controller a global catalog (GC) server	Specifies whether the replica will also be a GC server. When selected, the target computer will be configured as a GC server if the replication source domain controller is a GC server.
Wait for critical replication only	When selected, this setting specifies that only critical replication is sourced during the replication phase of Dcpromo. Noncritical replication resumes when the computer restarts as a domain controller.
Forest functional level	Specifies the functional level for a new forest. Available options are: - Windows Server 2003 - Windows Server 2008 - Windows Server 2008 R2
Domain functional level	Specifies the functional level for a new domain. Available options are: - Windows Server 2003 - Windows Server 2008

Name	Description
	- Windows Server 2008 R2
Database	Fully qualified, non-Universal Naming Convention (UNC) directory on a hard disk of the local computer that will host the AD DS database (NTDS.dit). If the directory exists, it must be empty. If it does not exist, it will be created. Free disk space on the logical drive selected must be 200 megabytes (MB) and possibly larger when rounding errors are encountered and to accommodate all objects in the domain. For best performance, the directory should be located on a dedicated hard disk.
Log files	Fully qualified, non-UNC directory on a hard disk on the local computer to host the AD DS log files. If the directory exists, it must be empty. If it does not exist, it will be created.
SYSVOL	Fully qualified, non-UNC directory on a hard disk of the local computer that will host the AD DS System Volume (SYSVOL) files. If the directory exists, it must be empty. If it does not exist, it will be created. The directory must be located on a partition that is formatted with the NTFS version 5.0 file system. For best performance, the directory should be located on a different physical hard disk than the operating system.
Site name	The value of an existing AD DS site on which to locate the new domain controller. If not specified, an appropriate site will be selected. This option only applies to the new tree in a new forest scenario. For all other scenarios, a site will be selected using the current site and subnet configuration of the forest.

Configure DHCP

This task sequence step configures the DHCP server service on the target computer. For more information about which script accomplishes this task and which properties you use, see ZTIConfigureDHCP.wsf.

The unique properties and settings for the **Configure DHCP** task sequence step type are:

Properties

Expand table

Name	Description
Type	Set this read-only type to **Configure DHCP Server**.

Settings

Expand table

Name	Description
Name	Configure DHCP
Scope Details	These options apply to any client computers that obtain a lease within that particular scope. Configured scope option values always apply to all computers obtaining a lease in a given scope unless they are overridden by options assigned to class or client reservation. Within the **Scope Details** setting, the following sub-settings are configurable: - **Scope Name**. A user-definable name - **Start IP address**. The starting IP address for the scope - **End IP address**. The ending IP address for the scope - **Subnet mask**. The subnet mask of the client subnet - **Lease duration for DHCP clients**. The duration that the DHCP lease is valid for the client - **Description**. A description of the scope - **Exclude IP address range, Start IP address**. The starting IP address for the range of IP addresses that are to be excluded from the scope - **Exclude IP address range, End IP address**. The ending IP address for the range of IP addresses that are to be excluded from the scope - **003 Router**. A list of IP addresses for routers on the client subnet - **006 DNS Servers**. A list of IP addresses for DNS name servers available to the client - **015 DNS Domain Name**. The domain name that the DHCP client should use when resolving unqualified domain names with DNS - **044 WINS/NBNS Servers**. Lists the IP addresses for NetBIOS name servers (NBNSes) on the network - **046 WINS/NBT Node Type**. Configures the client node type for NetBT clients - **060 PXE Client**. The address used for Pre-Boot Execution Environment (PXE) client bootstrap code

Name	Description
Server Options	These options apply globally for all scopes and classes defined at each DHCP server and for any clients that a DHCP server services. Configured server option values always apply unless they are overridden by options assigned to other scope, class, or client reservation. Within the **Server Options** setting, the following sub-settings are configurable: - **003 Router**. A list of IP addresses for routers on the client subnet - **006 DNS Servers**. A list of IP addresses for DNS name servers available to the client - **015 DNS Domain Name**. The domain name that the DHCP client should use when resolving unqualified domain names with the DNS - **044 WINS/NBNS Servers**. Lists the IP addresses for NBNSes on the network - **046 WINS/NBT Node Type**. Configures the client node type for NetBT clients - **060 PXE Client**. The address used for PXE client bootstrap code

Configure DNS

This task sequence step configures DNS on the target computer. For more information about which script accomplishes this task and which properties you use, see ZTIConfigureDNS.wsf.

The unique properties and settings for the **Configure DNS** task sequence step type are:

Properties

Expand table

Name	Description
Type	Set this read-only type to **Configure DNS Server**.

Settings

Expand table

Name	Description
Name	Configure DNS
Zones	Within the **Scope Details** setting, the following sub-settings are configurable: - **DNS zone name**. A user-definable name - Type. The type of DNS zone to be created - **Replication**. Specifies the replication scheme used to share information among DNS servers - **Zone file name**. The zone's DNS database file - **Dynamic updates**. Enables DNS client computers to register and dynamically update their resource records with a DNS server whenever changes occur - **Scavenge stale resource records**. Removes stale resource records
Server Properties	Within the Server Properties setting, the following sub-settings are configurable: - **Disable recursion**. Specifies that the DNS server will not perform recursion on any query - **BIND secondaries**. Specifies whether to use fast transfer format to transfer a zone to DNS servers running legacy Berkeley Internet Name Domain (BIND) implementations - **Fail on load if bad data**. Specifies the DNS server should parse files strictly - **Enable round robin**. Specifies the DNS server should use the round robin mechanism to rotate and reorder a list of resource records if multiple resource records exist of the same type exist for a query answer - **Enable netmask ordering**. Specifies whether the DNS server should reorder resource records within the same resource record set in its response to a query based on the IP address of the source of the query - **Secure cache against pollution**. Specifies whether the DNS server will attempt to clean up responses to avoid cache pollution - **Name checking**. Configures the name-checking method to be used

> **Note**
>
> The **Configure DNS** task sequence step uses the Dnscmd tool, which is included in Windows Support Tools, to configure DNS. Be sure that Windows Support Tools is

installed before running the **Configure DNS** task sequence step.

> **Note**
>
> For more information about these server properties, see Dnscmd.

Enable BitLocker

This task sequence step configures BitLocker® Drive Encryption on the target computer. For more information about this step type, see Enable BitLocker.

The unique properties and settings for the **Enable BitLocker** task sequence step type are:

Properties

Expand table

Name	Description
Type	Set this read-only type to **Enable BitLocker**.

Settings

Expand table

Name	Description
Current operating system drive	When selected, the operating system drive will be configured. This is the default selection.
Specific drive	When selected, the specified drive will be configured.
TPM only	When selected, the Trusted Platform Module (TPM) is required. This is the default selection.
Startup key on USB only	When selected, a startup key is required on the specified USB drive.
TPM and startup key on USB	When selected, the TPM is required in addition to a startup key on the specified USB drive.
In Active Directory	When selected, the recovery key is stored in AD DS. This is the default selection.

Name	Description
Do not create a recovery key	When selected, the recovery key is not created. Using this option is not recommended.
Wait for BitLocker to complete	When selected, this step will not finish until after BitLocker has finished processing all drives.

Execute Runbook

This task sequence step runs Microsoft System Center 2012 Orchestrator runbooks on the target computer. An Orchestrator *runbook* is the sequence of activities that orchestrate actions on computers and networks. You can initiate Orchestrator runbooks in MDT using this task sequence step type.

> **Note**
>
> This task sequence step is not included any MDT task sequence templates. You must add this task sequence step to any task sequences you create.

The unique properties and settings for the **Execute Runbook** task sequence step type are:

Properties

Expand table

Name	Description
Type	Set this read-only type to **Execute Runbook**.
Name	The name of the task sequence step, which should reflect the name of the runbook being run.
Description	Informative text that provides additional information about the task sequence step

Settings

Expand table

Name	Description

Orchestrator Server	Type the URL for the Orchestrator web service, which includes the server name. The Orchestrator web service can use either Hypertext Transfer Protocol (HTTP) or HTTP over Secure Sockets Layer (HTTPS). The Orchestrator web service defaults to port 81.
	
	The Orchestrator web service supports multiple runbook servers. By default, a runbook can run on any runbook server. A runbook can be configured to specify which runbook servers should be used to run the runbook.
	
	Note:
	
	The Orchestrator web service supports the ability to run a runbook on a specific runbook server. This feature is not supported in MDT.
	
	Specify the URL in any of the following formats:
	
	- ***servername***. When using this format, the URL defaults to:
	
	`https://<servername>:81/Orchestrator2012/Orchestrator.svc`
	
	- ***servername:port***. When using this format, the URL defaults to:
	
	`https://<servername:port>/Orchestrator2012/Orchestrator.svc.`
	
	- *https://servername:port. When using this format, the URL defaults to:*
	
	`https://<servername:port>/Orchestrator2012/Orchestrator.svc.`
	
	- ***https://servername:port**. When using this format, the URL defaults to:*
	
	`https://<servername:port>/Orchestrator2012/Orchestrator.svc.`
	
	- *https://servername:port/Orchestrator2012/Orchestrator.svc*. When using this format, MDT assumes that you are providing the fully qualified URL, because the value ends with .svc.
	
	- **https://*servername:port*/Orchestrator2012/Orchestrator.svc**. When using this format, MDT assumes that you are providing the fully qualified URL, because the value ends with .svc.

Runbook	Select **Browse**, and then select the name of the Orchestrator runbook that this task sequence should run. Note: To successfully browse for Orchestrator runbooks, install the ADO.NET Data

Name	Description
	Services Update for .NET Framework 3.5 SP1 for Windows 7 and Windows Server 2008 R2.
Automatically provide runbook parameters	Select this option to automatically provide the Orchestrator runbook input parameter values(which assumes that the runbook parameter values are task sequence variables). For example, if a runbook has an input parameter named **OSDComputerName**, then the **OSDComputerName** task sequence variable value is passed to the runbook. Note: This option works only for input parameters that are valid task sequence variable names and do not contain spaces or other special characters. Although spaces and other special characters are supported as Orchestrator parameter names, they are not valid task sequence variable names. If you need to pass values to parameters with spaces or other special characters, use the **Specify explicit runbook parameters** option. The other option is **Specify explicit runbook parameters**. Note: The values provided for the runbook input parameters to the Orchestrator web service are formatted as XML. Passing values that contain data that is or resembles XML-formatted data may cause errors.
Specify explicit runbook parameters	Select this option to explicitly provide the Orchestrator runbook input parameters. You must configure the following settings for each input parameter that the Orchestrator runbook requires: - **Name**. This is the name of the input runbook parameter. Note: If you change the parameters for an existing Orchestrator runbook, you need to browse (reselect) for the runbook again, because MDT only retrieves the parameter list when initially adding the Orchestrator runbook. - **Value**. This can be a constant or a variable, such as a task sequence variable or an environment variable. For example, you can specify a value of **%OSDComputerName%**, which will pass the value of the **OSDComputerName** task sequence variable to the runbook input parameter.
Wait for the runbook to	This check box controls whether the task sequence step will wait for the runbook to finish before proceeding to the next task sequence step.

Name	Description
finish before continuing	If this check box is: - **Selected**, then the task sequence step will wait for the runbook to finish before proceeding on to the next task sequence step. When this check box is selected, the task sequence step will poll the Orchestrator web service for the runbook to finish. The amount of time between polls starts at 1 second, then increases to 2, 4, 8, 16, 32, and 64 seconds between each poll. Once the amount of time reaches 64 seconds, the task sequence step continues to poll every 64 seconds. - **Cleared**, then the task sequence step will not wait for the runbook to finish before proceeding to the next task sequence step. Note: This check box must be selected if the runbook returns output parameters.

Format and Partition Disk

This task sequence step partitions and formats disks on the target computer. For more information about this step type, see Format and Partition Disk.

The unique properties and settings for the **Format and Partition Disk** task sequence step type are:

Properties

Expand table

Name	Description
Type	Set this read-only type to **Format and Partition Disk**.

Settings

Expand table

Name	Description
Disk number	The physical number of the disk to be configured.

Name	Description
Disk type	The type of drive to be created. Values are: - **Standard (MBR)** (Master Boot Record) - **GPT** (GUID [globally unique identifier] Partition Table). The default selection is **Standard (MBR)**.
Volume	Within the **Volume** setting, the following sub-settings are configurable: - Partition **Name**. A user-definable name. - **Partition Type.** Values vary by disk type: - MBR: **Primary** only - GPT: **Primary**, **EFI**, or **MSR** - **Use a percentage of remaining space.** - **Use specific drive size.** Values are in increments of 1 MB or 1 gigabyte (GB). - **Make this a boot partition.** - **File System.** Values are **NTFS** or **FAT32**. - **Quick Format.** When selected, a quick format is performed. - **Variable.** The drive letter that was assigned to this newly configured partition.

> **Note**
>
> When using the CustomSettings.ini file to specify the hard disk and partition configurations, only the first hard disk and first two partitions will be configured. Edit ZTIGather.xml to configure additional hard disks or partitions.

Gather

This task sequence step gathers data and processing rules for the target computer. The unique properties and settings for the **Gather** task sequence step type are:

Properties

Expand table

Name	Description
Type	Set this read-only type to **Gather**.

Settings

Expand table

Name	Description
Gather only local data	When selected, this step processes only the properties contained in the ZTIGather.xml file.
Gather local data and process rules	When selected, this step processes the properties contained in the ZTIGather.xml file and the properties contained in the file that the Rules file specifies. This is the default selection.
Rules file	The name of the Rules file to process. If left blank, the task sequence step attempts to locate and process the CustomSettings.ini file.

> **Note**
>
> This task sequence step is natively available in System Center 2012 R2 Configuration Manager as **Set Dynamic Variables**in the General group.

Inject Drivers

This task sequence step injects drivers that have been configured for deployment to the target computer. The unique properties and settings for the **Inject Drivers** task sequence step type are:

Properties

Expand table

Name	Description
Type	Set this read-only type to **Inject Drivers**.

Settings

Expand table

Name	Description
Install only matching drivers	Injects only the drivers that the target computer requires and that match what is available in Out-of-Box Drivers
Install all drivers	Installs all drivers
Selection profile	Installs all drivers in the selected profile

Install Application

This task sequence step installs applications on the target computer. For more information about this step type, see Install Software.

The unique properties and settings for the **Install Application** task sequence step type are:

Properties

Expand table

Name	Description
Type	Set this read-only type to **Install Application**.

Settings

Expand table

Name	Description
Install multiple applications	Install mandatory applications that the **MandatoryApplications** property has specified and optional applications that the **Applications** property has specified. These properties are configured by rules or are specified during the Deployment Wizard interview process. This is the default selection.
Install a single application	The specific application to install. You select the application from a drop-down list that consists of applications that have been configured in the Applications node of the Deployment Workbench.
Success codes	A space-delimited list of application installation exit codes that should be used when determining the successful installation of applications.

Install Operating System

This task sequence step installs an operating system on the target computer. MDT can deploy Windows 8.1, Windows 8, Windows 7, Windows Server 2012 R2, Windows Server 2012, and Windows Server 2008 R2 using:

- **setup.exe**. This method is the traditional method used, initiated by running setup.exe from the installation media. MDT uses setup.exe by default.
- **imagex.exe**. This method installs the operating system image using imagex.exe with the **/apply** option. MDT uses this method when the setup.exe method cannot be used (i.e., it falls back to using imagex.exe).

  You can control which of these methods is used by using the **ForceApplyFallback** property, which also affects which operating system task sequences are listed in the Deployment Wizard for a specific processor architecture boot image. For more information, see the ForceApplyFallback property.

  The unique properties and settings for the **Install Operating System** task sequence step type are:

Properties

Expand table

Name	Description
Type	Set this read-only type to **Install Operating System**.

Settings

Expand table

Name	Description
Operating system to install	The name of the operating system to be installed on the target computer. You select the operating system from a drop-down list compiled from operating systems that have been configured in the Operating Systems node of the Deployment Workbench.
Disk	The disk on which to install the operating system.
Partition	The partition on which to install the operating system.

Install Roles and Features

This task sequence step installs the selected roles and features on the target computer. For more information about which script accomplishes this task and the properties used, see ZTIOSRole.wsf.

The unique properties and settings for the **Install Roles and Features** task sequence step type are:

Properties

Expand table

Name	Description
Type	Set this read-only type to **Install Roles and Features**.
Description	Informative text that describes the purpose of the task sequence step.

Settings

Expand table

Name	Description
Select the operating system for which the roles are to be installed	Select the operating system to be deployed to the target computer.
Select the roles and features that should be installed	Select one or more roles and features for installation on the target computer.

Install Language Packs Offline

This task sequence step installs updates to the image on the target computer after the operating system has been deployed but before the target computer has been restarted. These updates include language packs. For more information about which script accomplishes this task and which properties you use, see ZTIPatches.wsf.

The unique properties and settings for the **Install Language Packs Offline** task sequence step type are:

Properties

Expand table

Name	Description
Type	Set this read-only type to **Install Updates Offline**.

Settings

Expand table

Name	Description
Package Name	The name of the language pack package that should be applied to the target computer

> **Note**
>
> This task sequence step is valid only when using MDT with Configuration Manager.

Install Language Packs Online

This task sequence step installs language packs to the image on the target computer after the operating system has been deployed and after the target computer has been restarted. For more information about which script accomplishes this task and which properties you use, see ZTILangPacksOnline.wsf.

The unique properties and settings for the **Install Language Packs Online** task sequence step type are:

Properties

Expand table

Name	Description
Type	Set this read-only type to **Install Language Packs Online**.

Settings

Expand table

Name	Description
Package Name	The name of the language pack package that should be applied to the target computer

> **Note**
>
> This task sequence step is valid only when using MDT with Configuration Manager.

Install Updates Offline

This task sequence step installs updates to the image on the target computer after the operating system has been deployed but before the target computer has been restarted. These updates include language packs. For more information about which script accomplishes this task and which properties you use, see ZTIPatches.wsf.

The unique properties and settings for the **Install Updates Offline** task sequence step type are:

Properties

Expand table

Name	Description
Type	Set this read-only type to **Install Updates Offline**.

Settings

Expand table

Name	Description
Selection Profile	The name of the selection profile that should be applied to the target computer Note: When using MDT with Configuration Manager, specify the name of the update package that should be applied.

Recover from Domain Join Failure

This task sequence step verifies that the target computer has joined a domain. The unique properties and settings for the **Recover from Domain Join Failure** task sequence step type are:

Properties

Expand table

Name	Description
Type	Set this read-only type to Recover from **Domain Join Failure**.

Settings

Expand table

Name	Description
Auto recover	The task sequence step attempts to join the target computer to a domain.
Manual recover	If the target computer fails to join a domain, the task sequence step causes the Task Sequencer to pause, allowing you to attempt to join the target computer to a domain.
No recover	If the target computer is not able to join a domain, the task sequence fails, stopping the task sequence.

Restart computer

This task sequence step restarts the target computer. The unique properties and settings for the **Restart computer** task sequence step type are:

Properties

Expand table

Name	Description
Type	Set this read-only type to **Restart computer**.

Settings

Expand table

Name	Description
None	None

Run Command Line

This task sequence step runs the specified commands on the target computer. For more information about this step type, see Run Command Line.

The unique properties and settings for the **Run Command Line** task sequence step type are:

Properties

Expand table

Name	Description
Type	Set this read-only type to **Run Command Line**.

Settings

Expand table

Name	Description
Command Line	The commands to be run when this task sequence step is processed
Start in	The starting folder for the application (The path must be a valid path on the target computer.)
Run this step as the following account	Allows specification of user credentials that will be used to run the specified command
Account	The user credentials that will be used to run the specified command
Load the user's profile	When selected, loads the user profile for the specified account

Run PowerShell Script

This task sequence step runs the specified Windows PowerShell™ script on the target computer. For more information about what script accomplishes this task and which properties are used, see ZTIPowerShell.wsf.

The unique properties and settings for the **Run PowerShell Script** task sequence step type are:

Properties

Expand table

Name	Description
Type	Set this read-only type to **Run PowerShell Script**.

Settings

Expand table

Name	Description
PowerShell script	The Windows PowerShell script to be run when this task sequence step is processed

Parameters	The parameters to be passed to the Windows PowerShell script. These parameters should be specified the same as if you were adding them to the Windows PowerShell script from a command line.
	
	The parameters provided should be only those parameters the script consumes, not for the Windows PowerShell command line.
	
	The following example would be a valid value for this setting:
	
	`-MyParameter1 MyValue1 -MyParameter2 MyValue2`
	
	The following example would be an invalid value for this setting (bold items are incorrect):
	
	`-nologo -executionpolicy unrestricted -File MyScript.ps1 -MyParameter1 MyValue1 -MyParameter2 MyValue2`
	
	The previous example is invalid, because the value includes Windows PowerShell command-line parameters (**-nologo** and **-executionpolicy unrestricted**).

> ⓘ Note

This task sequence step is natively available in System Center 2012 R2 Configuration Manager as **Run PowerShell Script** in the General group.

Set Task Sequence Variable

This task sequence step sets the specified task sequence variable to the specified value. For more information about this step type, see Set Task Sequence Variable.

The unique properties and settings for the **Set Task Sequence Variable** task sequence step type are:

Properties

Expand table

Name	Description
Type	Set this read-only type to **Set Task Sequence Variable**.

Settings

Expand table

Name	Description
Task Sequence Variable	The name of the variable to modify
Value	The value to assign to the specified variable

Uninstall Roles and Features

This task sequence step uninstalls the selected roles and features from the target computer. For more information about which script accomplishes this task and the properties used, see ZTIOSRole.wsf.

The unique properties and settings for the **Uninstall Roles and Features** task sequence step type are:

Properties

Expand table

Name	Description
Type	Set this read-only type to **Uninstall Roles and Features**.
Description	Informative text that describes the purpose of the task sequence step.

Settings

Expand table

Name	Description
Select the operating system for which the roles are to be installed	Select the operating system to be deployed to the target computer.
Select the roles and features that should be installed	Select one or more roles and features for unstallation from the target computer.

Validate

This task sequence step verifies that the target computer meets the specified deployment prerequisite conditions. The unique properties and settings for the **Validate** task sequence step type are:

Properties

Expand table

Name	Description
Type	Set this read-only type to **Validate**.

Settings

Expand table

Name	Description
Ensure minimum memory	When selected, this step verifies that the amount of memory, in megabytes, installed on the target computer meets or exceeds the amount specified. This is a default selection.

Name	Description
Ensure minimum processor speed	When selected, this step verifies that the speed of the processor, in megahertz (MHz), installed in the target computer meets or exceeds the amount specified. This is a default selection.
Ensure specified image size will fit	When selected, this step verifies that the amount of free disk space, in megabytes, on the target computer meets or exceeds the amount specified.
Ensure current operating system to be refreshed	When selected, this step verifies that the operating system installed on the target computer meets the requirement specified. This is a default selection.

> ⓘ Note
>
> This task sequence step is natively available in System Center 2012 R2 Configuration Manager as **Check Readiness** in the General group.

Out-of-Box Task Sequence Steps

The following task sequence steps are referenced by one or more of the available task sequence templates included with MDT. Each of the following examples lists the preconfigured properties, parameters, and options and can be used as a basis for building custom task sequences.

Only the task sequence step properties, parameters, and options, and their corresponding values are listed in the examples.

> ⓘ Note
>
> For more information about each task sequence step, see the corresponding topics in Common Properties and Options for Task Sequence Step Types and Specific Properties and Settings for Task Sequence Step Types.

Apply Network Settings

This task sequence step configures the network adapter on the target computer. Following is a brief listing of the settings that show how this step was originally configured in one of the MDT task sequence templates. For more information about which script accomplishes this task and which properties are used, see ZTINICConfig.wsf.

The default configuration of the **Apply Network Settings** task sequence step is:

Properties

Expand table

Name	Value
Type	Apply Network Settings
Name	Apply Network Settings
Description	Not specified

Settings

Expand table

Name	Value
	No parameters are preconfigured for this step. This causes this step, by default, to configure the network adapter to use DHCP.

Options

Expand table

Name	Value
Disable this step	Not selected
Success codes	0 3010
Continue on error	Not selected
Conditional qualifier	Not specified

> **Note**
>
> When using the CustomSettings.ini file to specify the network adapter configurations, only the first network adapter will be configured. Edit ZTIGather.xml to configure additional network adapters.

Apply Patches

This task sequence step installs updates to the image on the target computer after the operating system has been deployed but before the target computer has been restarted. Following is a brief listing of the settings that show how this step was originally configured in one of the MDT task sequence templates. For more information about which script accomplishes this task and which properties you use, see ZTIPatches.wsf.

The default configuration of the **Install Updates Offline** task sequence step is:

Properties

Expand table

Name	Value
Type	Install Updates Offline
Name	Apply Patches
Description	Not specified

Settings

Expand table

Name	Value
Selection profile	The name of the profile used when selecting the patches to install on the target computer

Options

Expand table

Name	Value
Disable this step	Not selected
Success codes	0 3010
Continue on error	Not selected

Name	Value
Conditional qualifier	Not specified

Apply Windows PE

This task sequence step prepares the target computer to start in Windows Preinstallation Environment (Windows PE). Following is a brief listing of the settings that show how this step was originally configured in one of the MDT task sequence templates. For more information about which script accomplishes this task and which properties you use, see LTIApply.wsf.

The default configuration of the **Apply Windows PE** task sequence step is:

Properties

Expand table

Name	Value
Type	Run Command Line
Name	Apply Windows PE
Description	Not specified

Settings

Expand table

Name	Value
Command line	`cscript.exe "%SCRIPTROOT%\LTIApply.wsf" /PE`
Start in	Not specified
Run this step as the following account	Not specified

Options

Expand table

Name	Value
Disable this step	Not selected
Success codes	0 3010
Continue on error	Not selected
Conditional qualifier	Not specified

Backup

This task sequence step backs up the target computer before starting the operating system deployment. Following is a brief listing of the settings that show how this step was originally configured in one of the MDT task sequence templates. For more information about which script accomplishes this task and which properties you use, see ZTIBackup.wsf.

The default configuration of the **Backup** task sequence step is:

Properties

Expand table

Name	Value
Type	Run Command Line
Name	Backup
Description	Not specified

Settings

Expand table

Name	Value
Command line	`cscript.exe "%SCRIPTROOT%\ZTIBackup.wsf"`
Start in	Not specified
Run this step as the following account	Not specified

Options

Expand table

Name	Value
Disable this step	Not selected
Success codes	0 3010
Continue on error	Not selected
Conditional qualifier	Not specified

Capture Groups

This task sequence step captures group membership of local groups that exist on the target computer. Following is a brief listing of the settings that show how this step was originally configured in one of the MDT task sequence templates. For more information about which script accomplishes this task and which properties you use, see ZTIGroups.wsf.

The default configuration of the **Capture Groups** task sequence step is:

Properties

Expand table

Name	Value
Type	Run Command Line
Name	Capture Groups
Description	Not specified

Settings

Expand table

Name	Value
Command line	`cscript.exe "%SCRIPTROOT%\ZTIGroups.wsf" /capture`
Start in	Not specified.

Name	Value
Run this step as the following account	Not specified

Options

Expand table

Name	Value
Disable this step	Not selected
Success codes	0 3010
Continue on error	Not selected
Conditional qualifier	Not specified

Capture User State

This task sequence step captures the user state for user profiles that exist on the target computer. Following is a brief listing of the settings that show how this step was originally configured in one of the MDT task sequence templates. For more information about what script accomplishes this task and what properties are used, see [ZTIUserState.wsf]((scripts.md#ztiuserstatewsf). For more information about this step type, see Capture User State.

The default configuration of the **Capture User State** task sequence step is:

Properties

Expand table

Name	Value
Type	Run Command Line
Name	Capture User State
Description	Not specified

Settings

Expand table

Name	Value
Command line	`cscript.exe "%SCRIPTROOT%\ZTIUserState.wsf" /capture`
Start in	Not specified
Run this step as the following account	Not specified

Options

Expand table

Name	Value
Disable this step	Not selected
Success codes	0 3010
Continue on error	Not selected
Conditional qualifier	Not specified

Check BIOS

This task sequence step checks the basic input/output system (BIOS) of the target computer to ensure that it is compatible with the operating system you are deploying. Following is a brief listing of the settings that show how this step was originally configured in one of the MDT task sequence templates. For more information about which script accomplishes this task and which properties are used, see [ZTIBIOSCheck.wsf]((scripts.md#ztibioscheckwsf).

The default configuration of the **Check BIOS** task sequence step is:

Properties

Expand table

Name	Value
Type	Run Command Line
Name	Check BIOS

Name	Value
Description	Not specified

Settings

Expand table

Name	Value
Command line	`cscript.exe "%SCRIPTROOT%\ZTIBIOSCheck.wsf"`
Start in	Not specified
Run this step as the following account	Not specified

Options

Expand table

Name	Value
Disable this step	Not selected
Success codes	0 3010
Continue on error	Not selected
Conditional qualifier	Not specified

Configure

This task sequence step configures the Unattend.xml file with the required property values that are applicable to the operating system you are deploying to the target computer. Following is a brief listing of the settings that show how this step was originally configured in one of the MDT task sequence templates. For more information about which script accomplishes this task and which properties you use, see [ZTIConfigure.wsf]((scripts.md#zticonfigurewsf).

The default configuration of the **Configure** task sequence step is:

Properties

Expand table

Name	Value
Type	Run Command Line
Name	Configure
Description	Not specified

Settings

Expand table

Name	Value
Command line	`cscript.exe "%SCRIPTROOT%\ZTIConfigure.wsf"`
Start in	Not specified
Run this step as the following account	Not specified

Options

Expand table

Name	Value
Disable this step	Not selected
Success codes	0 3010
Continue on error	Not selected
Conditional qualifier	Not specified

Copy Scripts

This task sequence step copies the deployment scripts used during the deployment processes to a local hard disk on the target computer. Following is a brief listing of the settings that show how this step was originally configured in one of the MDT task sequence templates. For more information about which script accomplishes this task and which properties you use, see LTICopyScripts.wsf.

The default configuration of the **Copy Scripts** task sequence step is:

Properties

Expand table

Name	Value
Type	Run Command Line
Name	Copy Scripts
Description	Not specified

Settings

Expand table

Name	Value
Command line	`cscript.exe "%SCRIPTROOT%\LTICopyScripts.wsf"`
Start in	Not specified
Run this step as the following account	Not specified

Options

Expand table

Name	Value
Disable this step	Not selected
Success codes	0 3010
Continue on error	Not selected
Conditional qualifier	Not specified

Copy Sysprep Files

This task sequence step copies the Sysprep files to the target computer. Following is a brief listing of the settings that show how this step was originally configured in one of the MDT task sequence templates. For more information about which script accomplishes this task and which properties you use, see LTISysprep.wsf.

The default configuration of the **Copy Sysprep Files** task sequence step is:

Properties

Expand table

Name	Value
Type	Run Command Line
Name	Copy Sysprep Files
Description	Not specified

Settings

Expand table

Name	Value
Command line	`cscript.exe "%SCRIPTROOT%\LTISysprep.wsf"`
Start in	Not specified
Run this step as the following account	Not specified

Options

Expand table

Name	Value
Disable this step	Not selected
Success codes	0 3010
Continue on error	Not selected
Conditional qualifier	Not specified

Create BitLocker Partition

This task sequence step sets the **BDEInstall** property to True, indicating that BitLocker should be installed on the target computer. The unique properties and settings for the

Create BitLocker Partition task sequence step type are:

Properties

Expand table

Name	Value
Type	Set Task Sequence Variable
Name	Create BitLocker Partition
Description	None

Settings

Expand table

Name	Value
Task Sequence Variable	BDE Install
Value	True

Options

Expand table

Name	Value
Disable this step	Not selected
Success codes	0 3010
Continue on error	Not selected
Conditional qualifier	Not specified

Create WIM

This task sequence step creates a backup of the target computer. The unique properties and settings for the **Create WIM** task sequence step type are:

Properties

Expand table

Name	Value
Type	Run Command Line
Name	Create WIM
Description	None

Settings

Expand table

Name	Value
Command line	`cscript.exe "%SCRIPTROOT%\ZTIBackup.wsf"`
Start in	Not specified
Run this step as the following account	Not specified

Options

Expand table

Name	Value
Disable this step	Not selected
Success codes	0 3010
Continue on error	Not selected
Conditional qualifier	Not specified

Disable BDE Protectors

If BitLocker is installed on the target computer, this task sequence step disables the BitLocker protectors.

The unique properties and settings for the **Disable BDE Protectors** task sequence step type are:

Properties

Expand table

Name	Value
Type	Run Command Line
Name	Disable BDE Protectors
Description	None

Settings

Expand table

Name	Value
Command line	`cscript.exe "%SCRIPTROOT%\ZTIDisableBDEProtectors.wsf"`
Start in	Not specified
Run this step as the following account	Not specified

Options

Expand table

Name	Value
Disable this step	Not selected
Success codes	0 3010
Continue on error	Not selected
Conditional qualifier	Not specified

Enable BitLocker

This task sequence step enables BitLocker on the target computer. Following is a brief listing of the settings that show how this step was originally configured in one of the

MDT task sequence templates. For more information about which script accomplishes this task and what properties are used, see ZTIBde.wsf.

The default configuration of the **Enable BitLocker** task sequence step is:

Properties

Expand table

Name	Value
Type	Enable BitLocker
Name	Enable BitLocker
Description	None

Settings

Expand table

Name	Value
Current operating system drive	Selected
TPM only	Selected
Startup key on USB only	Not selected
TPM and startup key on USB	Not selected
Specific drive	Not selected
In Active Directory	Selected
Do not create a recovery key	Not selected
Wait for BitLocker to complete	Not selected

Options

Expand table

Name	Value
Disable this step	Not selected

Name	Value
Success codes	0 3010
Continue on error	Not selected
Conditional qualifier	**BdeInstallSuppress** does not equal YES

Enable OEM Disk Configuration

This task sequence step sets the **DeploymentType**property to **NEWCOMPUTER**, which allows the target computer's disk to be partitioned and formatted.

The unique properties and settings for the **Enable OEM Disk Configuration** task sequence step type are:

Properties

Expand table

Name	Value
Type	Set Task Sequence Variable
Name	Enable OEM Disk Configuration
Description	None

Settings

Expand table

Name	Value
Task Sequence Variable	DeploymentType
Value	NEWCOMPUTER

Options

Expand table

Name	Value
Disable this step	Not selected
Success codes	0 3010
Continue on error	Not selected
Conditional qualifier	Not specified

End Phase

This task sequence step ends the current deployment phase and restarts the target computer. Following is a brief listing of the settings that show how this step was originally configured in one of the MDT task sequence templates.

The default configuration of the **End Phase** task sequence step is:

Properties

Expand table

Name	Value
Type	Restart computer
Name	End Phase
Description	Not specified

Settings

Expand table

Name	Value
None	None

Options

Expand table

Name	Value
Disable this step	Not selected
Success codes	0 3010
Continue on error	Not selected
Conditional qualifier	Not specified

Execute Sysprep

This task sequence step starts Sysprep on the target computer. Following is a brief listing of the settings that show how this step was originally configured in one of the MDT task sequence templates. For more information about what script accomplishes this task and what properties are used, see LTISysprep.wsf.

The default configuration of the **Execute Sysprep** task sequence step is:

Properties

Expand table

Name	Value
Type	Run Command Line
Name	Execute Sysprep
Description	None

Settings

Expand table

Name	Value
Command line	`cscript.exe "%SCRIPTROOT%\LTISysprep.wsf"`
Start in	Not specified
Run this step as the following account	Not specified

Options

Expand table

Name	Value
Disable this step	Not selected
Success codes	0 3010
Continue on error	Not selected
Conditional qualifier	Not specified

Force Diskpart Action

If the C:\oem.wsf file exists, this task sequence step deletes the C:\oem.wsf file, which will allow the **Format and Partition Disk** task sequence step to run. Following is a brief listing of the settings that show how this step was originally configured in one of the MDT task sequence templates.

The default configuration of the **Force Diskpart Action** task sequence step is:

Properties

Expand table

Name	Value
Type	Run Command Line
Name	Force Diskpart Action
Description	Not specified

Settings

Expand table

Name	Value
Command line	`cmd.exe /c if exist c:\oem.wsf del /q c:\oem.wsf`
Start in	Not specified
Run this step as the following account	Not specified

Options

Expand table

Name	Value
Disable this step	Not selected
Success codes	0 1
Continue on error	Selected
Conditional qualifier	None

Format and Partition Disk

This task sequence step configures and formats disk partitions on the target computer. Following is a brief listing of the settings that show how this step was originally configured in one of the MDT task sequence templates.

For more information about what script accomplishes this task and what properties are used, see ZTIDiskpart.wsf.

The default configuration of the **Format and Partition Disk** task sequence step is:

Properties

Expand table

Name	Value
Type	Format and Partition Disk
Name	Format and Partition Disk
Description	Not specified

Settings

Expand table

Name	Value
Disk number	0

Name	Value
Disk type	Standard (MBR)
Volume	Within the Volume setting, the following sub-settings are configured: - **Partition Name**. OSDisk - **Partition Type**. Primary - **Use a percentage of remaining space**. Selected - **Size(%)**. 100 - **Use specific drive size**. Not selected - **Make this a boot partition**. Selected - **File System**. NTFS - **Quick Format**. Selected - **Variable**. Not specified

Options

Expand table

Name	Value
Disable this step	Not selected
Success codes	0 3010
Continue on error	Not selected
Conditional qualifier	Not specified

> **Note**
>
> When using the CustomSettings.ini file to specify the hard disk and partition configurations, only the first hard disk and first two partitions will be configured. Edit ZTIGather.xml to configure additional hard disks or partitions.

Gather local only

This task sequence step gathers deployment configurations settings from local sources that apply to the target computer. Following is a brief listing of the settings that show how this step was originally configured in one of the MDT task sequence templates.

For more information about what script accomplishes this task and what properties are used, see ZTIGather.wsf.

The default configuration of the **Gather local only** task sequence step is:

Properties

Expand table

Name	Value
Type	Gather
Name	Gather local only
Description	Not specified

Settings

Expand table

Name	Value
Gather only local data	Selected
Gather local data and process rules	Not selected
Rules file	Not specified

Options

Expand table

Name	Value
Disable this step	Not selected
Success codes	0 3010
Continue on error	Not selected

Name	Value
Conditional qualifier	None

Generate Application Migration File

This task sequence step generates the ZTIAppXmlGen.xml file, which contains a list of file associations that are installed on the target computer. Following is a brief listing of the settings that show how this step was originally configured in one of the MDT task sequence templates.

For more information about what script accomplishes this task and what properties are used, see ZTIAppXmlGen.wsf.

The default configuration of the **Generate Application Migration** File task sequence step is:

Properties

Expand table

Name	Value
Type	Run Command Line
Name	Generate Application Migration File
Description	Not specified

Settings

Expand table

Name	Value
Command Line	`cscript.exe "%SCRIPTROOT%\ZTIAppXmlGen.wsf" /capture`
Start in	Not specified
Run this step as the following account	Not specified

Options

Expand table

Name	Value
Disable this step	Not selected
Success codes	0 3010
Continue on error	Not selected
Conditional qualifier	None

Inject Drivers

This task sequence step injects drivers that have been configured for deployment to the target computer. Following is a brief listing of the settings that show how this step was originally configured in one of the MDT task sequence templates.

For more information about what script accomplishes this task and what properties are used, see ZTIDrivers.wsf.

The default configuration of the **Inject Drivers** task sequence step is:

Properties

Expand table

Name	Value
Type	Inject Drivers
Name	Inject Drivers
Description	Not specified

Settings

Expand table

Name	Value
Install only matching drivers	Injects only the drivers which are required by the target computer and match with what is available in Out-of-Box Drivers
Install all drivers	Injects all drivers

Name	Value
Selection profile	Injects drivers which are associated with the selected profile

Options

Expand table

Name	Value
Disable this step	Not selected
Success codes	0 3010
Continue on error	Not selected
Conditional qualifier	Not specified

Install Applications

This task sequence step installs applications on the target computer. Following is a brief listing of the settings that show how this step was originally configured in one of the MDT task sequence templates.

For more information about what script accomplishes this task and what properties are used, see ZTIApplications.wsf.

The default configuration of the **Install Applications** task sequence step is:

Properties

Expand table

Name	Value
Type	Install Applications
Name	Install Applications
Description	Not specified

Settings

Expand table

Name	Value
Install multiple applications	Selected
Install a single application	Not selected

Options

Expand table

Name	Value
Disable this step	Not selected
Success codes	0 3010
Continue on error	Not selected
Conditional qualifier	Not specified

Install Operating System

This task sequence step installs an operating system on the target computer. Following is a brief listing of the settings that show how this step was originally configured in one of the MDT task sequence templates.

The default configuration of the **Install Operating System** task sequence step is:

Properties

Expand table

Name	Value
Type	Install Operating System
Name	Install Operating System
Description	Not specified

Settings

Expand table

Name	Value
Operating system to install	This value corresponds to the operating system that was selected when the task sequence was created.
Disk	The disk where the operating system is to be installed.
Partition	The partition where the operating system is to be installed.

Options

Expand table

Name	Value
Disable this step	Not selected
Success codes	0 3010
Continue on error	Not selected
Conditional qualifier	Not specified

Next Phase

This task sequence step updates the **Phase** property to the next phase in the deployment process. Following is a brief listing of the settings that show how this step was originally configured in one of the MDT task sequence templates.

For more information about what script accomplishes this task and what properties are used, see ZTINextPhase.wsf.

The default configuration of the **Next Phase** task sequence step is:

Properties

Expand table

Name	Value
Type	Run Command Line
Name	Next Phase

Name	Value
Description	Not specified

Settings

Expand table

Name	Value
Command line	`cscript.exe "%SCRIPTROOT%\ZTINextPhase.wsf"`
Start in	Not specified
Run this step as the following account	Not specified

Options

Expand table

Name	Value
Disable this step	Not selected
Success codes	0 3010
Continue on error	Not selected
Conditional qualifier	Not specified

Post-Apply Cleanup

This task sequence step cleans up unnecessary files after the installation of an image on the target computer. Following is a brief listing of the settings that show how this step was originally configured in one of the MDT task sequence templates.

For more information about what script accomplishes this task and what properties are used, see LTIApply.wsf.

The default configuration of the **Post-Apply Cleanup** task sequence step is:

Properties

Expand table

Name	Value
Type	Run Command Line
Name	Post-Apply Cleanup
Description	Not specified

Settings

Expand table

Name	Value
Command line	`cscript.exe "%SCRIPTROOT%\LTIApply.wsf" /post`
Start in	Not specified
Run this step as the following account	Not specified

Options

Expand table

Name	Value
Disable this step	Not selected
Success codes	0 3010
Continue on error	Not selected
Conditional qualifier	Not specified

Recover from Domain

This task sequence step will verify the target computer has joined a domain. For more information about which script accomplishes this task and which properties are used, see ZTIDomainJoin.wsf.

The unique properties and settings for the Recover from Domain task sequence step type are:

Properties

Expand table

Name	Description
Type	This read-only type is set to Recover from Domain Join Failure.

Settings

Expand table

Name	Description
Auto recover	The task sequence step will attempt to join the target computer to a domain.
Manual recover	If the target computer fails to join a domain, the task sequence step will cause the task sequencer to pause, allowing the user attempts to join the target computer to a domain.
No recover	If the target computer is not able to join a domain, the task sequence fails, stopping the task sequence.

Restart computer

This task sequence step restarts the target computer. Following is a brief listing of the settings that show how this step was originally configured in one of the MDT task sequence templates.

The default configuration of the **Restart computer** task sequence step is:

Properties

Expand table

Name	Value
Type	Restart computer
Name	Restart computer
Description	Not specified

Settings

Expand table

Name	Value
None	None

Options

Expand table

Name	Value
Disable this step	Not selected
Success codes	0 3010
Continue on error	Not selected
Conditional qualifier	Not specified

Restore Groups

This task sequence step restores the previously captured group membership of local groups on the target computer. Following is a brief listing of the settings that show how this step was originally configured in one of the MDT task sequence templates.

For more information about what script accomplishes this task and what properties are used, see ZTIGroups.wsf.

The default configuration of the **Restore Groups** task sequence step is:

Properties

Expand table

Name	Value
Type	Run Command Line
Name	Restore Groups
Description	Not specified

Settings

Expand table

Name	Value
Command line	`cscript.exe "%SCRIPTROOT%\ZTIGroups.wsf" /restore`
Start in	Not specified
Run this step as the following account	Not specified

Options

Expand table

Name	Value
Disable this step	Not selected
Success codes	0 3010
Continue on error	Not selected
Conditional qualifier	If all conditions are true: - **DoCapture** does not equal YES - **DoCapture** does not equal PREPARE

Restore User State

This task sequence step restores previously captured user state to the target computer. Following is a brief listing of the settings that show how this step was originally configured in one of the MDT task sequence templates.

For more information about what script accomplishes this task and what properties are used, see [ZTIUserState.wsf]((scripts.md#ztiuserstatewsf).

For more information about this step type, see Restore User State.

The default configuration of the **Restore User State** task sequence step is:

Properties

Expand table

Name	Value
Type	Run Command Line
Name	Restore User State
Description	Not specified

Settings

Expand table

Name	Value
Command Line	`cscript.exe "%SCRIPTROOT%\ZTIUserState.wsf" /restore`
Start in	Not specified
Run this step as the following account	Not specified

Options

Expand table

Name	Value
Disable this step	Not selected
Success codes	0 3010
Continue on error	Not selected
Conditional qualifier	If all conditions are true: - If **DoCapture** does not equal YES - If **DoCapture** does not equal PREPARE

Set Image Build

This task sequence step sets the **ImageBuild** property to the value contained in **OSCurrentVersion**. Following is a brief listing of the settings that show how this step was originally configured in one of the MDT task sequence templates.

The default configuration of the **Set Image Build** task sequence step is:

Properties

Expand table

Name	Value
Type	Set Task Sequence Variable
Name	Set Image Build
Description	Not specified

Settings

Expand table

Name	Value
Task Sequence Variable	ImageBuild
Value	%OSCurrentVersion%

Options

Expand table

Name	Value
Disable this step	Not selected
Success codes	0 3010
Continue on error	Not selected
Conditional qualifier	Not specified

Set Image Flags

This task sequence step sets the **ImageFlags** property to the value contained in **OSSKU**. Following is a brief listing of the settings that show how this step was originally configured in one of the MDT task sequence templates.

The default configuration of the **Set Image Flags** task sequence step is:

Properties

Expand table

Name	Value
Type	Set Task Sequence Variable
Name	Set Image Flags
Description	Not specified

Settings

Expand table

Name	Value
Task Sequence Variable	ImageFlags
Value	%OSSKU%

Options

Expand table

Name	Value
Disable this step	Not selected
Success codes	0 3010
Continue on error	Not selected
Conditional qualifier	Not specified

Tattoo

This task sequence step tattoos the target computer with identification and version information. Following is a brief listing of the settings that show how this step was originally configured in one of the MDT task sequence templates.

For more information about what script accomplishes this task and what properties are used, see ZTITatoo.wsf.

The default configuration of the **Tattoo** task sequence step is:

Properties

Expand table

Name	Value
Type	Run Command Line
Name	Tattoo
Description	Not specified

Settings

Expand table

Name	Value
Command line	`cscript.exe "%SCRIPTROOT%\ZTITatoo.wsf"`
Start in	Not specified
Run this step as the following account	Not specified

Options

Expand table

Name	Value
Disable this step	Not selected
Success codes	0 3010
Continue on error	Not selected
Conditional qualifier	Not specified

Validate

This task sequence step validates that the target computer meets the specified deployment prerequisite conditions. Following is a brief listing of the settings that show how this step was originally configured in one of the MDT task sequence templates.

For more information about what script accomplishes this task and what properties are used, see ZTIValidate.wsf.

The default configuration of the **Validate** task sequence step is:

Properties

Expand table

Name	Value
Type	Validate
Name	Validate
Description	Not specified

Settings

Expand table

Name	Value
Ensure minimum memory (MB)	Selected. The value selector is set to **768**.
Ensure minimum processor speed (MHz)	Selected. The value selector is set to **800**.
Ensure specified image size will fit (MB)	Not selected.
Ensure current operating system to be refreshed	Selected. The value selector is set to **Server** or **Client**, depending on the template used to create the task sequence.

Options

Expand table

Name	Value
Disable this step	Not selected
Success codes	0 3010
Continue on error	Not selected
Conditional qualifier	Not specified

Windows Update (Pre-Application Installation)

This task sequence step installs updates to the target computer prior to the installation of applications. Following is a brief listing of the settings that show how this step was originally configured in one of the MDT task sequence templates.

For more information about what script accomplishes this task and what properties are used, see ZTIWindowsUpdate.wsf.

The default configuration of the **Windows Update (Pre-Application Installation)** task sequence step is:

Properties

Expand table

Name	Value
Type	Run Command Line
Name	Windows Update (Pre-Application Installation)
Description	Not specified

Settings

Expand table

Name	Value
Command line	`cscript.exe "%SCRIPTROOT%\ZTIWindowsUpdate.wsf"`
Start in	Not specified
Run this step as the following account	Not specified

Options

Expand table

Name	Value
Disable this step	Not selected
Success codes	0 3010
Continue on error	Not selected
Conditional qualifier	Not specified

Windows Update (Post-Application Installation)

This task sequence step is the same as the **Windows Update (Pre-Application Installation)** task sequence step.

Wipe Disk

This task sequence step wipes all information from the disk using the **Format** command.

For more information about what script accomplishes this task and what properties are used, see ZTIWipeDisk.wsf.

The default configuration of the **Wipe Disk** task sequence step is:

Properties

Expand table

Name	Value
Type	Run Command Line
Name	Wipe Disk
Description	This will only run if **WipeDisk**=TRUE in CustomSettings.ini

Settings

Expand table

Name	Value
Command line	`cscript.exe "%SCRIPTROOT%\ZTIWipeDisk.wsf"`
Start in	Not specified
Run this step as the following account	Not specified

Options

Expand table

Name	Value
Disable this step	Not selected
Success codes	0 3010
Continue on error	Not selected
Conditional qualifier	Not specified

Related articles

- Properties.
- Scripts.
- Support Files.
- Utilities.
- MDT Windows PowerShell Cmdlets.
- Tables and Views in the MDT DB.
- Windows 7 Feature Dependency Reference.
- UDI Reference.

Feedback

Was this page helpful? Yes No

Provide product feedback

Properties

Article • 02/12/2024

The scripts used in Lite Touch Installation (LTI) and ZTI reference properties to determine the process steps and configuration settings used during the deployment process. The scripts create some of these properties automatically. Other properties must be configured in the CustomSettings.ini file. Some of these properties are:

- Specific to ZTI only
- Specific to LTI only
- For use in both ZTI and LTI

  Use this reference to help determine the correct properties to configure and the valid values to include for each property.

  For each property the following information is provided:

- **Description**.Provides a description of the purpose of the property and any pertinent information regarding the customization of the property.

  > **Note**
  >
  > Unless explicitly specified for ZTI or LTI only, a property is valid for both ZTI and LTI.

- **Value and Description**.Indicates the valid values to be specified for the property and a brief description of what each value means. (Values in italics indicate that a value is substituted—for example the value *user1*, *user2* indicates that *user1* and *user2* would be replaced with the actual name of user accounts.)

- **Example**.Provides an example of a property use as it might appear in the .ini files.

  For more information about these and other task sequence properties that might be referenced while performing a ZTI deployment, see Operating System Deployment Task Sequence Variables.

  The deployment scripts generally require values to be specified in upper case so that they are properly read. Therefore, when specifying property values, use uppercase letters.

Property Definition

The following sections describe the properties that are available for LTI and ZTI deployments in MDT.

> **Tip**
>
> The properties are sorted in alphabetical order.

_SMSTSOrgName

Customizes the Task Sequencer engine's display banner

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	✅	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
name	The name that will be used in the Task Sequencer engine's display banner

Expand table

Example
`[Settings] Priority=Default [Default] _SMSTSOrgName=Woodgrove Bank`

ADDSLogPath

Fully qualified, non-UNC directory on a hard disk on the local computer to host the AD DS log files. If the directory exists it must be empty. If it does not exist, it will be created.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
log_path	Fully qualified, non-UNC directory on a hard disk on the local computer to host the AD DS log files

Expand table

Example
`[Settings] Priority=Default [Default] ADDSLogPath=%DestinationLogicalDrive%\Windows\NTDS`

ADDSPassword

Account credentials that can be used when promoting the server to a domain controller.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
password	Account credentials that can be used for the promotion operation

Expand table

Example
`[Settings] Priority=Default [Default] ADDSUserName=Administrator ADDSUserDomain=WoodGroveBank ADDSPassword=<complex_password>`

ADDSUserDomain

This is the domain the account specified by **ADDSUserName** should be taken from. If the operation is to create a new forest or to become a member server from a backup domain controller upgrade there is no default. If the operation is to create a new tree, the default is the DNS name of the forest the computer is currently joined to. If the operation is to create a new child domain or a replica then the default is the DNS name of the domain the computer is joined to. If the operation is to demote the computer and the computer is a domain controller in a child domain, the default is the DNS name of the parent domains. If the operation is to demote the computer, and the computer is a domain controller of a tree root domain, the default is the DNS name of the forest.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
domain	Domain the UserName account should be taken from

Expand table

Example
`[Settings] Priority=Default [Default] ADDSUserName=Administrator ADDSUserDomain=WoodGroveBank ADDSPassword=<complex_password>`

ADDSUserName

Account credentials that will be used when promoting the server to a domain controller.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
user_name	Account credentials that will be used for the promotion operation

Expand table

Example
`[Settings] Priority=Default [Default] ADDSUserName=Administrator ADDSUserDomain=WoodGroveBank ADDSPassword=complex_password`

Administrators

A list of user accounts and domain groups that will be added to the local Administrator group on the target computer. The **Administrators** property is a list of text values that can be any non-blank value. The **Administrators** property has a numeric suffix (for example, **Administrators001** or **Administrators002**).

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
name	The name of a user or group that is to be added to the local Administrator group

Expand table

Example
`[Settings] Priority=Default [Default] Administrators001=WOODGROVEBANK\NYC Help Desk Staff Administrators002=WOODGROVEBANK\North America East Help Desk Staff PowerUsers001=WOODGROVEBANK\User01 PowerUsers002=WOODGROVEBANK\User02`

AdminPassword

Defines the password that will be assigned to the local Administrator user account on the target computer. If not specified, the pre-deployment password of the Administrator user account will be used.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
admin_password	The password that is to be assigned to the Administrator user account on the target computer

Expand table

Example
`[Settings] Priority=Default [Default] Administrators001=WOODGROVEBANK\NYC Help Desk Staff AdminPassword=<admin_password>`

Applications

A list of application GUIDs that should be installed on the target computer. These applications are specified on the Applications node in Deployment Workbench. These GUIDs are stored in the Applications.xml file. The **Applications** property is a list of text

values that can be any non-blank value. The **Applications** property has a numeric suffix (for example, **Applications001** or **Applications002**).

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
application_guid	The GUID is specified by Deployment Workbench for the application to be deployed to the target computer. The GUID corresponds to the application GUID stored in the Applications.xml file.

Expand table

Example
`[Settings] Priority=Default [Default] Applications001={1D7DF331-47B7-472C-87B3-442597EC2F7D} Applications002={9d2b8999-5e4d-4f3d-bb05-edaaf4fe5628}`

ApplicationSuccessCodes

A space-delimited list of error codes used by the ZTIApplications script that determine the successful installation of applications.

> ⓘ Note
>
> This property is only applicable to the **Install Application** task sequence step type and when **Install multiple applications** is selected.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅

Component	Configured By		Scenario	Property Is Applicable
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
error_codes	The error codes that determine when applications have been successfully installed. Default values are **0** and **3010**.

Expand table

Example
`[Settings] Priority=Default [Default] ApplicationSuccessCodes=0 3010`

ApplyGPOPack

This property is used to determine whether the **Apply Local GPO Package** task sequence step is performed.

> ⓘ Note
>
> The default value for this property always performs the **Apply Local GPO Package** task sequence step. You must explicitly provide a value of "NO" to override this behavior..

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
YES	The **Apply Local GPO Package** task sequence step is performed. This is the default value.
NO	The **Apply Local GPO Package** task sequence step is not performed.

Expand table

Example
`[Settings] Priority=Default [Default] ApplyGPOPack=NO`

Architecture

The processor architecture of the processor that is currently running, which is not necessarily the processor architecture supported by the target computer. For example, when running a 32-bit-compatible operating system on a 64-bit processor, **Architecture** will indicate that the processor architecture is 32 bit.

Use the **CapableArchitecture** property to identify the actual processor architecture that the target computer supports.

> **Note**
>
> This property is dynamically set by MDT scripts and is not configured in CustomSettings.ini. Treat this property as read only. However, you can use this property within CustomSettings.ini, as shown in the following examples, to aid in defining the configuration of the target computer.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	❌	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
x86	Processor architecture is 32 bit.
x64	Processor architecture is 64 bit.

Expand table

Example
None

AreaCode

The area code to be configured for the operating system on the target computer. This property allows only numeric characters. This value is inserted into the appropriate configuration settings in Unattend.xml.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
area_code	The area code where the target computer is to be deployed

Expand table

Example
`[Settings] Priority=Default [Default] AreaCode=206 CountryCode=001 Dialing=TONE LongDistanceAccess=9`

AssetTag

The asset tag number associated with the target computer. The format for asset tag numbers is undefined. Use this property to create a subsection that contains settings targeted to a specific computer.

> **Note**
>
> This property is dynamically set by MDT scripts and cannot have its value set in CustomSettings.ini or the MDT DB. Treat this property as read only. However, you can use this property within CustomSettings.ini or the MDT DB, as shown in the following examples, to aid in defining the configuration of the target computer.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	❌	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
asset_tag	The format of the asset tag is undefined and is determined by the asset tag standard of each organization.

Expand table

Example 1

```
[Settings] Priority=Default [Default] OSDComputerName=HP-%AssetTag%
```

Expand table

Example 2

```
[Settings] Priority=AssetTag, Default [Default] OSInstall=YES [0034034931]
OSDComputerName=HPD530-1 [0034003233] OSDNEWMACHINENAME=BVMXP
```

AutoConfigDNS

Specifies whether the Active Directory Installation Wizard configures DNS for the new domain if it detects that the DNS dynamic update protocol is not available.

> **⊗ Caution**
>
> This property value must be specified in uppercase so that the deployment scripts can properly read it.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
YES	Configures DNS for the new domain if the DNS dynamic update protocol is not available
NO	Does not configure DNS for the domain

Expand table

Example
`[Settings] Priority=Default [Default] AutoConfigDNS=YES`

BackupDir

The folder in which backups of the target computer are stored. This folder exists beneath the UNC path specified in the **BackupShare** property. If the folder does not already exist, it will be created automatically.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅

Component	Configured By		Scenario	Property Is Applicable
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
Folder	The name of the folder that exists beneath the shared folder specified in the **BackupShare** property

Expand table

Example
`[Settings] Priority=Default [Default] DoCapture=YES BackupShare=\\NYC-AM-FIL-01\Backup$ BackupDir=%OSDComputerName% BackupDrive=C:`

BackupDrive

The drive to include in the backup of the target computer. This property defaults to the drive that contains disk 0 partition 1. It can be also set to **ALL**.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
backup_drive	The drive letter of the drive to back up
ALL	Back up all drives on the target computer

Expand table

Example
`[Settings] Priority=Default [Default] DoCapture=YES BackupShare=\\NYC-AM-FIL-01\Backup$ BackupDir=%OSDComputerName% BackupDrive=C:`

BackupFile

Specifies the WIM file that will be used by the ZTIBackup.wsf script. For more information about what script uses this property, see ZTIBackup.wsf.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
BackupDir	The name of the Windows Imaging Format (WIM) file to be used during back up.

Expand table

Example
`[Settings] Priority=Default [Default] DoCapture=YES BackupShare=\\NYC-AM-FIL-01\Backup$ BackupDir=%OSDComputerName% BackupFile=%OSDComputerName%.wim`

BackupShare

The shared folder in which backups of the target computer are stored.

The credentials used to access this shared folder for:

- LTI are the credentials entered in the Deployment Wizard.
- ZTI are the credentials used by the Configuration Manager Advanced Client Network Access account.

  The permissions required on this share are as follows:

- **Domain Computers**. Allow the Create Folders/Append Data permission.
- **Domain Users**. Allow the Create Folders/Append Data permission.
- **Creator Owner**. Allow the Full Control permission.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
UNC_path	The UNC path of the shared folder Note: The UNC path specified in this property must exist before deploying the target operating system.

Expand table

Example
`[Settings] Priority=Default [Default] DoCapture=YES BackupShare=\\NYC-AM-FIL-01\Backup$ BackupDir=%OSDComputerName% BackupDrive=C:`

BDEAllowAlphaNumericPin

This property configures whether BitLocker PINs contain alphanumeric values.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		

Component	Configured By		Scenario	Property Is Applicable
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
YES	Alphanumeric characters are allowed in the PIN. Note: In addition to setting this property to **YES**, the **Allow enhanced PINs for startup** group policy setting must be enabled.
NO	Only numeric characters are allowed in the PIN.

Expand table

Example
`[Settings] Priority=Default [Default] BDEInstallSuppress=NO BDEAllowAlphaNumericPin=YES BDEDriveLetter=S: BDEDriveSize=2000 BDEInstall=TPMKey BDERecoveryKey=AD BDEKeyLocation=C:`

BDEDriveLetter

The drive letter for the partition that is not encrypted by BitLocker, also known as the *System Volume*. SYSVOL is the directory that contains the hardware-specific files needed to load Windows computers after the BIOS has booted the platform.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
drive_letter	The letter designation for the logical drive for the System Volume (such as S or T).

Value	Description
	The default value is **S**.

Expand table

Example
`[Settings] Priority=Default [Default] BDEInstallSuppress=NO BDEDriveLetter=S: BDEDriveSize=2000 BDEInstall=TPMKey BDERecoveryKey=AD BDEKeyLocation=C:`

BDEDriveSize

The size of the BitLocker system partition. The value is specified in megabytes. In the example, the size of the BitLocker partition to create is almost 2 GB (2,000 MB).

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
drive_size	The size of the partition in megabytes; the default sizes are: - Windows 7 and Windows Server 2008 R2: 300 MB

Expand table

Example
`[Settings] Priority=Default [Default] BDEInstallSuppress=NO BDEDriveLetter=S: BDEDriveSize=2000 BDEInstall=TPMKey BDERecoveryKey=AD BDEKeyLocation=C:`

BDEInstall

The type of BitLocker installation to be performed. Protect the target computer using one of the following methods:

- A TPM microcontroller
- A TPM and an external startup key (using a key that is typically stored on a USB flash drive [UFD])
- A TPM and PIN
- An external startup key

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
TPM	Protect the computer with TPM only. The TPM is a microcontroller that stores keys, passwords, and digital certificates. The microcontroller is typically an integral part of the computer motherboard.
TPMKey	Protect the computer with TPM and a startup key. Use this option to create a startup key and to save it on a UFD. The startup key must be present in the port each time the computer starts.
TPMPin	Protect the computer with TPM and a pin. Use this option in conjunction with the **BDEPin** property.
Key	Protect the computer with an external key (the recovery key) that can be stored in a folder, in AD DS, or printed.

Expand table

Example
`[Settings] Priority=Default [Default] BDEInstallSuppress=NO BDEDriveLetter=S: BDEDriveSize=2000 BDEInstall=TPMKey BDERecoveryKey=AD BDEKeyLocation=C:`

BDEInstallSuppress

Indicates whether the deployment process should skip the BitLocker installation.

> **⊗ Caution**
>
> This property value must be specified in uppercase so that the deployment scripts can read it properly.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
YES	Do not attempt to install BitLocker.
NO	Attempt to install BitLocker.

Expand table

Example
`[Settings] Priority=Default [Default] BDEInstallSuppress=YES`

BDEKeyLocation

The location for storing the BitLocker recovery key and startup key.

> **ⓘ Note**
>
> If this property is configured using the Deployment Wizard, the property must be the drive letter of a removable disk. If the **SkipBitLocker** property is set to **TRUE** so

that the **Specify the BitLocker configuration** wizard page is skipped, this property can be set to a UNC path in CustomSettings.ini or in the MDT database (MDT DB).

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
Location	Specifies where the recovery key will be stored; must be a UNC path or the drive letter of a removable disk. If not set, the first available removable drive will be used.

Expand table

Example
`[Settings] Priority=Default [Default] BDEInstallSuppress=NO BDEDriveLetter=S: BDEDriveSize=2000 BDEInstall=TPMKey BDERecoveryKey=AD BDEKeyLocation=C:`

BDEPin

The PIN to be assigned to the target computer when configuring BitLocker and the **BDEInstall** or **OSDBitLockerMode** properties are set to **TPMPin**.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
Pin	The PIN to be used for BitLocker. The PIN can be between 4 and 20 digits long.

Expand table

Example
`[Settings] Priority=Default [Default] BDEInstallSuppress=NO BDEDriveLetter=S: BDEDriveSize=2000 BDEInstall=TPMPin BDEPin=123456789`

BDERecoveryKey

A Boolean value that indicates whether the process creates a recovery key for BitLocker. The key is used for recovering data encrypted on a BitLocker volume. This key is cryptographically equivalent to a startup key. If available, the recovery key decrypts the volume master key (VMK), which, in turn, decrypts the full volume encryption key (FVEK).

> **Note**
>
> The recovery key is stored in the location specified in the **BDEKeyLocation** property.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
AD	A recovery key is created.
Not specified	A recovery key is not created.

Expand table

Example
`[Settings] Priority=Default [Default] BDEInstallSuppress=NO BDEDriveLetter=S: BDEDriveSize=2000 BDEInstall=TPMKey BDERecoveryKey=AD BDEKeyLocation=C:`

BDEWaitForEncryption

Specifies that the deployment process should not proceed until BitLocker has completed the encryption process for all specified drives. Specifying TRUE could dramatically increase the time required to complete the deployment process.

> **⊗ Caution**
>
> This property value must be specified in uppercase so that the deployment scripts can read it properly.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
TRUE	Specifies that the deployment process should wait for drive encryption to complete.
FALSE	Specifies that the deployment process should not wait for drive encryption to complete.

Expand table

Example
`[Settings] Priority=Default [Default] BDEInstallSuppress=NO BDEDriveLetter=S: BDEDriveSize=2000 OSDBitLockerMode=TPMKey OSDBitLockerStartupKeyDrive=C: OSDBitLockerCreateRecoveryPassword=AD BDEWaitForEncryption=TRUE`

BitsPerPel

A setting for displaying colors on the target computer. The property can contain numeric digits and corresponds to the color quality setting. In the example, **32** indicates 32 bits per pixel for color quality. This value is inserted into the appropriate configuration settings in Unattend.xml.

> **Note**
>
> The default values (in the Unattend.xml template file) are 1,024 pixels horizontal resolution, 768 pixels vertical resolution, 32-bit color depth, and 60 Hertz (Hz) vertical refresh rate.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
bits_per_pixel	The number of bits per pixel to use for color. The default value is the default for the operating system being deployed.

Expand table

Example
`[Settings] Priority=Default [Default] BitsPerPel=32 VRefresh=60 XResolution=1024 YResolution=768`

BuildID

Identifies the operating system task sequence to be deployed to the target computer. You create the task sequence ID on the Task Sequences node in the Deployment Workbench. The **BuildID** property allows alphanumeric characters, hyphens (-), and underscores (_). The **BuildID** property cannot be blank or contain spaces.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
build_id	Identifier of the operating system task sequence as defined in the Deployment Workbench for the target operating system being deployed Note: Make certain to use the **TaskSequenceID** specified in the Deployment Workbench user interface (UI) and not the GUID of the **TaskSequenceID**.

Expand table

Example
`[Settings] Priority=Default [Default] BuildID=BareMetal`

CapableArchitecture

The processor architecture of the processor supported by the target computer, not the current processor architecture that is running. For example, when running a 32-bit-compatible operating system on a 64-bit processor, **CapableArchitecture** will indicate that the processor architecture is 64 bit.

Use the **Architecture** property to see the processor architecture that is currently running.

> ⓘ **Note**
>
> This property is dynamically set by the MDT scripts and is not configured in CustomSettings.ini or the MDT DB. Treat this property as read only.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	❌	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
x86	Processor architecture is 32 bit.
x64	Processor architecture is 64 bit.

Expand table

Example
None

CaptureGroups

Controls whether the group membership of local groups on the target computer is captured. This group membership is captured during the State Capture Phase and is restored during the State Restore Phase.

> **Note**
>
> This property value must be specified in uppercase so that the deployment scripts can read it properly.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
NO	Captures no group membership information.
ALL	Captures the membership of all local groups on the target computer.
YES	Captures the membership of the Administrator and Power Users built-in groups and the groups listed in the groups' properties. This is the default value if some other value is specified. (**YES** is the typical value.)

Expand table

Example
`[Settings] Priority=Default [Default] DeployRoot=\\NYC-AM-FIL-01\Distribution$ ResourceRoot=\\NYC-AM-FIL-01\Resource$ UDShare=\\NYC-AM-FIL-01\MigData$ CaptureGroups=YES Groups1=NYC Application Management Groups2=NYC Help Desk Users`

ChildName

Specifies whether to append the DNS label at the beginning of the name of an existing directory service domain when installing a child domain.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
name	The name of the child domain

Expand table

Example
`[Settings] Priority=Default [Default] ChildName=childdom.parentdom.WoodGroveBank.com`

ComputerBackupLocation

The network shared folder where the computer backup is stored. If the target folder does not already exist, it is automatically created.

> **⊗ Caution**
>
> This property value must be specified in uppercase so that the deployment scripts can read it properly.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
blank	Same as **AUTO**.
UNC_path	The UNC path to the network shared folder where the backup is stored.
AUTO	Creates a backup on a local hard disk if space is available. Otherwise, the backup is saved to a network location specified in the **BackupShare** and **BackupDir** properties.
NETWORK	Creates a backup on a network location specified in **BackupShare** and **BackupDir**.
NONE	No backup will be performed.

Expand table

Example

```
[Settings] Priority=Default [Default] DeployRoot=\\NYC-AM-FIL-01\Distribution$
ResourceRoot=\\NYC-AM-FIL-01\Resource$ UDShare=\\NYC-AM-FIL-01\MigData$
ComputerBackupLocation=NETWORK BackupShare=\\NYC-AM-FIL-01\Backup$
BackupDir=%OSDComputerName% UDDir=%OSDComputerName% SLShare=\\NYC-AM-FIL-01\Logs$
UDProfiles=Administrator, User-01, ExtranetUser UserDataLocation=NONE
```

ComputerName

This property has been deprecated. Use **OSDComputerName** instead.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	❌
CustomSettings.ini	❌	|		
MDT DB	❌	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
None	None

Expand table

Example
None

ConfigFileName

Specifies the name of the configuration file used during OEM deployments.

> **Note**
>
> This property is dynamically set by the MDT scripts and is not configured in CustomSettings.ini or the MDT DB. Treat this property as read only.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	❌	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
file_name	Specifies the name of the configuration file used during OEM deployments

Expand table

Example
None

ConfigFilePackage

Specifies the package ID for the configuration package used during OEM deployments.

> **Note**
>
> This property is dynamically set by the MDT scripts and is not configured in CustomSettings.ini or the MDT DB. Treat this property as read only.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	❌	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
package	Specifies the package ID for the configuration package used during OEM deployments

Expand table

Example
None

ConfirmGC

Specifies whether the replica is also a global catalog.

> ⊗ **Caution**
>
> This property value must be specified in uppercase so that the deployment scripts can read it properly.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
YES	Makes the replica a global catalog if the backup was a global catalog.
NO	Does not make the replica a global catalog.

Expand table

Example
`[Settings] Priority=Default [Default] ConfirmGC=YES`

CountryCode

The country code to be configured for the operating system on the target computer. This property allows only numeric characters. This value is inserted into the appropriate configuration settings in Unattend.xml.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅

Component	Configured By		Scenario	Property Is Applicable
CustomSettings.ini	✅			
MDT DB	✅		ZTI (Configuration Manager)	✅

Expand table

Value	Description
country_code	The country code where the target computer is to be deployed

Expand table

Example
`[Settings] Priority=Default [Default] AreaCode=206 CountryCode=001 Dialing=TONE LongDistanceAccess=9`

CriticalReplicationOnly

Specifies whether the promotion operation performs only critical replication and then continues, skipping the noncritical (and potentially lengthy) portion of replication.

> ⊗ **Caution**
>
> This property value must be specified in uppercase so that the deployment scripts can read it properly.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌		LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅			
MDT DB	✅		ZTI (Configuration Manager)	✅

Expand table

Value	Description
YES	Skips noncritical replication

Value	Description
NO	Does not skip noncritical replication

Expand table

Example
`[Settings] Priority=Default [Default] CriticalReplicationOnly=YES`

CustomDriverSelectionProfile

Specifies the custom selection profile used during driver installation.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
profile	Custom selection profile used during driver installation

Expand table

Example
`[Settings] Priority=Default [Default] CustomDriverSelectionProfile=CustomDrivers`

CustomPackageSelectionProfile

Specifies the custom selection profile used during package installation.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
profile	Custom selection profile used during package installation

Expand table

Example
`[Settings] Priority=Default [Default] CustomPackageSelectionProfile=CustomPackages`

CustomWizardSelectionProfile

Specifies the custom selection profile used by the wizard for filtering the display of various items.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
profile	Custom selection profile by the wizard for filtering the display of various items

Expand table

Example
`[Settings] Priority=Default [Default] CustomWizardSelectionProfile=CustomWizard`

Database

The property that specifies the database to be used for querying property values from columns in the table specified in the **Table** property. The database resides on the computer specified in the **SQLServer** property. The instance of Microsoft SQL Server® on the computer is specified in the **Instance** property.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	✅	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
database	The name of the database to be used for querying property values

Expand table

Example
`[Settings] Priority=Computers, Default [Default] OSInstall=YES [Computers] SQLServer=NYC-SQL-01 SQLShare=SQL$ Database=MDTDB Instance=SQLEnterprise2005 Table=Computers Parameters=SerialNumber, AssetTag ParameterCondition=OR`

DatabasePath

Specifies the fully qualified, non-UNC path to a directory on a fixed disk of the target computer that contains the domain database.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
path	Specifies the fully qualified, non-UNC path to a directory on a fixed disk of the local computer that contains the domain database

Expand table

Example
`[Settings] Priority=Default [Default] DatabasePath=%DestinationLogicalDrive%\Windows\NTSD`

DBID

Specifies the user account used to connect to the computer running SQL Server (specified by the **SQLServer** property) using SQL Server authentication. The **DBPwd** property provides the password for the user account in the **DBID** property.

> ⓘ **Note**
>
> SQL Server authentication is not as secure as Integrated Windows authentication. Integrated Windows authentication is the recommended authentication method. Using the **DBID** and **DBPwd** properties stores the credentials in clear text in the CustomSettings.ini file and therefore is not secure. For more information about using Integrated Windows authentication, see the SQLShare property.

> ⓘ **Note**
>
> This property is configurable only by manually editing the CustomSettings.ini and BootStrap.ini files.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	✅	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
user_id	The name of the user account credentials used to access the computer running SQL Server using SQL Server authentication

Expand table

Example
`[Settings] Priority=Computers, Default [Default] OSInstall=YES [Computers] SQLServer=NYC-SQL-01 DBID=SQL_User-01 DBPwd=<complex_password> NetLib=DBNMPNTW Database=MDTDB Instance=SQLEnterprise2005 Table=Computers Parameters=SerialNumber, AssetTag ParameterCondition=OR`

DBPwd

Specifies the password for the user account specified in the **DBID** property. The **DBID** and **DBPwd** properties provide the credentials for performing SQL Server authentication to the computer running SQL Server (specified by the **SQLServer** property).

> ⓘ Note
>
> SQL Server authentication is not as secure as Integrated Windows authentication. Integrated Windows authentication is the recommended authentication method. Using the **DBID** and **DBPwd** properties stores the credentials in clear text in the CustomSettings.ini file and therefore is not secure. For more information about using Integrated Windows authentication, see the SQLShare property.

> ⓘ Note
>
> This property is configurable only by manually editing the CustomSettings.ini and BootStrap.ini files.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	✅	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
user_password	The password for the user account credentials specified in the **DBID** property for using SQL Server authentication

Expand table

Example
`[Settings] Priority=Computers, Default [Default] OSInstall=YES [Computers] SQLServer=NYC-SQL-01 DBID=SQL_User-01 DBPwd=<complex_password> NetLib=DBNMPNTW Database=MDTDB Instance=SQLEnterprise2005 Table=Computers Parameters=SerialNumber, AssetTag ParameterCondition=OR`

Debug

Controls the verbosity of messages written to the MDT log files. This property can be configured to help assist in troubleshooting deployments by providing extended information about the MDT deployment process.

You can set this property by starting the LiteTouch.vbs script with the **/debug:true** command-line parameter as follows:

Windows Command Prompt

```
cscript.exe LiteTouch.vbs /debug:true
```

After the LiteTouch.vbs script is started, the **Debug** property's value is set to **TRUE**, and all other scripts are automatically read the value of this property and provide verbose information.

> ⓘ **Note**

This property is dynamically set by the MDT scripts and is not configured in CustomSettings.ini or in the MDT DB. Treat this property as read only.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	❌	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
TRUE	Debug logging is enabled, which includes the following: - Verbose messages are logged. - Deprecated messages are logged as errors.
FALSE	Debug logging is not enabled. This is the default value.

Expand table

Example
None

DefaultGateway

The IP address of the default gateway being used by the target computer. The format of the IP address returned by the property is standard dotted-decimal notation; for example, 192.168.1.1. Use this property to create a subsection that contains settings targeted to a group of computers based on the IP subnets on which they are located.

> **Note**
>
> This property is dynamically set by MDT scripts and cannot have its value set in CustomSettings.ini or the MDT DB. Treat this property as read only. However, you

can use this property within CustomSettings.ini or the MDT DB, as shown in the following examples, to aid in defining the configuration of the target computer.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
default_gateway	The IP address of the default gateway in standard dotted-decimal notation

Expand table

Example
`[Settings] Priority=DefaultGateway, Default [Default] OSInstall=YES [DefaultGateway] 192.168.0.1=HOUSTON 11.1.1.11=REDMOND 172.28.20.1=REDMOND [REDMOND] Packages001=XXX00004:Program4 Packages002=XXX00005:Program5 [HOUSTON] Packages001=XXX00006:Program6 Packages002=XXX00007:Program7 Packages003=XXX00008:Program8`

DeployDrive

The value used by the scripts to access files and run programs in the deployment share that the Deployment Workbench creates. The property returns the drive letter mapped to the **DeployRoot** property. ZTIApplications.wsf uses the **DeployDrive** property when running any command-line programs with a .cmd or .bat extension.

> **Note**
>
> This property is dynamically set by the MDT scripts and is not configured in CustomSettings.ini or the MDT DB. Treat this property as read only.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	❌	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
drive_letter	The letter designation for the logical drive where the target operating system is to be installed (such as C or D)

Expand table

Example
None

DeploymentMethod

The method being used for the deployment (UNC, media, or Configuration Manager).

> **Note**
>
> This property is dynamically set by the MDT scripts and is not configured in CustomSettings.ini or the MDT DB. Treat this property as read only.

> **Caution**
>
> This property value must be specified in uppercase so that the deployment scripts can read it properly.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	❌	|		

Component	Configured By		Scenario	Property Is Applicable
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
UNC	The deployment is made to the target computer over the network.
Media	The deployment is made from local media (such as DVD or hard disk) at the target computer.
SCCM	ZTI uses this method for Configuration Manager.

Expand table

Example
None

DeploymentType

The type of deployment being performed based on the deployment scenario. For ZTI, this property is set dynamically by MDT scripts and is not configured in CustomSettings.ini. For LTI, you can bypass the page in the Deployment Wizard on which the deployment type is selected. In addition, you can specify the deployment type by passing one of the values listed below to the LiteTouch.wsf script as a command-line option.

> **⊗ Caution**
>
> This property value must be specified in uppercase so that the deployment scripts can read it properly.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
NEWCOMPUTER	The target computer is a new computer that has never been a member of the network.
REFRESH	The target computer is an existing computer on the network that needs the desktop environment standard to be redeployed.
REPLACE	An existing computer on the network is being replaced with a new computer. The user state migration data is transferred from the existing computer to a new computer.

Expand table

Example
`[Settings] Priority=Default [Default] DeploymentType=NEWCOMPUTER`

DeployRoot

Specifies the UNC or local path to the folder that is the root of the folder structure that MDT uses. This folder structure contains configuration files, scripts, and other folders and files that MDT uses. The value of this property is set based on the following MDT deployment technologies:

- **LTI**. This property is the UNC path to the deployment share that the Deployment Workbench creates. Use this property to select a specific deployment share. The most common use of this property is in the BootStrap.ini file to identify a deployment share before the connection to the deployment share is established. All other deployment share folders are relative to this property (such as device drivers, language packs, or operating systems).
- **ZTI**. This property is the local path to the folder to which the MDT files package is copied. The **Use Toolkit Package** task sequence step copies the MDT files package to a local folder on the target computer, and then automatically sets this property to the local folder.

  > ⓘ Note

For ZTI, this property is dynamically set by the MDT scripts and is not configured in CustomSettings.ini or in the MDT DB. Treat this property as read only.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	✅	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	❌	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
path	The UNC or local path to the .

Expand table

Example
`[Settings] Priority=Default [Default] DeployRoot=\\NYC-AM-FIL-01\Distribution$ UserDataLocation=NONE`

DestinationDisk

Disk number that the image will be deployed to.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	✅	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	❌	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
disk_number	The number of the disk to which the image will be deployed

Expand table

Example
`[Settings] Priority=Default [Default] DestinationDisk=0`

DestinationLogicalDrive

The logical drive to which the image will be deployed.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	✅	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	❌	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
logical_drive_number	The logical drive to which the image will be deployed

Expand table

Example 1
`[Settings] Priority=Default [Default] DestinationLogicalDrive=0`

Expand table

Example 2

`[Settings] Priority=Default [Default] DestinationLogicalDrive=0`
 	
`[Settings] Priority=Default [Default] InstallDNS=YES DomainNetBIOSName=WoodGroveBank NewDomain=Child DomainLevel=3 ForestLevel=3 NewDomainDNSName=newdom.WoodGroveBank.com ParentDomainDNSName=WoodGroveBank.com AutoConfigDNS=YES ConfirmGC=YES`

Example 2
`CriticalReplicationOnly=NO ADDSUserName=Administrator ADDSUserDomain=WoodGroveBank ADDSPassword=<complex_password> DatabasePath=%DestinationLogicalDrive%\Windows\NTDS ADDSLogPath=%DestinationLogicalDrive%\Windows\NTDS SysVolPath=%DestinationLogicalDrive%\Windows\SYSVOL SafeModeAdminPassword=<complex_password>`

DestinationPartition

Disk partition to which the image will be deployed.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	✅	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	❌	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
partition_number	The number of the partition to which the image will be deployed

Expand table

Example
`[Settings] Priority=Default [Default] DestinationPartition=1`

DHCPScopes

Specifies the number of DHCP scopes to configure.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		

Component	Configured By		Scenario	Property Is Applicable
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
scopes	Specifies the number of DHCP scopes to configure

Expand table

Example
`[Settings] Priority=Default [Default] DHCPScopes=1`

DHCPScopesxDescription

The description of the DHCP scope.

> **Note**
>
> The *x* in this properties name is a placeholder for a zero-based array that contains DHCP configurations.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
description	The description of the DHCP scope

Expand table

Example
`[Settings] Priority=Default [Default] DHCPScopes0Description=DHCPScope0`

DHCPScopesxEndIP

Specifies the ending IP address for the DHCP scope.

The *x* in this properties name is a placeholder for a zero-based array that contains DHCP configurations.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
end_IP	Specifies the ending IP address for the DHCP scope

Expand table

Example
`[Settings] Priority=Default [Default] DHCPScopes0EndIP=192.168.0.30`

DHCPScopesxExcludeEndIP

Specifies the ending IP address for the DHCP scope exclusion. IP addresses that are excluded from the scope are not offered by the DHCP server to clients obtaining leases from this scope.

> ⓘ **Note**
>
> The *x* in this properties name is a placeholder for a zero-based array that contains DHCP configurations.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
exclude_end_IP	Specifies the ending IP address for the DHCP scope exclusion

Expand table

Example
`[Settings] Priority=Default [Default] DHCPScopes0ExcludeEndIP=192.168.0.15`

DHCPScopesxExcludeStartIP

Specifies the starting IP address for the DHCP scope exclusion. IP addresses that are excluded from the scope are not offered by the DHCP server to clients obtaining leases from this scope.

> **Note**
>
> The *x* in this properties name is a placeholder for a zero-based array that contains DHCP configurations.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
exclude_start_IP	Specifies the starting IP address for the DHCP scope exclusion

Expand table

Example
`[Settings] Priority=Default [Default] DHCPScopes0ExcludeStartIP=192.168.0.10`

DHCPScopesxIP

Specifies the IP subnet of the scope.

The *x* in this properties name is a placeholder for a zero-based array that contains DHCP configurations.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
IP	Specifies the IP subnet of the scope

Expand table

Example
`[Settings] Priority=Default [Default] DHCPScopes0IP=192.168.0.0`

DHCPScopesxName

A user-definable name to be assigned to the scope.

The *x* in this properties name is a placeholder for a zero-based array that contains DHCP configurations.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
name	A user-definable name to be assigned to the scope

Expand table

Example
`[Settings] Priority=Default [Default] DHCPScopes0Name=DHCPScope0`

DHCPScopesxOptionDNSDomainName

Specifies the domain name that the DHCP client should use when resolving unqualified domain names with the DNS.

> **Note**
>
> The *x* in this properties name is a placeholder for a zero-based array that contains DHCP configurations.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
DNS_domain_name	Specifies the domain name that the DHCP client should use when resolving unqualified domain names with the DNS

Expand table

Example
`[Settings] Priority=Default [Default] DHCPScopes0OptionDNSDomainName=WoodGroveBank.com`

DHCPScopesxOptionDNSServer

Specifies a list of IP addresses for DNS name servers available to the client. When more than one server is assigned, the client interprets and uses the addresses in the specified order.

> **Note**
>
> The *x* in this properties name is a placeholder for a zero-based array that contains DHCP configurations.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
DNS_server	Specifies a list of IP addresses for DNS name servers available to the client

Expand table

Example
`[Settings] Priority=Default [Default] DHCPScopes0OptionDNSServer=192.168.0.2`

DHCPScopesxOptionLease

The duration that the DHCP lease is valid for the client.

> **Note**
>
> The *x* in this properties name is a placeholder for a zero-based array that contains DHCP configurations.

Expand table

Component	Configured By	Scenario	Property Is Applicable
BootStrap.ini	❌	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅		
MDT DB	✅	ZTI (Configuration Manager)	✅

Expand table

Value	Description
lease	The duration that the DHCP lease is valid for the client

Expand table

Example
`[Settings] Priority=Default [Default] DHCPScopes0OptionLease=7`

DHCPScopesxOptionNBTNodeType

Specifies the client node type for NetBT clients.

> **Note**

> The *x* in this properties name is a placeholder for a zero-based array that contains DHCP configurations.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
1	Configures the node type as b-node
2	Configures the node type as p-node
4	Configures the node type as m-node
8	Configures the node type as h-node

Expand table

Example
`[Settings] Priority=Default [Default] DHCPScopes0OptionNBTNodeType=4`

DHCPScopesxOptionPXEClient

Specifies the IP address used for PXE client bootstrap code.

> ⓘ Note
>
> The *x* in this properties name is a placeholder for a zero-based array that contains DHCP configurations.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
PXE_client	Specifies the IP address used for PXE client bootstrap code

Expand table

Example
`[Settings] Priority=Default [Default] DHCPScopes0OptionPXEClient=192.168.0.252`

DHCPScopesxOptionRouter

Specifies a list of IP addresses for routers on the client subnet. When more than one router is assigned, the client interprets and uses the addresses in the specified order. This option is normally used to assign a default gateway to DHCP clients on a subnet.

> ⓘ **Note**
>
> The *x* in this properties name is a placeholder for a zero-based array that contains DHCP configurations.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
router	Specifies a list of IP addresses for routers on the client subnet

Expand table

Example
`[Settings] Priority=Default [Default] DHCPScopes0OptionRouter=192.168.0.253`

DHCPScopesxOptionWINSServer

Specifies the IP addresses to be used for NBNSes on the network.

> **Note**
>
> The *x* in this properties name is a placeholder for a zero-based array that contains DHCP configurations

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
WINS_server	Specifies the IP addresses to be used for NBNSes on the network

Expand table

Example
`[Settings] Priority=Default [Default] DHCPScopes0OptionWINSServer=192.168.0.2`

DHCPScopesxStartIP

The starting IP address for the range of IP addresses that are to be included in the scope.

> **Note**
>
> The *x* in this properties name is a placeholder for a zero-based array that contains DHCP configurations.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
start_IP	The starting IP address for the range of IP addresses that are to be excluded from the scope

Expand table

Example
`[Settings] Priority=Default [Default] DHCPScopes0StartIP=192.168.0.20`

DHCPScopesxSubnetMask

Specifies the subnet mask of the client subnet.

> **Note**
>
> The *x* in this properties name is a placeholder for a zero-based array that contains DHCP configurations.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
subnet_mask	Specifies the subnet mask of the client IP subnet

Expand table

Example
`[Settings] Priority=Default [Default] DHCPScopes0SubnetMask=255.255.255.0`

DHCPServerOptionDNSDomainName

Specifies the connection-specific DNS domain suffix of client computers.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
DNS_domain_name	Specifies the connection-specific DNS domain suffix of client computers

Expand table

Example
`[Settings] Priority=Default [Default] DHCPServerOptionDNSDomainName=Fabrikam.com`

DHCPServerOptionDNSServer

Specifies a list of IP addresses to be used as DNS name servers that are available to the client.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
DNS_server	Specifies a list of IP addresses to be used as DNS name servers that are available to the client

Expand table

Example
`[Settings] Priority=Default [Default] DHCPServerOptionDNSServer=192.168.0.1,192.168.0.2`

DHCPServerOptionNBTNodeType

Specifies the client node type for NetBT clients.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
1	Configures the node type as b-node
2	Configures the node type as p-node
4	Configures the node type as m-node
8	Configures the node type as h-node

Expand table

Example
`[Settings] Priority=Default [Default] DHCPServerOptionNBTNodeType=4`

DHCPServerOptionPXEClient

Specifies the IP address used for PXE client bootstrap code.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
PXE_client	Specifies the IP address used for PXE client bootstrap code

Expand table

Example
`[Settings] Priority=Default [Default] DHCPServerOptionPXEClient=192.168.0.252`

DHCPServerOptionRouter

Specifies a list of IP addresses for routers on the client subnet. When more than one router is assigned, the client interprets and uses the addresses in the specified order. This option is normally used to assign a default gateway to DHCP clients on a subnet.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
router	Specifies a list of IP addresses for routers on the client subnet

Expand table

Example
`[Settings] Priority=Default [Default] DHCPServerOptionRouter=192.168.0.253`

DHCPServerOptionWINSServer

Specifies the IP addresses to be used for NBNSes on the network.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
WINS_server	Specifies the IP addresses to be used for NBNSes on the network

Expand table

Example
`[Settings] Priority=Default [Default] DHCPServerOptionWINSServer=192.168.0.2`

Dialing

The type of dialing supported by the telephony infrastructure where the target computer is located. This value is inserted into the appropriate configuration settings in Unattend.xml.

> **⊗ Caution**
>
> This property value must be specified in uppercase so that the deployment scripts can read it properly.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	✗	|	LTI (Stand-alone MDT)	✓
CustomSettings.ini	✓	|		
MDT DB	✓	|	ZTI (Configuration Manager)	✓

Expand table

Value	Description
PULSE	The telephony infrastructure supports pulse dialing.
TONE	The telephony infrastructure supports touch-tone dialing.

Expand table

Example
`[Settings] Priority=Default [Default] AreaCode=206 CountryCode=001 Dialing=TONE LongDistanceAccess=9`

DisableTaskMgr

This property controls a user's ability to start Task Manager by pressing CTRL+ALT+DEL. After the user starts Task Manager, they could interrupt the LTI task sequence while running in the new operating system on the target computer. This property is used in conjunction with the **HideShell** property and is only valid when the **HideShell** property is set to **YES**.

> ⓘ **Note**
>
> This property and the **HideShell** property must both be set to **YES** to prevent the user pressing CTRL+ALT+DEL and interrupting the LTI task sequence.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
YES	Prevent the user from being able to start Task Manager by pressing CTRL+ALT+DEL and subsequently interrupting the LTI task sequence.
NO	Allow the user to start Task Manager by pressing CTRL+ALT+DEL and subsequently interrupt the LTI task sequence. This is the default value.

Expand table

Example
`[Settings] Priority=Default [Default] DisableTaskMgr=YES HideShell=YES`

DNSServerOptionBINDSecondaries

Determines whether to use fast transfer format for transfer of a zone to DNS servers running legacy BIND implementations.

By default, all Windows-based DNS servers use a fast zone transfer format. This format uses compression, and it can include multiple records per TCP message during a

connected transfer. This format is also compatible with more recent BIND-based DNS servers that run version 4.9.4 and later.

> **⊗ Caution**
>
> This property value must be specified in uppercase so that the deployment scripts can read it properly.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
TRUE	Allows BIND secondaries
FALSE	Does not allow to BIND secondaries

Expand table

Example
`[Settings] Priority=Default [Default] DNSServerOptionBINDSecondaries=TRUE`

DNSServerOptionDisableRecursion

Determines whether or not the DNS server uses recursion. By default, the DNS Server service is enabled to use recursion.

> **⊗ Caution**
>
> This property value must be specified in uppercase so that the deployment scripts can read it properly.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
TRUE	Disables recursion on the DNS server
FALSE	Enables recursion on the DNS server

Expand table

Example
`[Settings] Priority=Default [Default] DNSServerOptionDisableRecursion=TRUE`

DNSServerOptionEnableNetmaskOrdering

Determines whether the DNS server reorders address (A) resource records within the same resource record that is set in the server's response to a query based on the IP address of the source of the query.

By default, the DNS Server service uses local subnet priority to reorder A resource records.

> ⊗ **Caution**
>
> This property value must be specified in uppercase so that the deployment scripts can read it properly.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅

Component	Configured By		Scenario	Property Is Applicable
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
TRUE	Enables netmask ordering
FALSE	Disables netmask ordering

Expand table

Example
`[Settings] Priority=Default [Default] DNSServerOptionEnableNetmaskOrdering=TRUE`

DNSServerOptionEnableRoundRobin

Determines whether the DNS server uses the round robin mechanism to rotate and reorder a list of resource records if multiple resource records exist of the same type that exist for a query answer.

By default, the DNS Server service uses round robin.

> **⊗ Caution**
>
> This property value must be specified in uppercase so that the deployment scripts can read it properly.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
TRUE	Enables round robin
FALSE	Disables round robin

Expand table

Example
`[Settings] Priority=Default [Default] DNSServerOptionEnableRoundRobin=TRUE`

DNSServerOptionEnableSecureCache

Determines whether the DNS server attempts to clean up responses to avoid cache pollution. This setting is enabled by default. By default, DNS servers use a secure response option that eliminates adding unrelated resource records that are included in a referral answer to their cache. In most cases, any names that are added in referral answers are typically cached, and they help expedite the resolution of subsequent DNS queries.

With this feature, however, the server can determine that referred names are potentially polluting or insecure and then discard them. The server determines whether to cache the name that is offered in a referral on the basis of whether it is part of the exact, related, DNS domain name tree for which the original queried name was made.

> **⊗ Caution**
>
> This property value must be specified in uppercase so that the deployment scripts can read it properly.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
TRUE	Enables cache security
FALSE	Disables cache security

Expand table

Example
`[Settings] Priority=Default [Default] DNSServerOptionEnableSecureCache=TRUE`

DNSServerOptionFailOnLoad

Specifies that loading of a zone should fail when bad data is found.

> **⊗ Caution**
>
> This property value must be specified in uppercase so that the deployment scripts can read it properly.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
TRUE	Enable fail on load
FALSE	Disable fail on load

Expand table

Example
`[Settings] Priority=Default [Default] DNSServerOptionFailOnLoad=TRUE`

DNSServerOptionNameCheckFlag

Specifies which character standard is used when checking DNS names.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
0	Uses ANSI characters that comply with Internet Engineering Task Force (IETF) Request for Comments (RFCs). This value corresponds to the **Strict RFC (ANSI)** selection when configuring DNS in the Deployment Workbench.
1	Uses ANSI characters that do not necessarily comply with IETF RFCs. This value corresponds to the **Non RFC (ANSI)** selection when configuring DNS in the Deployment Workbench.
2	Uses multibyte UCS Transformation Format 8 (UTF-8) characters. This is the default setting. This value corresponds to the **Multibyte (UTF-8)** selection when configuring DNS in the Deployment Workbench.
3	Uses all characters. This value corresponds to the **All names** selection when configuring DNS in the Deployment Workbench.

Expand table

Example
`[Settings] Priority=Default [Default] DNSServerOptionNameCheckFlag=2`

DNSZones

Specifies the number of DNS zones to configure.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
zones	Specifies the number of DNS zones to configure

Expand table

Example
`[Settings] Priority=Default [Default] DNSZones=1 DNSZones0Name=MyNewZone DNSZones0DirectoryPartition=Forest DNSZones0FileName=MyNewZone.dns DNSZones0MasterIP=192.168.0.1,192.168.0.2 DNSZones0Type=Secondary`

DNSZonesxDirectoryPartition

Specifies the directory partition on which to store the zone when configuring secondary or stub zones.

> **Note**
>
> The *x* in this properties name is a placeholder for a zero-based array that contains DNS configurations.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
Domain	Replicates zone data to all DNS server in the AD DS domain
Forest	Replicates zone data to all DNS server in the AD DS forest
Legacy	Replicates zone data to all domain controllers in the AD DS domain

Expand table

Example
`[Settings] Priority=Default [Default] DNSZones0DirectoryPartition=Forest`

DNSZonesxFileName

Specifies the name of the file that will store the zone information.

> **Note**
>
> The *x* in this properties name is a placeholder for a zero-based array that contains DNS configurations.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
file_name	Specifies the name of the file that will store the zone information

Expand table

Example
`[Settings] Priority=Default [Default] DNSZones0FileName=MyNewZone.dns`

DNSZonesxMasterIP

A comma delimited list of IP addresses of the main servers to be used by the DNS server when updating the specified secondary zones. This property must be specified when configuring a secondary or stub DNS zone.

> **Note**
>
> The *x* in this properties name is a placeholder for a zero-based array that contains DNS configurations.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
IP1,IP2	A comma-delimited list of IP addresses of the main servers

Expand table

Example
`[Settings] Priority=Default [Default] DNSZones0MasterIP=192.168.0.1,192.168.0.2`

DNSZonesxName

Specifies the name of the zone.

> **Note**

> The *x* in this properties name is a placeholder for a zero-based array that contains DNS configurations.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
name	Specifies the name of the zone

Expand table

Example
`[Settings] Priority=Default [Default] DNSZones0Name=MyNewZone`

DNSZonesxScavenge

Configures the Primary DNS server to "scavenge" stale records—that is, to search the database for records that have aged and delete them.

> **Note**
>
> The *x* in this properties name is a placeholder for a zero-based array that contains DNS configurations.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		

Component	Configured By		Scenario	Property Is Applicable
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
TRUE	Allow stale DNS records to be scavenged.
FALSE	Do not allow stale DNS records to be scavenged.

Expand table

Example
`[Settings] Priority=Default [Default] DNSZones0Scavenge=TRUE`

DNSZonesxType

Specifies the type of zone to create.

> **Note**
>
> The *x* in this properties name is a placeholder for a zero-based array that contains DNS configurations.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
DSPrimary	Creates a primary zone and specifying that it should be stored in AD DS on a DNS server configured as a domain controller

Value	Description
DSStub	Creates a stub zone and specifying that it should be stored in AD DS on a DNS server configured as a domain controller
Primary	Creates a primary zone
Secondary	Creates a secondary zone
Stub	Creates a stub zone

Expand table

Example
`[Settings] Priority=Default [Default] DNSZones0Type=Secondary`

DNSZonesxUpdate

Configures the Primary DNS server to perform dynamic updates.

> **Note**
>
> The *x* in this properties name is a placeholder for a zero-based array that contains DNS configurations.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
0	Does not allow dynamic updates
1	Allows dynamic updates
2	Allows secure dynamic updates

Expand table

Example
`[Settings] Priority=Default [Default] DNSZones0Update=1`

DoCapture

Indicator of whether an image of the target computer is to be captured. If it is, Sysprep is run on the target computer to prepare for image creation. After Sysprep has run, a new WIM image is created and stored in the folder within the shared folder designated for target computer backups (**BackupDir** and **BackupShare**, respectively).

> **⊗ Caution**
>
> This property value must be specified in uppercase so that the deployment scripts can read it properly.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
YES	Copy the necessary files to run Sysprep on the target computer, run Sysprep on the target computer, and capture a WIM image.
NO	Do not run Sysprep on the target computer, and do not capture a WIM image.
PREPARE	Copy the necessary files to run Sysprep on the target computer, but do not run Sysprep or other image-capture processes.
SYSPREP	Copy the necessary files to run Sysprep on the target computer, run Sysprep on the target computer, but do not capture a WIM image. Note:

Value	Description
	The primary purpose of this value is to allow the creation of a VHD that contains an operating system after Sysprep has been run and no image capture is necessary.

Expand table

Example
`[Settings] Priority=Default [Default] DoCapture=YES DeployRoot=\\NYC-AM-FIL-01\Distribution$ ResourceRoot=\\NYC-AM-FIL-01\Resource$ UDShare=\\NYC-AM-FIL-01\MigData$ UDDir=%OSDComputerName%`

DomainAdmin

The user account credentials used to join the target computer to the domain specified in **JoinDomain**. Specify as *UserName*.

> **Note**
>
> For ZTI, the credentials that Configuration Manager specifies typically are used. If the **DomainAdmin** property is specified, the credentials in the **DomainAdmin** property override the credentials that Configuration Manager specifies.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
domain_admin	The name of the user account credentials

Expand table

Example
`[Settings] Priority=Default [Default] DomainAdmin=NYCAdmin DomainAdminDomain=WOODGROVEBANK DomainAdminPassword=<complex_password>`

DomainAdminDomain

The domain in which the user's credentials specified in **DomainAdmin** reside.

> **Note**
>
> For ZTI, the credentials that Configuration Manager specifies typically are used. If the **DomainAdmin** property is specified, the credentials in the **DomainAdmin** property override the credentials that Configuration Manager specifies.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
domain_admin_domain	The name of the domain where the user account credentials reside

Expand table

Example
`[Settings] Priority=Default [Default] DomainAdmin=NYCAdmin DomainAdminDomain=WOODGROVEBANK DomainAdminPassword=<complex_password>`

DomainAdminPassword

The password used for the domain Administrator account specified in the **DomainAdmin** property to join the computer to the domain.

> **Note**
>
> For ZTI, the credentials that Configuration Manager specifies typically are used. If the **DomainAdmin** property is specified, the credentials in the **DomainAdmin** property override the credentials that Configuration Manager specifies.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
domain_admin_password	The password for the domain Administrator account on the target computer

Expand table

Example
`[Settings] Priority=Default [Default] DomainAdmin=NYCAdmin DomainAdminDomain=WOODGROVEBANK DomainAdminPassword=<complex_password>`

DomainLevel

This entry specifies the domain functional level. This entry is based on the levels that exist in the forest when a new domain is created in an existing forest.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
Level	Sets the domain functional level to one of the following: - 2, Windows Server 2003 - 3, Windows Server 2008 - 4, Windows Server 2008 R2 - 5, Windows Server 2012

Expand table

Example
`[Settings] Priority=Default [Default] DomainLevel=3`

DomainNetBiosName

Assigns a NetBIOS name to the new domain.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
Name	Assigns a NetBIOS name to the new domain

Expand table

Example
`[Settings] Priority=Default [Default] DomainNetBiosName=NewDom`

DomainOUs

A list of AD DS organizational units (OUs) where the target computer account can be created. The **DomainOUs** property lists text values that can be any non-blank value. The **DomainOUs** property has a numeric suffix (for example, **DomainOUs1** or **DomainOUs2**). The values specified by DomainOUs will be displayed in the Deployment Wizard and selectable by the user. The **MachineObjectOU** property will then be set to the OU selected.

In addition, the same functionality can be provided by configuring the DomainOUList.xml file. The format of the DomainOUList.xml file is as follows:

XML

```
<?xml version="1.0" encoding="utf-8"?>
<DomainOUs>
<DomainOU>
  OU=Computers,OU=Tellers,OU=NYC,DC=WOODGROVEBANK,DC=Com
</DomainOU>
<DomainOU>
  OU=Computers,OU=Managers,OU=NYC,DC=WOODGROVEBANK,DC=Com
</DomainOU>
</DomainOUs>
```

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	❌	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
OU	The OU in which the target computer account can be created

Expand table

Example
`[Settings] Priority=Default [Default] OSInstall=Y DomainOUs1=OU=Computers, OU=Tellers, OU=NYC, DC=WOODGROVEBANK, DC=Com DomainOUs2=OU=Computers, OU=Managers, OU=NYC,`

Example
`DC=WOODGROVEBANK, DC=Com`

DoNotCreateExtraPartition

Specifies that deployments of Windows 7 and Windows Server 2008 R2 will not create the 300 MB system partition.

> **⊗ Caution**
>
> This property value must be specified in uppercase so that the deployment scripts can read it properly.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
YES	The additional system partition will not be created.
NO	The additional system partition will be created.

Expand table

Example
`[Settings] Priority=Default [Default] OSInstall=Y DoNotCreateExtraPartition=YES`

> **ⓘ Note**
>
> Do not use this property in conjunction with properties to configure BitLocker settings.

DoNotFormatAndPartition

This property is used to configure whether MDT performs any of the partitioning and formatting task sequence steps in task sequences created using the MDT task sequence templates.

> ⊗ **Caution**
>
> This property value must be specified in uppercase so that the deployment scripts can read it properly.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
YES	The partitioning and formatting task sequence steps in an MDT task sequence will be performed.
Any other value	The partitioning and formatting task sequence steps in an MDT task sequence will not be performed. This is the default value.

Expand table

Example
`[Settings] Priority=Default [Default] OSInstall=YES SkipUserData=YES USMTOfflineMigration=TRUE DoNotFormatAndPartition=YES OSDStateStorePath=\\WDG-MDT-01\StateStore$`

DriverGroup

A list of text values that associates out-of-box drivers created in the Deployment Workbench with each other (typically based on the make and model of a computer). A

driver can be associated with one or more driver groups. The **DriverGroup** property allows the drivers within one or more groups to be deployed to a target computer.

The text values in the list can be any non-blank value. The **DriverGroup** property value has a numeric suffix (for example, **DriverGroup001** or **DriverGroup002**). After it is defined, a driver group is associated with a computer. A computer can be associated with more than one driver group.

For example, there are two sections for each of the computer manufacturers [Mfgr01] and [Mfgr02]. Two driver groups are defined for the manufacturer Mfgr01: Mfgr01 Video Drivers and Mfgr01 Network Drivers. For the manufacturer Mfgr02, one driver group is defined, Mfgr02 Drivers. One driver group, Shared Drivers, is applied to all computers found in the [Default] section.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
driver_group_name	The name of the driver group defined in the Deployment Workbench

Expand table

Example
`[Settings] Priority=Make, Default [Default] DriverGroup001=Shared Drivers :: [Mfgr01] DriverGroup001=Mfgr01 Video Drivers DriverGroup002=Mfgr01 Network Drivers [Mfgr02] DriverGroup001=Mfgr02 Drivers`

DriverInjectionMode

This property is used to control the device drivers that are injected by the Inject Drivers task sequence step.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
Auto	Inject only matching drivers from the selection profile or folder. This is the same behavior as MDT 2008, which injects all drivers that matched one of the plug and play (PnP) identifiers (IDs) on the target computer.
All	Inject all drivers in the selection profile or folder.

Expand table

Example
`[Settings] Priority=Default [Default] DriverInjectionMode=ALL DriverSelectionProfile=Nothing DriverPaths001=\\NYC-AM-FIL-01\Drivers$ DriverPaths002=\\NYC-AM-FIL-03\WinDrvs`

DriverPaths

A list of UNC paths to shared folders where additional device drivers are located. These device drivers are installed with the target operating system on the target computer. The MDT scripts copy the contents of these folders to the C:\Drivers folder on the target computer. The **DriverPaths** property is a list of text values that can be any non-blank value. The **DriverPaths** property has a numeric suffix (for example, **DriverPaths001** or **DriverPaths002**).

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	❌	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
UNC_path	UNC path to the shared folder in which the additional drivers reside

Expand table

Example
`[Settings] Priority=Default [Default] DriverPaths001=\\NYC-AM-FIL-01\Drivers$ DriverPaths002=\\NYC-AM-FIL-03\Win8Drvs`

DriverSelectionProfile

Profile name used during driver installation.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	❌
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
profile_name	None

Expand table

Example
`[Settings] Priority=Default [Default] DriverSelectionProfile=MonitorDrivers`

EventService

The **EventService** property specifies the URL where the MDT monitoring service is running. By default, the service uses TCP port 9800 to communicate. The MDT monitoring service collects deployment information on the deployment process that can be viewed in the Deployment Workbench and using the Get-MDTMonitorData cmdlet.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
url_path	The URL to the MDT monitoring service.

Expand table

Example
`[Settings] Priority=Default [Default] EventService=https://WDG-MDT-01:9800 DeployRoot=\\NYC-AM-FIL-01\Distribution$ ResourceRoot=\\NYC-AM-FIL-01\Resource$`

EventShare

The **EventShare** property points to a shared folder in which the MDT scripts record events.

By default, the shared folder is created in C:\Events.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
UNC_path	The UNC path to the shared folder in which the MDT scripts record events. The default share name is Events.

Expand table

Example
`[Settings] Priority=Default [Default] EventShare=\\NYC-AM-FIL-01\Events DeployRoot=\\NYC-AM-FIL-01\Distribution$ ResourceRoot=\\NYC-AM-FIL-01\Resource$`

FinishAction

Specifies the action to be taken when an LTI task sequence finishes, which is after the **Summary** wizard page in the Deployment Wizard.

> **Tip**
>
> Use this property in conjunction with the **SkipFinalSummary** property to skip the **Summary** wizard page in the Deployment Wizard and automatically perform the action.

> **Caution**
>
> This property value must be specified in uppercase so that the deployment scripts can read it properly.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
action	Where *action* is one of the following: - **SHUTDOWN**. Shuts down the target computer. - **REBOOT**. Restarts the target computer.

Value	Description
	- **RESTART**. Same as **REBOOT**. - **LOGOFF**. Log off the current user. If the target computer is currently running Windows PE, then the target computer will be restarted. - **blank**. Exit the Deployment Wizard without performing any additional actions. This is the default setting.

Expand table

Example
`[Settings] Priority=Default [Default] FinishAction=REBOOT`

ForceApplyFallback

Controls the method used for installed Windows:

- **setup.exe**. This method is the traditional method, initiated by running setup.exe from the installation media. MDT uses this method by default.
- imagex.exe. This method installs the operating system image using imagex.exe with the **/apply** option. MDT uses this method when the setup.exe method cannot be used (i.e., MDT falls back to using imagex.exe).

  Besides controlling the method used to install these operating systems, this property affects which operating system task sequences are listed in the Deployment Wizard for a specific processor architecture boot image. When the value of this property is set to **NEVER**, only operating system task sequences that match the processor architecture of the boot image are displayed. If the value of this property is set to any other value or is blank, all task sequences that can use the imagex.exe installation method are shown, regardless of the processor architecture.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
NEVER	MDT always uses the imagex.exe method if necessary. Only task sequences that deploy an operating system that matches the boot image are displayed in the Deployment Wizard.
Any other value, including blank	Any task sequence that supports the imagex.exe method is displayed in the Deployment Wizard.

Expand table

Example
`[Settings] Priority=Default [Default] OSInstall=YES ForceApplyFallback=NEVER`

ForestLevel

This entry specifies the forest functional level when a new domain is created in a new forest.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
level	Sets the domain functional level to one of the following: - 2, Windows Server 2003 - 3, Windows Server 2008 - 4, Windows Server 2008 R2 - 5, Windows Server 2012

Expand table

Example
`[Settings] Priority=Default [Default] ForestLevel=3`

FullName

The full name of the user of the target computer provided during the installation of the operating system. This value is inserted into the appropriate configuration settings in Unattend.xml.

> **Note**
>
> This value is different from the user credentials created after the operating system is deployed. The **FullName** property is provided as information to systems administrators about the user running applications on the target computer.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
full_name	The full name of the user of the target computer

Expand table

Example
`[Settings] Priority=MACAddress, Default Properties=CustomProperty, ApplicationInstall [Default] CustomProperty=TRUE OrgName=Woodgrove Bank [00:0F:20:35:DE:AC] OSDNEWMACHINENAME=HPD530-1 ApplicationInstall=Custom FullName=Woodgrove Bank User [00:03:FF:FE:FF:FF] OSDNEWMACHINENAME=BVMXP ApplicationInstall=Minimum FullName=Woodgrove Bank Manager`

GPOPackPath

This property is used to override the default path to the folder in which the GPO packs reside. The path specified in this property is relative to the Templates\GPOPacks folder in a distribution share. MDT automatically scans a specific subfolder of this folder based on the operating system being deployed to the target computer, such as Templates\GPOPacks*operating_system* (where *operating_system* is the operating system being deployed). Table 3 list the supported operating systems and the subfolders that correspond to each operating system.

Table 3. Windows Operating Systems and Corresponding GPO Pack Subfolder

Expand table

Operating system	GPO pack subfolder
Windows 7 with SP1	Win7SP1-MDTGPOPack
Windows Server 2008 R2	WS2008R2SP1-MDTGPOPack

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
path	The path relative to the *distribution_share*\Templates\GPOPacks folder (where *distribution_share* is the root folder of the distribution share. The default value is the *distribution_share*\Templates\GPOPacks*operating_system* folder (where *operating_system* is a subfolder based on the operating system version). In the example below, setting the GPOPackPath property to a value of "Win7-HighSecurity" configures MDT to use the *distribution_share*\Templates\GPOPacks\Win7-HighSecurity folder as the folder where the GPO packs are stored.

Expand table

Example
`[Settings] Priority=Default [Default] GPOPackPath=Win7-HighSecurity`

Groups

The list of local groups on the target computer whose membership will be captured. This group membership is captured during the State Capture Phase and is restored during the State Restore Phase. (The default groups are Administrators and Power Users.) The **Groups** property is a list of text values that can be any non-blank value. The **Groups** property has a numeric suffix (for example, **Groups001** or **Groups002**).

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
group_name	The name of the local group on the target computer for which group membership will be captured

Expand table

Example
`[Settings] Priority=Default [Default] DeployRoot=\\NYC-AM-FIL-01\Distribution$ ResourceRoot=\\NYC-AM-FIL-01\Resource$ UDShare=\\NYC-AM-FIL-01\MigData$ CaptureGroups=YES Groups001=NYC Application Management Groups002=NYC Help Desk Users`

HideShell

This property controls the display of Windows Explorer while the LTI task sequence is running in the new operating system on the target computer. This property can be used in conjunction with the **DisableTaskMgr** property.

> **Note**
>
> This property can be used with the **DisableTaskMgr** property to help prevent users from interrupting the LTI task sequence. For more information, see the DisableTaskMgr property.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
YES	Windows Explorer is hidden until the task sequence is complete.
NO	Windows Explorer is visible while the task sequence is running. This is the default value.

Expand table

Example
`[Settings] Priority=Default [Default] DisableTaskMgr=YES HideShell=YES`

Home_Page

The URL to be used as the Windows Internet Explorer® home page after the target operating system is deployed.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
URL	The URL of the web page to be used as the home page for Internet Explorer on the target computer

Expand table

Example
`[Settings] Priority=Default [Default] Home_Page=https://portal.woodgrovebank.com`

HostName

The IP host name of the target computer (the name assigned to the target computer).

> **Note**
>
> This is the computer name of the target computer, not the NetBIOS computer name of the target computer. The NetBIOS computer name can be shorter than the computer name. Also, this property is dynamically set by MDT scripts and cannot have its value set in CustomSettings.ini or the MDT DB. Treat this property as read only. However, you can use this property within CustomSettings.ini or the MDT DB, as shown in the following examples, to aid in defining the configuration of the target computer.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	❌	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
host_name	The IP host name assigned to the target computer

Expand table

Example
None

ImagePackageID

The package ID used for the operating system to install during OEM deployments.

> **Note**
>
> This property is dynamically set by the MDT scripts and is not configured in CustomSettings.ini or the MDT DB. Treat this property as read only.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	❌	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
None	The package ID used for the operating system to install during OEM deployments

Expand table

Example
None

InputLocale

A list of input locales to be used with the target operating system. More than one input locale can be specified for the target operating system. Each locale must be separated by a semicolon (;). If not specified, the Deployment Wizard uses the input locale configured in the image being deployed.

Exclude this setting in the Windows User State Migration Tool (USMT) when backing up and restoring user state information. Otherwise, the settings in the user state information will override the values specified in the *InputLocale* property.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
input_locale1; input_locale2	The locale for the keyboard attached to the target computer

Expand table

Example
`[Settings] Priority=Default [Default] UserLocale=en-us InputLocale=0409:00000409;0413:00020409;0413:00000409;0409:00020409`

InstallPackageID

The package ID used for the operating system to install during OEM deployments.

> **Note**
>
> This property is dynamically set by the MDT scripts and is not configured in CustomSettings.ini or the MDT DB. Treat this property as read only.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	❌	|		

Component	Configured By		Scenario	Property Is Applicable
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
None	The package ID used for the operating system to install during OEM deployments

Expand table

Example
None

Instance

The instance of SQL Server used for querying property values from columns in the table specified in the **Table** property. The database resides on the computer specified in the **SQLServer** property. The instance of SQL Server on the computer is specified in the **Instance** property.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	✅	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
	instance

Expand table

Example
`[Settings] Priority=Computers, Default [Default] OSInstall=YES [Computers] SQLServer=NYC-SQL-01 Database=MDTDB Instance=SQLEnterprise2005 Table=Computers Parameters=SerialNumber, AssetTag ParameterCondition=OR`

IPAddress

The IP address of the target computer. The format of the IP address returned by the property is standard dotted-decimal notation; for example, 192.168.1.1. Use this property to create a subsection that contains settings targeted to a specific target computer based on the IP address.

> **Note**
>
> This property is dynamically set by the MDT scripts and is not configured in CustomSettings.ini or the MDT DB. Treat this property as read only.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	❌	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
ip_address	The IP address of the target computer in standard dotted-decimal notation

Expand table

Example
None

IsDesktop

Indicator of whether the computer is a desktop, because the **Win32_SystemEnclosure ChassisType** property value is **3**, **4**, **5**, **6**, **7**, **15**, **16**, **35**, or **36**.

> ⓘ Note
>
> Only one of the following properties will be true at a time: **IsDesktop**, **IsLaptop**, **IsServer**.

> ⓘ Note
>
> This property is dynamically set by the MDT scripts and is not configured in CustomSettings.ini or the MDT DB. Treat this property as read only.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	❌	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
TRUE	The target computer is a desktop computer.
FALSE	The target computer is not a desktop computer.

Expand table

Example
None

IsHypervisorRunning

Specifies whether a hypervisor is present on the target computer. This property is set using information from the **CPUID** interface.

For further information collected about VMs and information returned from the **CPUID** interface, see the following properties:

- **IsVM**
- **SupportsHyperVRole**
- **SupportsNX**
- **SupportsVT**
- **Supports64Bit**
- **VMPlatform**

> ⓘ Note
>
> This property is dynamically set by the MDT scripts and is not configured in CustomSettings.ini or the MDT DB. Treat this property as read only.

> ⓘ Note
>
> The IsVM property should be used to determine whether the target computer is a virtual or physical machine.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	❌	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
TRUE	A hypervisor is detected.
FALSE	A hypervisor is not detected.

Expand table

Example
None

IsLaptop

Indicator of whether the computer is a portable computer, because the **Win32_SystemEnclosure ChassisType** property value is **8**, **9**, **10**, **11**, **12**, **14**, **18**, **21**, **30**, **31**, or **32**.

> **Note**
>
> Only one of the following properties will be true at a time: **IsDesktop**, **IsLaptop**, **IsServer**.

> **Note**
>
> This property is dynamically set by the MDT scripts and is not configured in CustomSettings.ini or the MDT DB. Treat this property as read only.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	❌	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
TRUE	The target computer is a portable computer.
FALSE	The target computer is not a portable computer.

Expand table

Example
None

IsServer

Indicator of whether the computer is a server, because the **Win32_SystemEnclosure ChassisType** property value is **23** or **28**.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	❌	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
TRUE	The target computer is a server.
FALSE	The target computer is not a server.

Expand table

Example
None

IsServerCoreOS

Indicator of whether the current operating system running on the target computer is the Server Core installation option of the Windows Server operating system.

> **Note**
>
> This property is dynamically set by the MDT scripts and is not configured in CustomSettings.ini or the MDT DB. Treat this property as read only.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	❌	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
TRUE	The operating system on the target computer is the Server Core installation option of Windows Server.
FALSE	The operating system on the target computer is not the Server Core installation option of Windows Server.

Expand table

Example
None

IsServerOS

Indicator of whether the current operating system running on the target computer is a server operating system.

> **Note**
>
> This property is dynamically set by the MDT scripts and is not configured in CustomSettings.ini or the MDT DB. Treat this property as read only.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	❌	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
TRUE	The operating system on the target computer is a server operating system.
FALSE	The operating system on the target computer is not a server operating system.

Expand table

Example
None

IsUEFI

Specifies whether the target computer is currently running with Unified Extensible Firmware Interface (UEFI). The UEFI is a specification that defines a software interface between an operating system and platform firmware. UEFI is a more secure replacement for the older BIOS firmware interface present in some personal computers. For more information on UEFI, go to https://uefi.org.

> **Note**
>
> This property is dynamically set by the MDT scripts and is not configured in CustomSettings.ini or the MDT DB. Treat this property as read only.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	❌	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
TRUE	The target computer is currently running with UEFI.

Value	Description
FALSE	The target computer is not currently running with UEFI. Note: It is possible that the target computer may support UEFI, but is running in a compatibility mode that emulates the older BIOS firmware interface. In this situation this value of this property will set to **FALSE** even though the target computer supports UEFI.

Expand table

Example
None

IsVM

Specifies whether the target computer is a VM based on information gathered from the **CPUID** interface. You can determine the specific VM environment using the **VMPlatform** property.

For further information collected about VMs and information returned from the **CPUID** interface, see the following properties:

- **IsHypervisorRunning**
- **SupportsHyperVRole**
- **SupportsNX**
- **SupportsVT**
- **Supports64Bit**
- **VMPlatform**

> **Note**
>
> This property is dynamically set by the MDT scripts and is not configured in CustomSettings.ini or the MDT DB. Treat this property as read only.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	❌	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
TRUE	The target computer is a VM.
FALSE	The target computer is not a VM.

Expand table

Example
None

JoinDomain

The domain that the target computer joins after the target operating system is deployed. This is the domain where the computer account for the target computer is created. The **JoinDomain** property can contain alphanumeric characters, hyphens (-), and underscores (_). The **JoinDomain** property cannot be blank or contain spaces.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
domain_name	The name of the domain that the target computer joins

Expand table

Example
`[Settings] Priority=Default [Default] JoinDomain=WOODGROVEBANK MachineObjectOU=OU=Reception,OU=NYC,DC=Woodgrovebank,DC=com`

JoinWorkgroup

The workgroup that the target computer joins after the target operating system is deployed. The **JoinWorkgroup** property can contain alphanumeric characters, hyphens (-), and underscores (_). The **JoinWorkgroup** property cannot be blank or contain spaces.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
workgroup_name	The name of the workgroup that the target computer joins

Expand table

Example
`[Settings] Priority=Default [Default] JoinWorkgroup=WDGV_WORKGROUP`

KeyboardLocale

A list of keyboard locales to be used with the target operating system. More than one keyboard locale can be specified for the target operating system. Each locale must be separated by a semicolon (;). If not specified, the Deployment Wizard uses the keyboard locale configured in the image being deployed.

Exclude this setting in USMT when backing up and restoring user state information. Otherwise, the settings in the user state information will override the values specified in the **KeyboardLocale** property.

> **Note**
>
> For this property to function properly, it must be configured in both CustomSettings.ini and BootStrap.ini. BootStrap.ini is processed before a deployment share (which contains CustomSettings.ini) has been selected.

Expand table

Component	Configured By	Scenario	Property Is Applicable
BootStrap.ini	✅	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅		
MDT DB	✅	ZTI (Configuration Manager)	✅

Expand table

Value	Description
keyboard_locale1; keyboard_locale2	The locale of the keyboard attached to the target computer. The value can be specified in the following formats: - Text (en-us) - Hexadecimal (0409:00000409)

Expand table

Example 1
`[Settings] Priority=Default [Default] UserLocale=en-us KeyboardLocale=en-us`

Expand table

Example 2
`[Settings] Priority=Default [Default] UserLocale=en-us KeyboardLocale=0409:00000409;1809:00001809;041A:0000041A;083b:0001083b`

KeyboardLocalePE

The name of the keyboard locale to be used while in Windows PE only.

> **Note**
>
> For this property to function properly, it must be configured in both CustomSettings.ini and BootStrap.ini. BootStrap.ini is processed before a deployment share (which contains CustomSettings.ini) has been selected.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	✅	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
keyboard_locale	The locale of the keyboard attached to the target computer. The value can be specified in the following formats: - Text (en-us) - Hexadecimal (0409:00000409)

Expand table

Example 1
`[Settings] Priority=Default [Default] KeyboardLocalePE=en-us`

Expand table

Example 2
`[Settings] Priority=Default [Default] KeyboardLocalePE=0409:00000409`

LanguagePacks

A list of the GUIDs for the language packs to be deployed on the target computer. Deployment Workbench specifies these language packs on the OS Packages node. These GUIDs are stored in the Packages.xml file. The **LanguagePacks** property has a numeric suffix (for example, **LanguagePacks001** or **LanguagePacks002**).

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	❌	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
language_pack_guid	The GUID that the Deployment Workbench specifies for the language packs to install on the target computer. The GUID corresponds to the language pack GUID stored in Packages.xml.

Expand table

Example
`[Settings] Priority=Default [Default] LanguagePacks001={a1923f8d-b07b-44c7-ac1e-353b7cc4c1ad}`

LoadStateArgs

The arguments passed to the USMT Loadstate process. The ZTI script inserts the appropriate logging, progress, and state store parameters. If this value is not included in the settings file, the user state restore process is skipped.

If the Loadstate process finishes successfully, the user state information is deleted. In the event of a Loadstate failure (or non-zero return code), the local state store is moved to %WINDIR%\StateStore to prevent deletion and to ensure that no user state information is lost.

> ⓘ Note

Do not add any of the following command-line arguments when configuring this property: **/hardlink**, **/nocompress**, **/decrypt**, **/key**, or **/keyfile**. The MDT scripts will add these command-line arguments if applicable to the current deployment scenario.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
Arguments	The command-line arguments passed to Loadstate.exe. The default arguments specified by Deployment Workbench are as follows: - **/v**. Enables verbose output in the Loadstate log. The default is **0**. Specify any number from 0 to 15. The value **5** enables verbose and status output. - **/c**. When specified, Loadstate will continue to run even if there are nonfatal errors. Without the **/c** option, Loadstate exits on the first error. - **/lac**. Specifies that if the account being migrated is a local (non-domain) account, and it does not exist on the destination computer, then USMT will create the account but it will be disabled. For more information about these and other arguments, see the USMT Help files.

Expand table

Example
`[Settings] Priority=Default [Default] OSInstall=YES ScanStateArgs=/v:5 /o /c LoadStateArgs=/v:5 /c /lac DeployRoot=\\NYC-AM-FIL-01\Distribution$ ResourceRoot=\\NYC-AM-FIL-01\Resource$ UDShare=\\NYC-AM-FIL-01\MigData$ UDDir=%OSDComputerName%`

Location

The geographic location of the target computers. A list of IP addresses that correspond to the default gateways defined for the computers within that location defines the **Location** property. An IP address for a default gateway can be associated with more than one location.

Typically, the value for the **Location** property is set by performing a database query on the database managed using Deployment Workbench. Deployment Workbench can assist in creating the locations, defining property settings associated with the locations, and then in configuring CustomSettings.ini to perform the database query for the Location property and the property settings associated with the locations.

For example, a `LocationSettings` section in CustomSettings.ini can query the LocationSettings view in the database for a list of locations that contain the value specified in the **DefaultGateway** property listed in the **Parameters** property. The query returns all settings associated with each default gateway.

Then the scripts parse each section that corresponds to the locations returned in the query. For example, the value `[Springfield]` and the section `[Springfield-123 Oak Street-4th Floor]` in CustomSettings.ini can represent the corresponding locations. This is an example of how one computer can belong to two locations. The `[Springfield]` section is for all computers in a larger geographic area (an entire city), and the `[Springfield-123 Oak Street-4th Floor]` section is for all computers on the fourth floor at 123 Oak Street, in Springfield.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	✅	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
location1,location2	The list of locations to be assigned to an individual computer or a group of computers

Expand table

Example
`[Settings] Priority=LSettings, Default [Default] UserDataLocation=AUTO DeployRoot=\\W2K3-SP1\Distribution$ OSInstall=YES ScanStateArgs=/v:15 /o /c LoadStateArgs=/v:7 /c [LSettings] SQLServer=w2k3-sp1 Instance=MDT2010 Database=MDTDB Netlib=DBNMPNTW SQLShare=SQL$ Table=LocationSettings Parameters=DefaultGateway [Springfield] UDDir=%OSDComputerName% UDShare=\\Springfield-FIL-01\UserData [Springfield-123 Oak Street-4th Floor] DeployRoot=\\Springfield-BDD-01\Distribution1$`

LongDistanceAccess

The dialing digits to gain access to an outside line to dial long distance. The property can contain only numeric digits. This value is inserted into the appropriate configuration settings in Unattend.xml.

Expand table

Component	Configured By	Scenario	Property Is Applicable
BootStrap.ini	❌	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅		
MDT DB	✅	ZTI (Configuration Manager)	✅

Expand table

Value	Description
language_pack_guid	The GUID that the Deployment Workbench specifies for the language packs to install on the target computer. The GUID corresponds to the language pack GUID stored in Packages.xml.

Expand table

Example
`[Settings] Priority=Default [Default] AreaCode=206 CountryCode=001 Dialing=TONE LongDistanceAccess=9`

MACAddress

The media access control (MAC) layer address of the primary network adapter of the target computer. The **MACAddress** property is included on the **Priority** line so that

property values specific to a target computer can be provided. Create a section for each MAC address for each of the target computers (such as `[00:0F:20:35:DE:AC]` or `[00:03:FF:FE:FF:FF]`) that contain target computer-specific settings.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
mac_address	The MAC address of the target computer

Expand table

Example
`[Settings] Priority=MACAddress, Default [Default] CaptureGroups=YES Groups1=NYC Application Management Groups2=NYC Help Desk Users [00:0F:20:35:DE:AC] OSDNEWMACHINENAME=HPD530-1 [00:03:FF:FE:FF:FF] OSDNEWMACHINENAME=BVMXP`

MachineObjectOU

The AD DS OU in the target domain where the computer account for the target computer is created.

> ⓘ **Note**
>
> The OU specified in this property must exist before deploying the target operating system.

> ⓘ **Note**
>
> If a computer object already exists in AD DS, specifying **MachineObjectOU** will not cause the computer object to be moved to the specified OU.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
OU_name	The name of the OU where the computer account for the target computer will be created

Expand table

Example
`[Settings] Priority=Default [Default] JoinDomain=WOODGROVEBANK MachineObjectOU=OU=Reception,OU=NYC,DC=Woodgrovebank,DC=com`

Make

The manufacturer of the target computer. The format for **Make** is undefined. Use this property to create a subsection that contains settings targeted to a specific computer manufacturer (most commonly in conjunction with the **Model** and **Product** properties).

> **Note**
>
> This property is dynamically set by MDT scripts and cannot have its value set in CustomSettings.ini or the MDT DB. Treat this property as read only. However, you can use this property within CustomSettings.ini or the MDT DB, as shown in the following examples, to aid in defining the configuration of the target computer.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	❌	|		

Component	Configured By		Scenario	Property Is Applicable
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
make	The manufacturer of the target computer

Expand table

Example
`[Settings] Priority=Make, Default [Default] [Dell Computer Corporation] Subsection=Dell-%Model% [Dell-Latitude D600] Packages001=XXX00009:Program9 Packages002=XXX0000A:Program10`

MandatoryApplications

A list of application GUIDs that will be installed on the target computer. These applications are specified on the Applications node in the Deployment Workbench. The GUIDs are stored in the Applications.xml file. The **MandatoryApplications** property is a list of text values that can be any non-blank value. The **MandatoryApplications** property has a numeric suffix (for example, **MandatoryApplications001** or **MandatoryApplications002**).

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
application_guid	The GUID specified by the Deployment Workbench for the application to be deployed to the target computer. The GUID corresponds to the application GUID stored in the Applications.xml file.

Expand table

Example
`[Settings] Priority=Default [Default] MandatoryApplications001={1D7DF331-47B7-472C-87B3-442597EC2F7D} MandatoryApplications002={9d2b8999-5e4d-4f3d-bb05-edaaf4fe5628} Administrators001=WOODGROVEBANK\NYC Help Desk Staff`

Memory

The amount of memory installed on the target computer in megabytes. For example, the value **2038** indicates 2,038 MB (or 2 GB) of memory is installed on the target computer.

> **Note**
>
> This property is dynamically set by the MDT scripts and is not configured in CustomSettings.ini or the MDT DB. Treat this property as read only.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	❌	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
memory	The amount of memory installed on the target computer in megabytes

Expand table

Example
None

Model

The model of the target computer. The format for **Model** is undefined. Use this property to create a subsection that contains settings targeted to a specific computer model number for a specific computer manufacturer (most commonly in conjunction with the **Make** and **Product** properties).

> **Note**
>
> This property is dynamically set by MDT scripts and cannot have its value set in CustomSettings.ini or the MDT DB. Treat this property as read only. However, you can use this property within CustomSettings.ini or the MDT DB, as shown in the following examples, to aid in defining the configuration of the target computer.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	❌	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
model	The model of the target computer

Expand table

Example
`[Settings] Priority=Make, Default [Default] [Dell Computer Corporation] Subsection=Dell-%Model% [Dell-Latitude D600] Packages001=XXX00009:Program9 Packages002=XXX0000A:Program10`

NetLib

The protocol to be used to communicate with the computer running SQL Server specified in the **SQLServer** property.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	✅	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
DBNMPNTW	Use the named pipes protocol to communicate.
DBMSSOCN	Use TCP/IP sockets to communicate.

Expand table

Example
`[Settings] Priority=Computers, Default [Default] ScanStateArgs=/v:5 /o /c LoadStateArgs=/v:5 /c /lac [Computers] SQLServer=NYC-SQL-01 SQLShare=SQL$ NetLib=DBNMPNTW Database=MDTDB Instance=SQLEnterprise2005 Table=Computers Parameters=SerialNumber, AssetTag ParameterCondition=OR`

NewDomain

Indicates the type of a new domain: whether a new domain in a new forest, the root of a new tree in an existing forest, or a child of an existing domain.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
Child	The new domain is a child of an existing domain.

Value	Description
Forest	The new domain is the first domain in a new forest of domain trees.
Tree	The new domain is the root of a new tree in an existing forest.

Expand table

Example
`[Settings] Priority=Default [Default] NewDomain=Tree`

NewDomainDNSName

Specifies the required name of a new tree in an existing domain or when Setup installs a new forest of domains.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
name	Specifies the required name of a new tree in an existing domain or when Setup installs a new forest of domains

Expand table

Example
`[Settings] Priority=Default [Default] NewDomainDNSName=newdom.WoodGroveBank.com`

Order

The sorting order for the result set on a database query. The result set is based on the configuration settings of the **Database**, **Table**, **SQLServer**, **Parameters**, and

ParameterCondition properties. More than one property can be provided to sort the results by more than one property.

For example, if **Order=Sequence** is specified in the CustomSettings.ini file, then an **ORDER BY** sequence clause is added to the query. **Specifying Order=Make, Model** adds an **ORDER BY Make, Model** clause to the query.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	✅	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
property1, property2, ...	Properties to define the sort order for the result set (where *propertyn* represents the properties in the sort criteria)

Expand table

Example
`[Settings] Priority=Computers, Default [Default] OSInstall=YES ScanStateArgs=/v:5 /o /c LoadStateArgs=/v:5 /c /lac [Computers] SQLServer=NYC-SQL-01 SQLShare=SQL$ NetLib=DBNMPNTW Database=MDTDB Instance=SQLEnterprise2005 Table=MakeModelSettings Parameters=SerialNumber, AssetTag ParameterCondition=OR Order=Make, Model`

OrgName

The name of the organization that owns the target computer. This value is inserted into the appropriate configuration settings in Unattend.xml.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		

Component	Configured By		Scenario	Property Is Applicable
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
org_name	The name of the organization that owns the target computer

Expand table

Example
`[Settings] Priority=MACAddress, Default Properties=CustomProperty, ApplicationInstall [Default] OSInstall=YES ScanStateArgs=/v:5 /o /c LoadStateArgs=/v:5 /c /lac UserDataLocation=NONE CustomProperty=TRUE OrgName=Woodgrove Bank [00:0F:20:35:DE:AC] OSDNEWMACHINENAME=HPD530-1 ApplicationInstall=Custom FullName=Woodgrove Bank User [00:03:FF:FE:FF:FF] OSDNEWMACHINENAME=BVMXP ApplicationInstall=Minimum FullName=Woodgrove Bank Manager`

OSArchitecture

The processor architecture type for the target operating system. This property is referenced during OEM deployments. Valid values are **x86** and **x64**.

> ⓘ **Note**
>
> This property is dynamically set by the MDT scripts and is not configured in CustomSettings.ini or the MDT DB. Treat this property as read only.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	❌	|		
MDT DB	❌	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
x86	The processor architecture type for the operating system is 32 bit.
x64	The processor architecture type for the operating system is 64 bit.

Expand table

Example
None

OSCurrentBuild

The build number of the currently running operating system.

> **Note**
>
> This property is dynamically set by the MDT scripts and is not configured in CustomSettings.ini or the MDT DB. Treat this property as read only.

Expand table

Component	Configured By	Scenario	Property Is Applicable
BootStrap.ini	❌	LTI (Stand-alone MDT)	✅
CustomSettings.ini	❌		
MDT DB	❌	ZTI (Configuration Manager)	✅

Expand table

Value	Description
7600	Windows 7
9600	Windows 8.1

Expand table

Example
None

OSCurrentVersion

The version number of the currently running operating system.

> **Note**
>
> This property is dynamically set by the MDT scripts and is not configured in CustomSettings.ini or the MDT DB. Treat this property as read only.

Expand table

Component	Configured By	Scenario	Property Is Applicable
BootStrap.ini	❌	LTI (Stand-alone MDT)	✅
CustomSettings.ini	❌		
MDT DB	❌	ZTI (Configuration Manager)	✅

Expand table

Value	Description
version_number	The operating system major version, minor version, and build numbers (major.minor.build). For example, 6.3.9600 would represent Windows 8.1.

Expand table

Example
None

OSDAdapterxDescription

Specifies the name of the network connection as it appears in the Control Panel Network Connections item. The name can be between 0 and 255 characters in length.

This property is for LTI only. For the equivalent property for ZTI, see OSDAdapterxName.

> **Note**
>
> The*x*in this properties name is a placeholder for a zero-based array that contains network adapter information, such as **OSDAdapter0Description** or

OSDAdapter1Description.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	❌	|		
MDT DB	❌	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
Description	The name of the network connection as it appears in the Control Panel Network Connections item

Expand table

Example
None

OSDAdapterxDNSDomain

Specifies the DNS domain name (DNS suffix) that will be assigned to the network connection. This property is for ZTI only. For LTI, see the OSDAdapterxDNSSuffix property.

> ⓘ **Note**
>
> The*x*in this properties name is a placeholder for a zero-based array that contains network adapter information, such as **OSDAdapter0DNSDomain** or **OSDAdapter1DNSDomain**.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	❌

Component	Configured By		Scenario	Property Is Applicable
CustomSettings.ini	✅			
MDT DB	✅		ZTI (Configuration Manager)	✅

Expand table

Value	Description
DNS_domain_name	A DNS domain name (DNS suffix) that will be assigned to the network connection

Expand table

Example
`[Settings] Priority=Default [Default] OSDAdapter0DNSDomain=WoodGroveBank.com`

OSDAdapterxDNSServerList

This is a comma-delimited list of DNS server IP addresses that will be assigned to the network connection.

> ⓘ Note
>
> The*x*in this properties name is a placeholder for a zero-based array that contains network adapter information, such as **OSDAdapter0DNSServerList** or **OSDAdapter1DNSServerList**.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌		LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅			
MDT DB	✅		ZTI (Configuration Manager)	✅

Expand table

Value	Description
DNS_servers	A comma-delimited list of DNS server IP addresses that will be assigned to the network connection

Expand table

Example
`[Settings] Priority=Default [Default]` `OSDAdapter0DNSServerList=192.168.0.254,192.168.100.254`

OSDAdapterxDNSSuffix

A DNS suffix that will be assigned to the network connection. This property is for LTI only. For ZTI, see the OSDAdapterxDNSDomain property.

> **Note**
>
> The*x*in this properties name is a placeholder for a zero-based array that contains network adapter information, such as **OSDAdapter0DNSSuffix** or **OSDAdapter1DNSSuffix**.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
DNS_suffix	A DNS suffix that will be assigned to the network connection

Expand table

Example
`[Settings] Priority=Default [Default] OSDAdapter0DNSSuffix= WoodGroveBank.com`

OSDAdapterxEnableDHCP

Specifies whether the network connection will be configured via DHCP.

> **Note**
>
> The*x*in this properties name is a placeholder for a zero-based array that contains network adapter information, such as **OSDAdapter0EnableDHCP** or **OSDAdapter1EnableDHCP**.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
TRUE	The network connection will be configured via DHCP.
FALSE	The network connection will be configured with static configuration.

Expand table

Example
`[Settings] Priority=Default [Default] OSDAdapter0EnableDHCP=TRUE`

OSDAdapterxEnableDNSRegistration

Specifies whether DNS registration is enabled on the network connection.

> **Note**
>
> Thexin this properties name is a placeholder for a zero-based array that contains network adapter information, such as **OSDAdapter0EnableDNSRegistration** or **OSDAdapter1EnableDNSRegistration**.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
TRUE	Enables DNS registration
FALSE	Disables DNS registration

Expand table

Example
`[Settings] Priority=Default [Default] OSDAdapter0EnableDNSRegistration=TRUE`

OSDAdapterxEnableFullDNSRegistration

Specifies whether full DNS registration is enabled on the network connection.

> **Note**
>
> Thexin this properties name is a placeholder for a zero-based array that contains network adapter information, such as **OSDAdapter0EnableFullDNSRegistration** or **OSDAdapter1EnableFullDNSRegistration**.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
TRUE	Enables full DNS registration
FALSE	Disables full DNS registration

Expand table

Example
`[Settings] Priority=Default [Default] OSDAdapter0EnableFullDNSRegistration=TRUE`

OSDAdapterxEnableLMHosts

Specifies whether LMHOSTS lookup is enabled on the network connection.

> **Note**
>
> The*x*in this properties name is a placeholder for a zero-based array that contains network adapter information, such as **OSDAdapter0EnableLMHosts** or **OSDAdapter1EnableLMHosts**.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
TRUE	Enables LMHOSTS lookup
FALSE	Disables LMHOSTS lookup

Expand table

Example
`[Settings] Priority=Default [Default] OSDAdapter0EnableLMHosts=TRUE`

OSDAdapterxEnableIPProtocolFiltering

This property specifies whether IP protocol filtering should be enabled on the network connection.

The*x* in this property's name is a placeholder for a zero-based array that contains network adapter information, such as *OSDAdapter0EnableIPProtocolFiltering* or **OSDAdapter1EnableIPProtocolFiltering**.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
TRUE	Enables IP protocol filtering
FALSE	Disables IP protocol filtering

Expand table

Example
`[Settings] Priority=Default [Default] OSDAdapter0EnableIPProtocolFiltering =TRUE`

OSDAdapterxEnableTCPFiltering

Specifies whether TCP/IP filtering should be enabled on the network connection. This property is for ZTI only. For LTI, see the OSDAdapterxEnableTCPIPFiltering property.

> **Note**
>
> The*x*in this property's name is a placeholder for a zero-based array that contains network adapter information, such as **OSDAdapter0EnableTCPFiltering** or **OSDAdapter1EnableTFiltering**.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	❌
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
TRUE	Enables TCP/IP filtering
FALSE	Disables TCP/IP filtering

Expand table

Example
`[Settings] Priority=Default [Default] OSDAdapter0EnableTCPFiltering=TRUE`

OSDAdapterxEnableTCPIPFiltering

Specifies whether TCP/IP filtering should be enabled on the network connection. This property is for LTI only. For ZTI, see the OSDAdapterxEnableTCPFiltering property.

> **Note**

Thexin this properties name is a placeholder for a zero-based array that contains network adapter information, such as **OSDAdapter0EnableTCPIPFiltering** or **OSDAdapter1EnableTCPIPFiltering**.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
TRUE	Enables TCP/IP filtering
FALSE	Disables TCP/IP filtering

Expand table

Example
`[Settings] Priority=Default [Default] OSDAdapter0EnableTCPIPFiltering=TRUE`

OSDAdapterxEnableWINS

Specifies whether WINS will be enabled on the network connection.

> ⓘ Note
>
> Thexin this properties name is a placeholder for a zero-based array that contains network adapter information, such as **OSDAdapter0EnableWINS** or **OSDAdapter1EnableWINS**.

> ⊗ Caution
>
> This property value must be specified in uppercase letters so that the deployment scripts can properly read it.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
TRUE	Enables WINS
FALSE	Disables WINS

Expand table

Example
`[Settings] Priority=Default [Default] OSDAdapter0EnableWINS=TRUE OSDAdapter0WINSServerList=192.168.0.1,192.168.100.1`

OSDAdapterxGatewayCostMetric

A comma-delimited list of Gateway Cost Metrics specified as either integers or the string "Automatic" (if empty, uses "Automatic") that will be configured on the connection.

> ⓘ Note
>
> The*x*in this properties name is a placeholder for a zero-based array that contains network adapter information, such as **OSDAdapter0GatewayCostMetric** or **OSDAdapter1GatewayCostMetric**.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
cost_metrics	A comma-delimited list of Gateway Cost Metrics

Expand table

Example
`[Settings] Priority=Default [Default] OSDAdapter0GatewayCostMetrics=Automatic`

OSDAdapterxGateways

A comma-delimited list of gateways to be assigned to the network connection.

> **Note**
>
> The*x*in this properties name is a placeholder for a zero-based array that contains network adapter information, such as **OSDAdapter0Gateways** or **OSDAdapter1Gateways**.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
gateways	A comma-delimited list of gateways

Expand table

Example
`[Settings] Priority=Default [Default] OSDAdapter0Gateways=192.168.0.1,192.168.100.1`

OSDAdapterxIPAddressList

A comma-delimited list of IP addresses to be assigned to the network connection.

> **Note**
>
> The*x*in this properties name is a placeholder for a zero-based array that contains network adapter information, such as **OSDAdapter0IPAddressList** or **OSDAdapter1IPAddressList**.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
IP_addresses	A comma delimited list of IP addresses

Expand table

Example
`[Settings] Priority=Default [Default] OSDAdapter0IPAddressList=192.168.0.40,192.168.100.40 OSDAdapter0SubnetMask=255.255.255.0,255.255.255.0`

OSDAdapterxIPProtocolFilterList

A comma-delimited list of IP protocol filters to be assigned to the network connection. This property can be configured using the CustomSettings.ini file or the MDT DB but not the Deployment Workbench. If using Configuration Manager it is also configurable using an **Apply Network Settings** task sequence step.

> **Note**

> Thex*in* this properties name is a placeholder for a zero-based array that contains network adapter information, such as **OSDAdapter0IPProtocolFilterList** or **OSDAdapter1IPProtocolFilterList**.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
protocol_filter_list	A comma-delimited list of IP protocol filters

Expand table

Example
`[Settings] Priority=Default [Default] OSDAdapter0IPProtocolFilterList=a list of approved IP protocols`

OSDAdapterxMacAddress

Assign the specified configuration settings to the network interface card that matches the specified MAC address.

> ⓘ Note
>
> Thex*in* this properties name is a placeholder for a zero-based array that contains network adapter information, such as **OSDAdapter0MacAddress** or **OSDAdapter1MacAddress**.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
MAC_address	Network adapter MAC address

Expand table

Example
`[Settings] Priority=Default [Default] OSDAdapter0MacAddress=00:0C:29:67:A3:6B`

OSDAdapterxName

Assign the specified configuration settings to the network adapter that matches the specified name. This property is for ZTI only. For the equivalent property for LTI, see OSDAdapterxDescription.

> ⓘ **Note**
>
> The*x*in this properties name is a placeholder for a zero-based array that contains network adapter information, such as **OSDAdapter0Name** or **OSDAdapter1Name**.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	❌
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
name	Network adapter name

Expand table

Example
`[Settings] Priority=Default [Default] OSDAdapter0Name=3Com 3C920 Integrated Fast Ethernet Controller`

OSDAdapterxSubnetMask

A comma-delimited list of IP subnet masks to be assigned to the network connection.

> **Note**
>
> The*x*in this properties name is a placeholder for a zero-based array that contains network adapter information, such as **OSDAdapter0SubnetMask** or **OSDAdapter1SubnetMask**.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
subnet_masks	A comma-delimited list of IP subnet masks

Expand table

Example
`[Settings] Priority=Default [Default] OSDAdapter0IPAddressList=192.168.0.40,192.168.100.40 OSDAdapter0SubnetMask=255.255.255.0,255.255.255.0`

OSDAdapterxTCPFilterPortList

A comma-delimited list of TCP filter ports to be assigned to the network connection. This property can be configured using the CustomSettings.ini file or the MDT DB but not the Deployment Workbench. If using Configuration Manager it is also configurable using an **Apply Network Settings** task sequence step.

> ⓘ Note
>
> The*x*in this properties name is a placeholder for a zero-based array that contains network adapter information, such as **OSDAdapter0TCPFilterPortList** or **OSDAdapter1TCPFilterPortList**.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
port_list	A comma-delimited list of TCP/IP filter ports

Expand table

Example
`[Settings] Priority=Default [Default] OSDAdapter0TCPFilterPortList=a list of approved TCP ports`

OSDAdapterxTCPIPNetBiosOptions

Specifies the TCP/IP NetBIOS options to be assigned to the network connection.

> ⓘ Note

Thexin this properties name is a placeholder for a zero-based array that contains network adapter information, such as **OSDAdapter0TCPIPNetBiosOptions** or **OSDAdapter1TCPIPNetBiosOptions**.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
0	Disable IP forwarding.
1	Enable IP forwarding.

Expand table

Example
`[Settings] Priority=Default [Default] OSDAdapter0TCPIPNetBiosOptions=0`

OSDAdapterxUDPFilterPortList

A comma-delimited list of User Datagram Protocol (UDP) filter ports to be assigned to the network connection. This property can be configured using the CustomSettings.ini file and the MDT DB but not the Deployment Workbench. If using Configuration Manager it is also configurable using an **Apply Network Settings** task sequence step.

> **Note**
>
> Thexin this properties name is a placeholder for a zero-based array that contains network adapter information, such as **OSDAdapter0UDPFilterPortList** or **OSDAdapter1UDPFilterPortList**.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
port_list	A comma-delimited list of UDP filter ports

Expand table

Example
`[Settings] Priority=Default [Default] OSDAdapter0UDPFilterPortList=a list of approved UDP ports`

OSDAdapterxWINSServerList

A two-element, comma-delimited list of WINS server IP addresses to be assigned to the network connection.

> **Note**
>
> The*x*in this properties name is a placeholder for a zero-based array that contains network adapter information, such as **OSDAdapter0WINSServerList** or **OSDAdapter1WINSServerList**.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
WINS_server_list	A comma-delimited list of WINS server IP addresses

Expand table

Example
`[Settings] Priority=Default [Default] OSDAdapter0EnableWINS=TRUE OSDAdapter0WINSServerList=192.168.0.1,192.168.100.1`

OSDAdapterCount

Specifies the number of network connections that are to be configured.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
count	The number of network adapters

Expand table

Example
`[Settings] Priority=Default [Default] OSDAdapterCount=1 OSDAdapter0EnableDHCP=FALSE OSDAdapter0IPAddressList=192.168.0.40,192.168.100.40 OSDAdapter0SubnetMask=255.255.255.0,255.255.255.0 OSDAdapter0Gateways=192.168.0.1,192.168.100.1 OSDAdapter0EnableWINS=TRUE OSDAdapter0WINSServerList=192.168.0.1,192.168.100.1 OSDAdapter0TCPIPNetBiosOptions=0 OSDAdapter0MacAddress=00:0C:29:67:A3:6B OSDAdapter0GatewayCostMetrics=Automatic OSDAdapter0EnableTCPIPFiltering=TRUE OSDAdapter0EnableLMHosts=TRUE OSDAdapter0EnableFullDNSRegistration=TRUE OSDAdapter0EnableDNSRegistration=TRUE OSDAdapter0DNSSuffix=WoodGroveBank.com`

OSDAnswerFilePath

Specifies the path to the answer file to be used during OEM deployments.

> **Note**
>
> This property is dynamically set by the MDT scripts and is not configured in CustomSettings.ini or the MDT DB. Treat this property as read only.

Expand table

Component	Configured By	Scenario	Property Is Applicable
BootStrap.ini	❌	LTI (Stand-alone MDT)	✅
CustomSettings.ini	❌		
MDT DB	❌	ZTI (Configuration Manager)	❌

Expand table

Value	Description
file_path	Specifies the path to the answer file to be used during OEM deployments

Expand table

Example
None

OSDBitLockerCreateRecoveryPassword

A Boolean value that indicates whether the process creates a recovery key for BitLocker. The key is used for recovering data encrypted on a BitLocker volume. This key is cryptographically equivalent to a startup key. If available, the recovery key decrypts the VMK, which, in turn, decrypts the FVEK.

> **Note**
>
> This property value must be specified in uppercase letters so that the deployment scripts can properly read it.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
AD	A recovery key is created.
Not specified	A recovery key is not created.

Expand table

Example
`[Settings] Priority=Default [Default] BDEInstallSuppress=NO BDEDriveLetter=S: BDEDriveSize=2000 OSDBitLockerMode=TPMKey OSDBitLockerCreateRecoveryPassword=AD OSDBitLockerStartupKeyDrive=C:`

OSDBitLockerMode

The type of BitLocker installation to be performed. Protect the target computer using one of the following methods:

- A TPM microcontroller
- A TPM and an external startup key (using a key that is typically stored on a UFD)
- A TPM and PIN
- An external startup key

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		

Component	Configured By		Scenario	Property Is Applicable
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
TPM	Protect the computer with TPM only. The TPM is a microcontroller that stores keys, passwords, and digital certificates. The microcontroller is typically an integral part of the computer motherboard.
TPMKey	Protect the computer with TPM and a startup key. Use this option to create a startup key and to save it on a UFD. The startup key must be present in the port each time the computer starts.
TPMPin	Protect the computer with TPM and a pin. Use this option in conjunction with the **BDEPin** property. Note: This value is not valid when using ZTI.
Key	Protect the computer with an external key (the recovery key) that can be stored in a folder, in AD DS, or printed.

Expand table

Example
`[Settings] Priority=Default [Default] BDEInstallSuppress=NO BDEDriveLetter=S: BDEDriveSize=2000 OSDBitLockerMode=TPM OSDBitLockerCreateRecoveryPassword=AD`

OSDBitLockerRecoveryPassword

Instead of generating a random recovery password, the **Enable BitLocker** task sequence action uses the specified value as the recovery password. The value must be a valid numerical BitLocker recovery password.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		

Component	Configured By		Scenario	Property Is Applicable
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
password	A valid 48-digit password

Expand table

Example
`[Settings] Priority=Default [Default] BDEInstallSuppress=NO BDEDriveLetter=S: BDEDriveSize=2000 OSDBitLockerMode=TPMKey OSDBitLockerCreateRecoveryPassword=AD OSDBitLockerRecoveryPassword=621280128854709621167486709731081433315062587367 OSDBitLockerStartupKeyDrive=C:`

OSDBitLockerStartupKey

Instead of generating a random startup key for the key management option **Startup Key on USB only**, the **Enable BitLocker** task sequence action uses the value as the startup key. The value must be a valid, Base64-encoded BitLocker startup key.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
startupkey	Base64-encoded BitLocker startup key

Expand table

Example
`[Settings] Priority=Default [Default] BDEInstallSuppress=NO BDEDriveLetter=S: BDEDriveSize=2000 BDEInstall=KEY OSDBitLockerCreateRecoveryPassword=AD OSDBitLockerStartupKey=8F4922B8-2D8D-479E-B776-12629A361049`

OSDBitLockerStartupKeyDrive

The location for storing the BitLocker recovery key and startup key.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
location	The storage location for the recovery key and startup key (either local to the target computer or to a UNC that points to a shared network folder)

Expand table

Example
`[Settings] Priority=Default [Default] BDEInstallSuppress=NO BDEDriveLetter=S: BDEDriveSize=2000 OSDBitLockerMode=TPMKey OSDBitLocker CreateRecoveryPassword=AD OSDBitLockerStartupKeyDrive=C:`

OSDBitLockerTargetDrive

Specifies the drive to be encrypted. The default drive is the drive that contains the operating system.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
drive	The drive that is to be encrypted

Expand table

Example
`[Settings] Priority=Default [Default] BDEInstallSuppress=NO BDEDriveLetter=S: BDEDriveSize=2000 BDERecoveryPassword=TRUE OSDBitLockerMode=TPMKey OSDBitLockerCreateRecoveryPassword=AD OSDBitLockerTargetDrive=C:`

OSDBitLockerWaitForEncryption

Specifies that the deployment process should not proceed until BitLocker has completed the encryption process for all specified drives. Specifying TRUE could dramatically increase the time required to complete the deployment process.

> **⊗ Caution**
>
> This property value must be specified in uppercase letters so that the deployment scripts can properly read it.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
TRUE	Specifies that the deployment process should wait for drive encryption to finish
FALSE	Specifies that the deployment process should not wait for drive encryption to finish

Expand table

Example
`[Settings] Priority=Default [Default] BDEInstallSuppress=NO BDEDriveLetter=S: BDEDriveSize=2000 OSDBitLockerMode=TPMKey OSDBitLockerStartupKeyDrive=C: OSDBitLockerCreateRecoveryPassword=AD OSDBitLockerWaitForEncryption=TRUE`

OSDComputerName

The new computer name to assign to the target computer.

> **Note**
>
> This property can also be set within a task sequence using a customized **Set Task Sequence Variable** task sequence step.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
computer_name	The new computer name to assign to the target computer

Expand table

Example
`[Default] OSDComputerName=%_SMSTSMachineName%`

OSDDiskAlign

This property is used to pass a value to the **align** parameter of the **create partition primary** command in the **DiskPart** command. The **align** parameter is typically used with hardware RAID Logical Unit Number (LUN) arrays to improve performance when the logical units (LUs) are not cylinder aligned. The **align** parameter aligns a primary partition that is not cylinder aligned at the beginning of a disk and rounds the offset to the closest alignment boundary. For more information on the **align** parameter, see Create partition primary.

> **Note**
>
> This property can be used in conjunction with the **OSDDiskOffset** property to set the **offset** parameter for the **create partition primary** command in the **DiskPart** command. For more information, see the OSDDiskOffset property.

Expand table

Component	Configured By	Scenario	Property Is Applicable
BootStrap.ini	❌	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅		
MDT DB	✅	ZTI (Configuration Manager)	❌

Expand table

Value	Description
alignment_value	Specifies the number of kilobytes (KB) from the beginning of the disk to the closest alignment boundary.

Expand table

Example
`[Settings] Priority=Default [Default] OSDDiskAlign=1024 OSDDiskOffset=2048`

OSDDiskIndex

Specifies the disk index that will be configured.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
disk_index	Specifies the disk index that will be configured (The default value is **0**.)

Expand table

Example
`[Settings] Priority=Default [Default] OSDDiskIndex=0`

OSDDiskOffset

This property is used to pass a value to the **offset** parameter of the **create partition primary** command in the **DiskPart** command. For more information on the **offset** parameter, see Create partition primary.

This property can be used in conjunction with the **OSDDiskAlign** property to set the **align** parameter for the **create partition primary** command in the **DiskPart** command. For more information, see the OSDDiskAlign property.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
offset_value	Specifies the byte offset at which to create the partition. For master boot record (MBR) disks, the offset rounds to the closest cylinder boundary.

Expand table

Example
`[Settings] Priority=Default [Default] OSDDiskAlign=1024 OSDDiskOffset=2048`

OSDDiskPartBiosCompatibilityMode

This property specifies whether to disable cache alignment optimizations when partitioning the hard disk for compatibility with certain types of BIOS.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
TRUE	Enables cache alignment optimizations when partitioning the hard disk for compatibility with certain types of BIOS
FALSE	Disables cache alignment optimizations when partitioning the hard disk for compatibility with certain types of BIOS (This is the default value.)

Expand table

Example
`[Settings] Priority=Default [Default] OSDDiskPartBiosCompatibilityMode=TRUE`

OSDImageCreator

Specifies the name of the installation account that will be used during OEM deployments.

> **Note**
>
> This property is dynamically set by the MDT scripts and is not configured in CustomSettings.ini or the MDT DB. Treat this property as read only.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	❌	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
image_creator	Specifies the name of the installation account that will be used during OEM deployments

Expand table

Example
None

OSDImageIndex

Specifies the index of the image in the .wim file. This property is referenced during OEM deployments.

> **Note**
>
> This property is dynamically set by the MDT scripts and is not configured in CustomSettings.ini or the MDT DB. Treat this property as read only.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	❌	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
index	Specifies the index of the image in the WIM file

Expand table

Example
None

OSDImagePackageID

Specifies the package ID for the image to install during OEM deployments.

> **Note**
>
> This property is dynamically set by the MDT scripts and is not configured in CustomSettings.ini or the MDT DB. Treat this property as read only.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	❌	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
package_ID	Specifies the package ID for the image to install during OEM deployments

Expand table

Example
None

OSDInstallEditionIndex

Specifies the index of the image in the WIM file. This property is referenced during OEM deployments.

> **Note**
>
> This property is dynamically set by the MDT scripts and is not configured in CustomSettings.ini or the MDT DB. Treat this property as read only.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	❌	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
index	Specifies the index of the image in the WIM file

Expand table

Example
None

OSDInstallType

Specifies the installation type used for OEM deployments. The default is **Sysprep**.

> **Note**

This property is dynamically set by the MDT scripts and is not configured in CustomSettings.ini or the MDT DB. Treat this property as read only.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	❌	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
install_type	Specifies the installation type used for OEM deployments

Expand table

Example
None

OSDisk

Specifies the drive used to install the operating system during OEM deployments. The default value is **C:**.

> ⓘ **Note**
>
> This property is dynamically set by the MDT scripts and is not configured in CustomSettings.ini or the MDT DB. Treat this property as read only.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	❌	|		

Component	Configured By		Scenario	Property Is Applicable
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
disk	Specifies the drive used to install the operating system during OEM deployments

Expand table

Example
None

OSDPartitions

Specifies the number of defined partitions configurations. The maximum number of partitions that can be configured is two. The default is **None**.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
partitions	Specifies the number of defined partitions configurations

Expand table

Example
`[Settings] Priority=Default [Default] OSDPartitions=1 OSDPartitions0Bootable=TRUE OSDPartitions0FileSystem=NTFS OSDPartitions0QuickFormat=TRUE OSDPartitions0Size=60`

Example
`OSDPartitions0SizeUnits=GB OSDPartitions0Type=Primary OSDPartitions0VolumeName=OSDisk OSDPartitions0VolumeLetterVariable=NewDrive1`

OSDPartitionsxBootable

The partition at the specified index should be set bootable. The default first partition is set bootable.

> **Note**
>
> The *x* in this properties name is a placeholder for a zero-based array that contains partition configurations.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
TRUE	The partition should be set to bootable.
FALSE	Do not set the partition to bootable.

Expand table

Example
`[Settings] Priority=Default [Default] OSDPartitions0Bootable=TRUE`

OSDPartitionsxFileSystem

The type of file system for the partition at the specified index. Valid values are **NTFS** or **FAT32**.

> **Note**
>
> The*x* in this properties name is a placeholder for a zero-based array that contains partition configurations.

> **Caution**
>
> This property value must be specified in uppercase letters so that the deployment scripts can properly read it.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
file_system	The type of file system for the partition

Expand table

Example
`[Settings] Priority=Default [Default] OSDPartitions0FileSystem=NTFS`

OSDPartitionsxQuickFormat

The partition at the specified index should be quick formatted. The default is **TRUE**.

> **Note**
>
> The*x* in this properties name is a placeholder for a zero-based array that contains partition configurations.

> **⊗ Caution**
>
> This property value must be specified in uppercase letters so that the deployment scripts can properly read it.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
TRUE	Quick-format the partition.
FALSE	Do not quick-format the partition.

Expand table

Example
`[Settings] Priority=Default [Default] OSDPartitions0QuickFormat=TRUE`

OSDPartitionsxSize

The size of the partition at the specified index.

> **ⓘ Note**
>
> The*x* in this properties name is a placeholder for a zero-based array that contains partition configurations.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
Size	Partition size

Expand table

Example
`[Settings] Priority=Default [Default] OSDPartitions0Size=60 OSDPartitions0SizeUnits=GB`

OSDPartitionsxSizeUnits

The units of measure used when specifying the size of the partition. Valid values are **MB**, **GB**, or %. The default value is **MB**.

> **Note**
>
> The*x* in this properties name is a placeholder for a zero-based array that contains partition configurations.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
size_units	The units of measure used when specifying the size of the partition

Expand table

Example
`[Settings] Priority=Default [Default] OSDPartitions0Size=60 OSDPartitions0SizeUnits=GB`

OSDPartitionsxType

The type of partition to be created at the specified index.

> **Note**
>
> The*x* in this properties name is a placeholder for a zero-based array that contains partition configurations.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
Primary	Create a primary partition. This is the default value.
Logical	Create a logical partition.
Extended	Create an extended partition.

Expand table

Example
`[Settings] Priority=Default [Default] OSDPartitions0Type=Primary`

OSDPartitionsxVolumeLetterVariable

The property that receives the drive letter that is assigned to the partition being managed.

> **Note**
>
> The*x* in this properties name is a placeholder for a zero-based array that contains partition configurations.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
volume_letter_variable	The name of the variable that will be assigned the drive letter of the partition being managed

Expand table

Example
`[Settings] Priority=Default [Default] OSDPartitions0VolumeLetterVariable=NewDrive1`

OSDPartitionsxVolumeName

The volume name that will be assigned to the partition at the specified index.

> **Note**
>
> The*x* in this properties name is a placeholder for a zero-based array that contains partition configurations.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
volume_name	The volume name that will be assigned to the partition

Expand table

Example
`[Settings] Priority=Default [Default] OSDPartitions0VolumeName=OSDisk`

OSDStateStorePath

LTI and ZTI use this property to set the path where the user state migration data will be stored, which can be a UNC path, a local path, or a relative path.

> ⓘ **Note**
>
> The **OSDStateStorePath** property takes precedence over the StatePath or UserDataLocation property when those properties are also specified.

In a Replace Computer deployment scenario in ZTI, the Restore User State task sequence step is skipped if the **OSDStateStorePath** property is set to a valid local or UNC path. The workaround is to set the USMTLocal property to TRUE. Doing so forces ZTI UserState.wsf to recognize the path in the OSDStateStorePath property. This is caused by the **Request State Store** task sequence step being skipped and the previous value in the **OSDStateStorePath** property being retained.

In a Replace Computer deployment scenario in ZTI, where user state migration data and the entire computer are being backed up, the Backup.wim file is stored in the folder specified in the **OSDStateStorePath** property. This may be caused by specifying the wrong value for the ComputerBackupLocation property.

For example, the following CustomSettings.ini file will cause the Backup.wim file to be stored in the same folder specified in the **OSDStateStorePath** property:

```ini
USMTLocal=True
OSDStateStorePath=\\fs1\Share\Replace

ComputerBackupLocation=NETWORK
BackupShare=\\fs1\Share\ComputerBackup
BackupDir=Client01
```

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
Path	The path where the user state migration data will be stored, which can be a UNC path, a local path, or a relative path

Expand table

Example
`[Settings] Priority=Default [Default] USMTLocal=True OSDStateStorePath=\\fs1\Share\Replace ComputerBackupLocation=\\fs1\Share\ComputerBackup\Client01`

OSDTargetSystemDrive

Specifies the drive where the operating system will be installed during OEM deployments.

> ⓘ **Note**

This property is dynamically set by the MDT scripts and is not configured in CustomSettings.ini or the MDT DB. Treat this property as read-only.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	❌	|		
MDT DB	❌	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
system_drive	Specifies the drive where the operating system will be installed during OEM deployments

Expand table

Example
None

OSDTargetSystemRoot

Specifies the install path where the operating system will be installed during OEM deployments.

> **Note**
>
> This property is dynamically set by the MDT scripts and is not configured in CustomSettings.ini or the MDT DB. Treat this property as read only.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	❌	|		

Component	Configured By		Scenario	Property Is Applicable
MDT DB	❌	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
system_root	Specifies the install path where the operating system will be installed during OEM deployments

Expand table

Example
None

OSFeatures

A comma-delimited list of server feature IDs that will be installed on the target computer.

> **Note**
>
> Not all features listed in the ServerManager.xml file are compatible with all server operating systems.

> **Caution**
>
> This property value must be specified in uppercase letters so that the deployment scripts can properly read it.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
ID1,ID2	The server features that are to be installed on the target computer. Valid values are located in the *program_files*\Microsoft Deployment Toolkit\Bin\ServerManager.xml file on the MDT server.

Expand table

Example
`[Settings] Priority=Default [Default] OSFeatures=CMAK,MSMQ-Multicasting,RSAT`

OSInstall

Indicates whether the target computer is authorized to have the target operating system installed. If the **OSInstall** property is not listed, the default is to allow deployment of operating systems to any target computer.

> **⊗ Caution**
>
> This property value must be specified in uppercase letters so that the deployment scripts can properly read it.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
YES	Deployment of an operating system to the target computer is authorized. This is the default value.
NO	Deployment of an operating system to the target computer is not authorized.

Expand table

Example
`[Settings] Priority=Default [Default] OSInstall=YES`

OSRoles

A comma-delimited list of server role IDs that will be installed on the target computer.

> **Note**
>
> Not all roles are compatible with all server operating systems.

> **Caution**
>
> This property value must be specified in uppercase letters so that the deployment scripts can properly read it.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
ID1,ID2	The server role that is to be installed on the target computer.

See "C:\Program Files\Microsoft Deployment Toolkit\Bin\ServerManager.xml" for valid ID values.

Expand table

Example
`[Settings] Priority=Default [Default] OSRoles=ADDS`

OSRoleServices

A comma-delimited list of server role service IDs that will be installed on the target computer.

> **Note**
>
> Not all server role service IDs are compatible with all server operating systems.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
ID	The server role service that will be installed on the target computer. The valid value is: - **ADDS-Domain-Controller**

Expand table

Example
`[Settings] Priority=Default [Default] OSRoleServices=ADDS-Domain-Controller`

OSSKU

The edition of the currently running operating system. The operating system edition is determined by using the **OperatingSystemSKU** property of the **Win32_OperatingSystem WMI** class. For a list of the editions the **OperatingSystemSKU** property returns, see the section, "OperatingSystemSKU," at Win32_OperatingSystem class.

> **Note**

This property is dynamically set by the MDT scripts and is not configured in CustomSettings.ini or the MDT DB. Treat this property as read only.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	❌	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
edition	The operating system edition. For example, "BUSINESS" for a Business edition of an operating system or "ENTERPRISE" for an Enterprise edition of an operating system.

Expand table

Example
None

OSVersion

The version of the currently running operating system. This property should only be used to detect if the currently running operating system is Windows PE. Use the OSVersionNumber property to detect other operating systems.

> **Note**
>
> This property is dynamically set by the MDT scripts and is not configured in CustomSettings.ini or the MDT DB. Treat this property as read only.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅

Component	Configured By		Scenario	Property Is Applicable
CustomSettings.ini	❌	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
WinPE	Windows PE
2008R2	Windows Server 2008 R2
Win7Client	Windows 7
Other	Operating systems other than those listed, including Windows 8 and Windows Server 2012

Expand table

Example
None

OSVersionNumber

The operating system major and minor version number. This property is referenced during OEM deployments.

> **Note**
>
> This property is dynamically set by the MDT scripts and is not configured in CustomSettings.ini or the MDT DB. Treat this property as read only.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	❌	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
version_number	The operating system major and minor version number

Expand table

Example
None

OverrideProductKey

The Multiple Activation Key (MAK) string to be applied after the target operating is deployed to the target computer. The value specified in this property is used by the ZTILicensing.wsf script during the State Restore Phase to apply the MAK to the target operating system. The script also configures the volume licensing image to use MAK activation instead of Key Management Service (KMS). The operating system needs to be activated with Microsoft after the MAK is applied. This is used when the target computer is unable to access a server that is running KMS.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
MAK	The MAK string to be provided to the target operating system

Expand table

Example
`[Settings] Priority=Default [Default] ProductKey=AAAAA-BBBBB-CCCCC-DDDDD-EEEEE-FFFFF OverrideProductKey=AAAAA-BBBBB-CCCCC-DDDDD-EEEEE-FFFFF`

PackageGroup

A list of text values that associates operating system packages with each other (typically based on the type of operating system package). An operating system package can be associated with one or more package groups. The **PackageGroup** property allows the operating system packages within one or more groups to be deployed to a target computer.

The text values in the list can be any non-blank value. The **PackageGroup** property value has a numeric suffix (for example, **PackageGroup001** or **PackageGroup002**). After it is defined, a package group is associated with a computer. A computer can be associated with more than one package group.

> **Note**
>
> Operating system packages are created on the OS Packages node in the Deployment Workbench.

> **Note**
>
> The **PackageGroup** property can be specified in the format *PackageGroup1=Updates* or *PackageGroup001=Updates*.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	❌	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
package_group_name	Name of the package group to be deployed to the target computer

Expand table

Example
`[Settings] Priority=Default [Default] PackageGroup001=Updates`

Packages

The list of Configuration Manager packages to be deployed to the target computer. The **Packages** property has a numeric suffix (for example, Packages001 or Packages002).

> **Note**
>
> The **PackageGroup** property can be specified in the format *PackageGroup1=Updates* or *PackageGroup001=Updates*.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	❌
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
package_id:program_name	Name of the package to be deployed to the target computer

Expand table

Example
`[Settings] Priority=Default [Default] Packages001=NYC00010:Install Packages002=NYC00011:Install`

PackageSelectionProfile

Profile name used during package installation.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
profile_name	Profile name used during package installation

Expand table

Example
`[Settings] Priority=Default [Default] PackageSelectionProfile=CoreApplications`

Parameters

The parameters to be passed to a database query that returns property values from columns in the table specified in the **Table** property. The table is located in the database specified in the **Database** property on the computer specified in the **SQLServer** property. The instance of SQL Server on the computer is specified in the **Instance** property.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	✅	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
parameter1, parameter2	The list of parameters to pass to the database query

Expand table

Example
`[Settings] Priority=Computers, Default [Default] OSInstall=YES [Computers] SQLServer=NYC-SQL-01 SQLShare=SQL$ Database=MDTDB Instance=SQLEnterprise2005 Table=Computers Parameters=SerialNumber, AssetTag ParameterCondition=OR`

ParameterCondition

Indicator of whether a Boolean AND or OR operation is performed on the properties listed in the **Parameters** property.

> **⊗ Caution**
>
> This property value must be specified in uppercase letters so that the deployment scripts can properly read it.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	✅	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
AND	A Boolean AND operation is performed on the properties listed in the **Parameters** property. Only results that match all properties specified in the **Parameters** property are returned. This is the default value.
OR	A Boolean OR operation is performed on the properties listed in the **Parameters** property. Results that match any property specified in the **Parameters** property are returned.

Expand table

Example
`[Settings] Priority=Computers, Default [Default] OSInstall=YES [Computers] SQLServer=NYC-SQL-01 SQLShare=SQL$ Database=MDTDB Instance=SQLEnterprise2005 Table=Computers`

Example
`Parameters=SerialNumber, AssetTag ParameterCondition=OR`

ParentDomainDNSName

Specifies the DNS domain name of an existing directory service domain when installing a child domain.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
name	Specifies the DNS domain name of an existing directory service domain when installing a child domain

Expand table

Example
`[Settings] Priority=Default [Default] ParentDomainDNSName=WoodGroveBank.com`

Password

Specifies the password for the user name (account credentials) to use for promoting the member server to a domain controller.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		

Component	Configured By		Scenario	Property Is Applicable
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
password	Specifies the password for the user name (account credentials) to use for promoting the member server to a domain controller

Expand table

Example
`[Settings] Priority=Default [Default] Password=<complex_password>`

Phase

The current phase of the deployment process. The Task Sequencer uses these phases to determine which tasks must be completed.

> **Note**
>
> This property is dynamically set by the MDT scripts and is not configured in CustomSettings.ini or the MDT DB. Treat this property as read only.

> **Caution**
>
> This property value must be specified in uppercase letters so that the deployment scripts can properly read it.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	❌	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
VALIDATION	Identifies that the target computer is capable of running the scripts necessary to complete the deployment process.
STATECAPTURE	Saves any user state migration data before deploying the new target operating system.
PREINSTALL	Completes any tasks that need to be done (such as creating new partitions) before the target operating system is deployed.
INSTALL	Installs the target operating system on the target computer.
POSTINSTALL	Completes any tasks that need to be done before restoring the user state migration data. These tasks customize the target operating system before starting the target computer the first time (such as installing updates or adding drivers).
STATERESTORE	Restores the user state migration data saved during the State Capture Phase.

Expand table

Example
None

Port

The number of the port that should be used when connecting to the SQL Server database instance that is used for querying property values from columns in the table specified in the **Table** property. The database resides on the computer specified in the **SQLServer** property. The instance of SQL Server on the computer is specified in the **Instance** property. The port used during connection is specified in the **Port** property.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	✅	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
port	The number of the port used when connecting to SQL Server

Expand table

Example
`[Settings] Priority=Computers, Default [Default] OSInstall=YES [Computers] SQLServer=NYC-SQL-01 Database=MDTDB Instance=MDT2010 Port=1433 Table=Computers Parameters=SerialNumber, AssetTag ParameterCondition=OR`

PowerUsers

A list of user accounts and domain groups to be added to the local Power Users group on the target computer. The **PowerUsers** property is a list of text values that can be any non-blank value. The **PowerUsers** property has a numeric suffix (for example, **PowerUsers1** or **PowerUsers2**).

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
name	Name of the user or group to be added to the local Power Users group

Expand table

Example
`[Settings] Priority=Default [Default] Administrators001=WOODGROVEBANK\NYC Help Desk Staff PowerUsers001=WOODGROVEBANK\User01 PowerUsers002=WOODGROVEBANK\User02`

PrepareWinRE

This property specifies if the LiteTouchPE.wim file, which includes Windows RE and optionally DaRT, is applied to the system drive as the recovery partition. This allows the target computer to use the LiteTouchPE.wim image to perform recovery tasks. DaRT may optionally be included in the image, which makes DaRT recovery features available on the target computer.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	❌	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
YES	The LiteTouchPE.wim file, which includes Windows RE and optionally DaRT, is applied to the system drive as the recovery partition.
any other value	The LiteTouchPE.wim file, which includes Windows RE and optionally DaRT, is not applied to the system drive as the recovery partition. This is the default value.

Expand table

Example
`[Settings] Priority=Default [Default] PrepareWinRE=YES`

Priority

The reserved property that determines the sequence for finding configuration values. The **Priority** reserved property lists each section to be searched and the order in which the sections are searched. When a property value is found, the ZTIGather.wsf script quits searching for the property, and the remaining sections are not scanned for that property.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	✅	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
section1, section2	The sections to be searched in the order they are to be searched

Expand table

Example
`[Settings] Priority=MACAddress, Default [Default] UserDataLocation=NONE CustomProperty=TRUE [00:0F:20:35:DE:AC] OSDNEWMACHINENAME=HPD530-1 [00:03:FF:FE:FF:FF] OSDNEWMACHINENAME=BVMXP`

ProcessorSpeed

The speed of the processor installed on the target computer in MHz. For example, the value **1995** indicates the processor on the target computer is running at 1,995 MHz or 2 gigahertz.

> **Note**
>
> This property is dynamically set by the MDT scripts and is not configured in CustomSettings.ini or the MDT DB. Treat this property as read only.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	❌	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
processor_speed	The speed of the processor on the target computer in megahertz

Expand table

Example
None

Product

The product name of the target computer. With some computer vendors, the make and model might not be sufficiently unique to identify the characteristics of a particular configuration (for example, hyperthreaded or non-hyperthreaded chipsets). The **Product** property can help to differentiate.

The format for **Product** is undefined. Use this property to create a subsection that contains settings targeted to a specific product name for a specific computer model number for a specific computer manufacturer (most commonly in conjunction with the **Make** and **Model** properties).

> **Note**
>
> This property is dynamically set by the MDT scripts and is not configured in CustomSettings.ini or the MDT DB. Treat this property as read only.

Expand table

Component	Configured By	Scenario	Property Is Applicable
BootStrap.ini	❌	LTI (Stand-alone MDT)	✅
CustomSettings.ini	❌		
MDT DB	❌	ZTI (Configuration Manager)	✅

Expand table

Value	Description
product	The product name of the target computer

Expand table

Example
None

ProductKey

The product key string to be configured for the target computer. Before the target operating system is deployed, the product key specified is automatically inserted into the appropriate location in Unattend.xml.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
product_key	The product key to be assigned to the target computer

Expand table

Example
`[Settings] Priority=Default [Default] ProductKey=AAAAA-BBBBB-CCCCC-DDDDD-EEEEE-FFFFF`

Properties

A reserved property that defines any custom, user-defined properties. These user-defined properties are located by the ZTIGather.wsf script in the CustomSettings.ini file, BootStrap.ini file, or the MDT DB. These properties are additions to the predefined properties in MDT.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	✅	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
custom_property1,custom_property2	Custom, user-defined properties to be resolved

Expand table

Example
`[Settings] Priority=MACAddress, Default Properties=CustomProperty, ApplicationInstall [Default] OSInstall=YES ScanStateArgs=/v:5 /o /c LoadStateArgs=/v:5 /c /lac UserDataLocation=NONE CustomProperty=TRUE [00:0F:20:35:DE:AC] OSDNEWMACHINENAME=HPD530-1 ApplicationInstall=Custom [00:03:FF:FE:FF:FF] OSDNEWMACHINENAME=BVMXP ApplicationInstall=Minimum`

ReplicaDomainDNSName

Specifies the DNS domain name of the domain to replicate.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
name	Specifies the DNS domain name of the domain to replicate

Expand table

Example
`[Settings] Priority=Default [Default] ReplicaDomainDNSName=WoodGroveBank.com`

ReplicaOrNewDomain

Specifies whether to install a new domain controller as the first domain controller in a new directory service domain or to install it as a replica directory service domain controller.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
Replica	Installs the new domain controller as a replica directory service domain controller.
Domain	Installs the new domain controller as the first domain controller in a new directory service domain. You must specify the **TreeOrChild** entry with a valid value.

Expand table

Example
`[Settings] Priority=Default [Default] ReplicaOrNewDomain=Domain`

ReplicationSourceDC

Indicates the full DNS name of the domain controller from which you replicate the domain information.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
name	Indicates the full DNS name of the domain controller from which you replicate the domain information

Expand table

Example
`[Settings] Priority=Default [Default] ReplicationSourceDC=dc01.WoodGroveBank.com`

ResourceDrive

The drive letter mapped to the **ResourceRoot** property for the ZTIDrivers.wsf and ZTIPatches.wsf scripts to use to install drivers and patches to the target computer.

> **Note**
>
> This property is dynamically set by the MDT scripts and is not configured in CustomSettings.ini or the MDT DB. Treat this property as read only.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	❌	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
drive_letter	The letter designation for the logical drive that contains the resources

Expand table

Example
None

ResourceRoot

The value of this property is used by the ZTIDrivers.wsf and ZTIPatches.wsf scripts to install drivers and patches to the target computer.

> **Note**
>
> For LTI, the scripts automatically set the **ResourceRoot** property to be the same as the **DeployRoot** property. For ZTI, the values in the **DeployRoot** and **ResourceRoot** properties can be unique.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	✅	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
UNC_path	The UNC path to the shared folder that contains the resources

Expand table

Example
`[Settings] Priority=Default [Default] DeployRoot=\\NYC-AM-FIL-01\Distribution$ ResourceDrive=R: ResourceRoot=\\NYC-AM-FIL-01\Resource$ UserDataLocation=NONE`

Role

The purpose of a computer based on the tasks performed by the user on the target computer. The **Role** property lists text values that can be any non-blank value. The **Role** property value has a numeric suffix (for example, **Role1** or **Role2**). When defined, a role is associated with a computer. A computer can perform more than one role.

Typically, the value for the **Role** property is set by performing a database query in the MDT DB. The Deployment Workbench can assist in creating the role and property settings associated with the role, and then the Deployment Workbench can configure CustomSettings.ini to perform the database query for the **Role** property and the property settings associated with the role.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	✅	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
Role	The roles to be assigned to an individual computer or a group of computers

Expand table

Example 1

```
[Settings] Priority=RoleSettings, Default [Default] SkipCapture=NO UserDataLocation=AUTO DeployRoot=\\W2K3-SP1\Distribution$ OSInstall=YES ScanStateArgs=/v:15 /o /c LoadStateArgs=/v:7 /c [RoleSettings] SQLServer=w2k3-sp1 Instance=MDT2010 Database=MDTDB Netlib=DBNMPNTW SQLShare=SQL_Share Table=RoleSettings Parameters=Role
```

Expand table

Example 2

```
[Settings] Priority=RoleSettings, Default [Default] SkipCapture=NO UserDataLocation=AUTO DeployRoot=\\W2K3-SP1\Distribution$ OSInstall=YES Role1=Teller Role2=Woodgrove User
```

Example 2
`[RoleSettings] SQLServer=w2k3-sp1 Instance=MDT2010 Database=MDTDB Netlib=DBNMPNTW SQLShare=SQL_Share Table=RoleSettings Parameters=Role`

SafeModeAdminPassword

Supplies the password for the administrator account when starting the computer in Safe mode or a variant of Safe mode, such as Directory Services Restore mode.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
password	Supplies the password for the administrator account when starting the computer in Safe mode or a variant of Safe mode, such as Directory Services Restore mode

Expand table

Example
`[Settings] Priority=Default [Default] SafeModeAdminPassword=<complex_password>`

ScanStateArgs

Arguments passed to the **USMT Scanstate** process. The scripts call Scanstate.exe, and then insert the appropriate logging, progress, and state store parameters. If this value is not included in the settings file, the user state backup process is skipped.

> **Note**
>
> Use the **USMTMigFiles** property to specify the .xml files to be used by Scanstate.exe instead of using the /I parameter in the **ScanStateArgs** property. This

prevents the ZTIUserState.wsf script from potentially duplicating the same list of .xml files.

> **Note**
>
> Do not add any of the following command line arguments when configuring this property: **/hardlink**, **/nocompress**, **/encrypt**, **/key**, or **/keyfile**. The MDT scripts will add these command-line arguments if applicable to the current deployment scenario.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
arguments	The command-line arguments passed to Scanstate.exe. The default arguments specified by the Deployment Workbench are as follows: - **/v**. Enables verbose output in the Scanstate log. The default is **0**. Specify any number from 0 to 15. The value **5** enables verbose and status output. - **/o**. Overwrites any existing data in the store. If not specified, Scanstate will fail if the store already contains data. This option cannot be specified more than once in a Command Prompt window. - **/c**. When specified, Scanstate will continue to run even if there are nonfatal errors. Without the **/c** option, Scanstate exits on the first error. For more information about these and other arguments, see the USMT Help files.

Expand table

Example
`[Settings] Priority=Default [Default] ScanStateArgs=/v:5 /o /c LoadStateArgs=/v:5 /c /lac DeployRoot=\\NYC-AM-FIL-01\Distribution$ ResourceRoot=\\NYC-AM-FIL-01\Resource$ UDShare=\\NYC-AM-FIL-01\MigData$ UDDir=%OSDComputerName%`

SerialNumber

The serial number of the target computer. The format for serial numbers is undefined. Use this property to create a subsection that contains settings targeted to a specific computer.

> **Note**
>
> This property is dynamically set by the MDT scripts and is not configured in CustomSettings.ini or the MDT DB. Treat this property as read only.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	❌	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
serial_number	The format of the serial number is undefined and is determined by the serial number standard of each computer manufacturer.

Expand table

Example
None

SiteName

Specifies the name of an existing site where you can place the new domain controller.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
name	Specifies the name of an existing site where you can place the new domain controller

Expand table

Example
`[Settings] Priority=Default [Default] SiteName=FirstSite`

SkipAdminAccounts

Indicates whether the **Local Administrators** wizard page is skipped.

> ⓘ **Note**
>
> This default value for this property is **YES**, which means that the **Local Administrators** wizard page will be skipped by default. To display this wizard page, you must specifically set the value of this property to **NO** in CustomSettings.ini or in the MDT DB.

For other properties that must be configured when this property is set to **YES**, see Providing Properties for Skipped Deployment Wizard Pages.

> ⊗ **Caution**
>
> This property value must be specified in uppercase letters so that the deployment scripts can properly read it.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
YES	Wizard page is not displayed, and the information on that page is not collected. This is the default value.
NO	Wizard page is displayed, and the information on that page is collected.

Expand table

Example
`[Settings] Priority=Default [Default] SkipWizard=NO SkipCapture=NO SkipAdminAccounts=NO SkipAdminPassword=NO SkipApplications=NO SkipComputerBackup=NO SkipDomainMembership=NO SkipUserData=NO SkipPackageDisplay=NO SkipLocaleSelection=NO SkipProductKey=YES`

SkipAdminPassword

Indicates whether the **Administrator Password** wizard page is skipped.

For other properties that must be configured when this property is set to **YES**, see Providing Properties for Skipped Deployment Wizard Pages.

> ⊗ **Caution**
>
> This property value must be specified in uppercase letters so that the deployment scripts can properly read it.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		

Component	Configured By		Scenario	Property Is Applicable
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
YES	Wizard page is not displayed, and the information on that page is not collected.
NO	Wizard page is displayed, and the information on that page is collected. This is the default value.

Expand table

Example
`[Settings] Priority=Default [Default] SkipWizard=NO SkipCapture=NO SkipAdminPassword=YES SkipApplications=NO SkipComputerBackup=NO SkipDomainMembership=NO SkipUserData=NO SkipPackageDisplay=NO SkipLocaleSelection=NO SkipProductKey=YES`

SkipApplications

Indicates whether the **Select one or more applications to install** wizard page is skipped.

For other properties that must be configured when this property is set to **YES**, see Providing Properties for Skipped Deployment Wizard Pages.

> ⊗ **Caution**
>
> This property value must be specified in uppercase letters so that the deployment scripts can properly read it.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
YES	Wizard page is not displayed, and the information on that page is not collected.
NO	Wizard page is displayed, and the information on that page is collected. This is the default value.

Expand table

Example
`[Settings] Priority=Default [Default] SkipWizard=NO SkipCapture=NO SkipAdminPassword=NO SkipApplications=YES SkipComputerBackup=NO SkipDomainMembership=NO SkipUserData=NO SkipPackageDisplay=NO SkipLocaleSelection=NO SkipProductKey=YES`

SkipBDDWelcome

Indicates whether the **Welcome to Windows Deployment** wizard page is skipped.

For other properties that must be configured when this property is set to **YES**, see Providing Properties for Skipped Deployment Wizard Pages.

> **Note**
>
> For this property to function properly it must be configured in both CustomSettings.ini and BootStrap.ini. BootStrap.ini is processed before a deployment share (which contains CustomSettings.ini) has been selected.

> **Caution**
>
> This property value must be specified in uppercase letters so that the deployment scripts can properly read it.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		

Component	Configured By		Scenario	Property Is Applicable
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
YES	Wizard page is not displayed, and the information on that page is not collected.
NO	Wizard page is displayed, and the information on that page is collected. This is the default value.

Expand table

Example
`[Settings] Priority=Default [Default] SkipWizard=NO SkipCapture=NO SkipAdminPassword=YES SkipApplications=NO SkipBDDWelcome=YES SkipComputerBackup=NO SkipDomainMembership=NO SkipUserData=NO SkipPackageDisplay=NO SkipLocaleSelection=NO SkipProductKey=YES`

SkipBitLocker

Indicates whether the **Specify the BitLocker configuration** wizard page is skipped.

For other properties that must be configured when this property is set to **YES**, see Providing Properties for Skipped Deployment Wizard Pages.

> ⊗ **Caution**
>
> This property value must be specified in uppercase letters so that the deployment scripts can properly read it.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
YES	Wizard page is not displayed, and the information on that page is not collected.
NO	Wizard page is displayed, and the information on that page is collected. This is the default value.

Expand table

Example
`[Settings] Priority=Default [Default] SkipWizard=NO SkipCapture=NO SkipApplications=NO SkipBDDWelcome=YES SkipBitLocker=YES SkipComputerBackup=NO SkipDomainMembership=NO SkipUserData=NO SkipPackageDisplay=NO SkipLocaleSelection=NO`

SkipBuild

Indicates whether the **Select a task sequence to execute on this computer** wizard page is skipped.

For other properties that must be configured when this property is set to **YES**, see Providing Properties for Skipped Deployment Wizard Pages.

This property value must be specified in uppercase letters so that the deployment scripts can properly read it.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
YES	Wizard page is not displayed, and the information on that page is not collected.
NO	Wizard page is displayed, and the information on that page is collected. This is the default value.

Expand table

Example
`[Settings] Priority=Default [Default] SkipWizard=NO SkipCapture=NO SkipAdminPassword=YES SkipApplications=NO SkipBDDWelcome=YES SkipBuild=YES SkipComputerBackup=NO SkipComputerName=NO SkipDomainMembership=NO SkipFinalSummary=NO SkipSummary=NO SkipUserData=NO SkipPackageDisplay=NO SkipLocaleSelection=NO`

SkipCapture

Indicates whether the **Specify whether to capture an image** wizard page is skipped.

For other properties that must be configured when this property is set to **YES**, see Providing Properties for Skipped Deployment Wizard Pages.

> **⊗ Caution**
>
> This property value must be specified in uppercase letters so that the deployment scripts can properly read it.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
YES	The wizard page is not displayed, and the information on that page is not collected.
NO	The wizard page is displayed, and the information on that page is collected. This is the default value.

Expand table

Example
`[Settings] Priority=Default [Default] SkipWizard=NO SkipCapture=YES SkipApplications=NO SkipComputerBackup=NO SkipDomainMembership=NO SkipUserData=NO SkipPackageDisplay=NO SkipLocaleSelection=NO`

SkipComputerBackup

Indicates whether the **Specify where to save a complete computer backup** wizard page is skipped.

For other properties that must be configured when this property is set to **YES**, see Providing Properties for Skipped Deployment Wizard Pages.

> **⊗ Caution**
>
> This property value must be specified in uppercase letters so that the deployment scripts can properly read it.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
YES	The wizard page is not displayed, and the information on that page is not collected.
NO	The wizard page is displayed, and the information on that page is collected. This is the default value.

Expand table

Example
`[Settings] Priority=Default [Default] SkipWizard=NO SkipCapture=NO SkipAdminPassword=NO SkipApplications=NO SkipComputerBackup=YES SkipDomainMembership=NO SkipUserData=NO`

Example
`SkipPackageDisplay=NO SkipLocaleSelection=NO`

SkipComputerName

Indicates whether the **Configure the computer name** wizard page is skipped.

For other properties that must be configured when this property is set to **YES**, see Providing Properties for Skipped Deployment Wizard Pages.

> **⊗ Caution**
>
> This property value must be specified in uppercase letters so that the deployment scripts can properly read it.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
YES	Wizard page is not displayed, and the information on that page is not collected.
NO	Wizard page is displayed, and the information on that page is collected. This is the default value.

Expand table

Example
`[Settings] Priority=Default [Default] SkipWizard=NO SkipCapture=NO SkipAdminPassword=NO SkipApplications=NO SkipComputerBackup=NO SkipComputerName=YES SkipDomainMembership=NO SkipUserData=NO SkipPackageDisplay=NO SkipLocaleSelection=NO`

SkipDomainMembership

Indicates whether the **Join the computer to a domain or workgroup** wizard page is skipped.

For other properties that must be configured when this property is set to **YES**, see Providing Properties for Skipped Deployment Wizard Pages.

> **⊗ Caution**
>
> This property value must be specified in uppercase letters so that the deployment scripts can properly read it.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	✗	|	LTI (Stand-alone MDT)	✓
CustomSettings.ini	✓	|		
MDT DB	✓	|	ZTI (Configuration Manager)	✗

Expand table

Value	Description
YES	The wizard page is not displayed, and the information on that page is not collected.
NO	The wizard page is displayed, and the information on that page is collected. This is the default value.

Expand table

Example
`[Settings] Priority=Default [Default] SkipWizard=NO SkipCapture=NO SkipAdminPassword=NO SkipApplications=NO SkipComputerBackup=NO SkipUserData=NO SkipPackageDisplay=NO SkipLocaleSelection=NO SkipDomainMembership=NO`

SkipFinalSummary

Indicates whether the **Operating system deployment completed successfully** wizard page is skipped.

For other properties that must be configured when this property is set to **YES**, see Providing Properties for Skipped Deployment Wizard Pages.

> **Note**
>
> This property value must be specified in uppercase letters so that the deployment scripts can properly read it.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
YES	The wizard page is not displayed, and the information on that page is not collected.
NO	The wizard page is displayed, and the information on that page is collected. This is the default value.

Expand table

Example
`[Settings] Priority=Default [Default] SkipWizard=NO SkipCapture=NO SkipApplications=NO SkipBDDWelcome=YES SkipComputerBackup=NO SkipComputerName=NO SkipDomainMembership=NO SkipFinalSummary=NO SkipUserData=NO SkipPackageDisplay=NO SkipLocaleSelection=NO SkipProductKey=YES`

SkipGroupSubFolders

By default, when specifying folders to be included when injecting drivers, patches (packages), and so on, values are specified something like:

ini

```ini
DriverGroup001=TopFolder\SecondFolder
```

```
PackageGroup001=TopFolder\SecondFolder
```

This would, by default, also include all sub-folders located under the "SecondFolder." If **SkipGroupSubFolders** is set to **YES** in CustomSettings.ini, this behavior will change so that the subfolders will be excluded and only the contents of "SecondFolder" will be added.

To exclude subfolders when matching against groups such as DriverGroup001, PackageGroup001, and so on, set **SkipGroupSubFolders** to **YES**.

> ⊗ **Caution**
>
> This property value must be specified in uppercase letters so that the deployment scripts can properly read it.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
YES	Do not include subfolders when matching against groups.
NO	Include subfolders when matching against groups. This is the default behavior.

Expand table

Example
`[Settings] Priority=Default [Default] SkipGroupSubFolders=NO`

SkipLocaleSelection

Indicates whether the **Locale Selection** wizard page is skipped.

For other properties that must be configured when this property is set to **YES**, see Providing Properties for Skipped Deployment Wizard Pages.

> ⊗ **Caution**
>
> This property value must be specified in uppercase letters so that the deployment scripts can properly read it.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
YES	The wizard page is not displayed, and the information on that page is not collected.
NO	The wizard page is displayed, and the information on that page is collected. This is the default value.

Expand table

Example
`[Settings] Priority=Default [Default] SkipWizard=NO SkipCapture=NO SkipApplications=NO SkipComputerBackup=NO SkipDomainMembership=NO SkipUserData=NO SkipPackageDisplay=NO SkipLocaleSelection=NO`

SkipPackageDisplay

Indicates whether the **Packages** wizard page is skipped.

For other properties that must be configured when this property is set to **YES**, see Providing Properties for Skipped Deployment Wizard Pages.

> ⊗ **Caution**

This property value must be specified in uppercase letters so that the deployment scripts can properly read it.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
YES	The wizard page is not displayed, and the information on that page is not collected.
NO	The wizard page is displayed, and the information on that page is collected. This is the default value.

Expand table

Example
`[Settings] Priority=Default [Default] SkipWizard=NO SkipCapture=NO SkipApplications=NO SkipComputerBackup=NO SkipDomainMembership=NO SkipUserData=NO SkipPackageDisplay=YES SkipLocaleSelection=NO`

SkipProductKey

Indicates whether the **Specify the product key needed to install this operating system** wizard page is skipped.

For other properties that must be configured when this property is set to **YES**, see Providing Properties for Skipped Deployment Wizard Pages.

⊗ Caution

This property value must be specified in uppercase letters so that the deployment scripts can properly read it.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
YES	The wizard page is not displayed, and the information on that page is not collected.
NO	The wizard page is displayed, and the information on that page is collected. This is the default value.

Expand table

Example
`[Settings] Priority=Default [Default] SkipWizard=NO SkipCapture=NO SkipAdminPassword=YES SkipApplications=NO SkipComputerBackup=NO SkipDomainMembership=NO SkipUserData=NO SkipPackageDisplay=NO SkipLocaleSelection=NO SkipProductKey=YES`

SkipRearm

This property is used to configure whether MDT rearms the Microsoft Office 2010 25-day activation grace period. If Microsoft Office 2010 is captured in a custom image, the user sees activation notification dialog boxes immediately after the image is deployed instead of 25-days after deployment.

By default, MDT rearms the Microsoft Office 2010 25-day activation grace period when running the LTISysprep.wsf script. You can set the value of this property to **YES** so that MDT skips the rearming of the Microsoft Office 2010 25-day activation grace period.

> ⊗ **Caution**
>
> This property value must be specified in uppercase letters so that the deployment scripts can properly read it.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
YES	MDT does not rearm the Microsoft Office 2010 25-day activation grace period.
NO	MDT rearms the Microsoft Office 2010 25-day activation grace period. This is the default value.

Expand table

Example
`[Settings] Priority=Default [Default] OSInstall=Y SkipCapture=YES SkipAdminPassword=NO SkipProductKey=YES SkipRearm=YES DoCapture=YES`

SkipRoles

Indicates whether the **Roles and Features** wizard page is skipped.

For other properties that must be configured when this property is set to **YES**, see Providing Properties for Skipped Deployment Wizard Pages.

> ⊗ **Caution**
>
> This property value must be specified in uppercase letters so that the deployment scripts can properly read it.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		

Component	Configured By		Scenario	Property Is Applicable
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
YES	The wizard page is not displayed, and the information on that page is not collected.
NO	The wizard page is displayed, and the information on that page is collected. This is the default value.

Expand table

Example
`[Settings] Priority=Default [Default] SkipWizard=NO SkipCapture=NO SkipAdminPassword=YES SkipApplications=NO SkipBDDWelcome=YES SkipTaskSequence=Yes SkipComputerBackup=NO SkipComputerName=NO SkipDomainMembership=NO SkipFinalSummary=NO SkipRoles=YES SkipSummary=NO SkipUserData=NO SkipPackageDisplay=NO SkipLocaleSelection=NO`

SkipSummary

Indicates whether the **Ready to begin** wizard page is skipped.

For other properties that must be configured when this property is set to **YES**, see Providing Properties for Skipped Deployment Wizard Pages.

> **⊗ Caution**
>
> This property value must be specified in uppercase letters so that the deployment scripts can properly read it.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
YES	The wizard page is not displayed, and the information on that page is not collected.
NO	The wizard page is displayed, and the information on that page is collected. This is the default value.

Expand table

Example
`[Settings] Priority=Default [Default] SkipWizard=NO SkipCapture=NO SkipAdminPassword=YES SkipApplications=NO SkipBDDWelcome=YES SkipTaskSequence=Yes SkipComputerBackup=NO SkipComputerName=NO SkipDomainMembership=NO SkipFinalSummary=NO SkipSummary=NO SkipUserData=NO SkipPackageDisplay=NO SkipLocaleSelection=NO`

SkipTaskSequence

Indicates whether the **Select a task sequence to execute on this computer** wizard page is skipped.

For other properties that must be configured when this property is set to **YES**, see Providing Properties for Skipped Deployment Wizard Pages.

> ⓘ Note
>
> Specify the **SkipBuild** property when using the Deployment Workbench to configure the Deployment Wizard to skip the **Select a task sequence to execute on this computer** wizard page.

> ⊗ Caution
>
> This property value must be specified in uppercase letters so that the deployment scripts can properly read it.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅

Component	Configured By		Scenario	Property Is Applicable
CustomSettings.ini	✅			
MDT DB	✅		ZTI (Configuration Manager)	❌

Expand table

Value	Description
YES	The wizard page is not displayed, and the information on that page is not collected.
NO	The wizard page is displayed, and the information on that page is collected. This is the default value.

Expand table

Example
`[Settings] Priority=Default [Default] SkipWizard=NO SkipCapture=NO SkipApplications=NO SkipBDDWelcome=YES SkipTaskSequence=NO SkipComputerBackup=NO SkipComputerName=NO SkipDomainMembership=NO SkipFinalSummary=NO SkipSummary=NO SkipUserData=NO SkipPackageDisplay=NO SkipLocaleSelection=NO`

SkipTimeZone

Indicates whether the **Set the Time Zone** wizard page is skipped.

For other properties that must be configured when this property is set to **YES**, see Providing Properties for Skipped Deployment Wizard Pages.

> ⊗ **Caution**
>
> This property value must be specified in uppercase letters so that the deployment scripts can properly read it.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌		LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅			

Component	Configured By		Scenario	Property Is Applicable
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
YES	The wizard page is not displayed, and the information on that page is not collected.
NO	The wizard page is displayed, and the information on that page is collected. This is the default value.

Expand table

Example
`[Settings] Priority=Default [Default] SkipWizard=NO SkipCapture=NO SkipAdminPassword=YES SkipApplications=NO SkipBDDWelcome=YES SkipTaskSequence=YES SkipComputerBackup=NO SkipComputerName=NO SkipDomainMembership=NO SkipFinalSummary=NO SkipSummary=NO SkipTimeZone=NO SkipUserData=NO SkipPackageDisplay=NO SkipLocaleSelection=NO`

SkipUserData

Indicates whether the **Specify whether to restore user data** and **Specify where to save your data and settings** wizard page is skipped.

For other properties that must be configured when this property is set to **YES**, see Providing Properties for Skipped Deployment Wizard Pages.

> ⊗ **Caution**
>
> This property value must be specified in uppercase letters so that the deployment scripts can properly read it.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
YES	The wizard page is not displayed, and the information on that page is not collected.
NO	The wizard page is displayed, and the information on that page is collected. This is the default value.

Expand table

Example
`[Settings] Priority=Default [Default] SkipWizard=NO SkipCapture=NO SkipAdminPassword=YES SkipApplications=NO SkipComputerBackup=NO SkipDomainMembership=NO SkipUserData=NO SkipPackageDisplay=NO SkipLocaleSelection=NO SkipProductKey=YES`

SkipWizard

Indicates whether the entire **Deployment Wizard** is skipped.

For other properties that must be configured when this property is set to **YES**, see Providing Properties for Skipped Deployment Wizard Pages.

> **⊗ Caution**
>
> This property value must be specified in uppercase letters so that the deployment scripts can properly read it.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
YES	The entire wizard is not displayed, and none of the information on the wizard pages is

Value	Description
	collected.
NO	The wizard is displayed, and the information on the enabled wizard pages is collected. This is the default value.

Expand table

Example
`[Settings] Priority=Default [Default] SkipWizard=YES`

SLShare

The network shared folder in which the deployment logs are stored at the end of the deployment process.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
shared_folder	The name of the network shared folder in which script logs are stored

Expand table

Example
`[Settings] Priority=Default [Default] DeployRoot=\\NYC-AM-FIL-01\Distribution$ ResourceRoot=\\NYC-AM-FIL-01\Resource$ UDShare=\\NYC-AM-FIL-01\MigData$ UDDir=%OSDComputerName% SLShare=\\NYC-AM-FIL-01\Logs$ UDProfiles=Administrator, User-01, ExtranetUser UserDataLocation=NONE SkipCapture=NO SkipAdminPassword=YES SkipProductKey=YES`

SLShareDynamicLogging

The network shared folder in which all MDT logs should be written during deployment. This is used for advanced real-time debugging only.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
shared_folder	The name of the network shared folder in which script logs are stored

Expand table

Example
`[Settings] Priority=Default [Default] DeployRoot=\\NYC-AM-FIL-01\Distribution$ ResourceRoot=\\NYC-AM-FIL-01\Resource$ UDShare=\\NYC-AM-FIL-01\MigData$ UDDir=%OSDComputerName% SLShare=\\NYC-AM-FIL-01\Logs$ SLShareDynamicLogging=\\NYC-AM-FIL-01\Logs$ UDProfiles=Administrator, User-01, ExtranetUser UserDataLocation=NONE SkipCapture=NO SkipAdminPassword=YES SkipProductKey=YES`

SMSTSAssignUserMode

Specifies whether user device affinity (UDA) should be enabled and whether approval is required. This property only works with the UDA feature in Configuration Manager.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	❌
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
Auto	The affinity between a user and the target device is established, and approval is automatically performed.
Pending	The affinity between a user and the target device is established, and approval is submitted for Configuration Manager administrator approval.
Disable	The affinity between a user and the target device is not established.

Expand table

Example
`[Settings] Priority=Default [Default] SMSTSAssignUserMode=Auto SMSTSUdaUsers=Fabrikam\Ken, Fabrikam\Pilar`

SMSTSRunCommandLineUserName

Specifies the user name in *Domain\User_Name* format that should be used with a **Run Command Line** step that is configured to run as a user.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
user_name	Specifies the user name in that should be used with a **Run Command Line** step

Expand table

Example
`[Settings] Priority=Default [Default] SMSTSRunCommandLineUserName=Fabrikam\Ken SMSTSRunCommandLineUserPassword=<complex_password>`

SMSTSRunCommandLineUserPassword

Specifies the password that should be used with a **Run Command Line** step that is configured to run as a user.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
user_password	Specifies the password that should be used with a **Run Command Line** step

Expand table

Example
`[Settings] Priority=Default [Default] SMSTSRunCommandLineUserName=Fabrikam\Ken SMSTSRunCommandLineUserPassword=<complex_password>`

SMSTSUdaUsers

Specifies the users who will be assigned affinity with a specific device using the UDA feature, which is available only in Configuration Manager.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	❌
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
user1, user2, ...	The comma-separated list of users in *Domain\User_Name* format that will be assigned affinity with the target device. Note: You can only use the NetBIOS domain name in this value, such as *Fabrikam\Ken*. You cannot use the fully qualified domain name (fabrikam.com\Ken) or the UPN notation (ken@fabrikam.com).

Expand table

Example
`[Settings] Priority=Default [Default] SMSTSAssignUserMode=Auto SMSTSUdaUsers=Fabrikam\Ken, Fabrikam\Pilar`

SQLServer

The identity of the computer running SQL Server that performs a database query that returns property values from columns in the table specified in the **Table** property. The query is based on parameters specified in the **Parameters** and **ParameterCondition** properties. The instance of SQL Server on the computer is specified in the **Instance** property.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	✅	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
SQL_server	The name of the computer running SQL Server

Expand table

Example
`[Settings] Priority=Computers, Default [Default] OSInstall=YES ScanStateArgs=/v:5 /o /c LoadStateArgs=/v:5 /c /lac [Computers] SQLServer=NYC-SQL-01 SQLShare=SQL$ Database=MDTDB Instance=SQLEnterprise2005 Table=Computers Parameters=SerialNumber, AssetTag ParameterCondition=OR`

SQLShare

The name of a shared folder on the computer running SQL Server (specified by the **SQLServer** property). The credentials used for authentication are provided by the **UserDomain**, **UserID**, and **UserPassword** properties (for LTI and ZTI) or by the Configuration Manager Advanced Client account credentials (ZTI only).

> **Note**
>
> This property must be specified to perform Integrated Windows authentication. This is the recommended authentication method, rather than using the **DBID** and **DBPwd** properties (which support the SQL Server authentication method).

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	✅	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
shared_folder	The name of a shared folder on the computer running SQL Server

Expand table

Example
`[Settings] Priority=Computers, Default Properties=MyCustomProperty [Default] OSInstall=YES ScanStateArgs=/v:5 /o /c LoadStateArgs=/v:5 /c /lac [Computers] SQLServer=NYC-SQL-01`

Example
`SQLShare=SQL$ Database=MDTDB Instance=MDT2010 Table=Computers Parameters=SerialNumber, AssetTag ParameterCondition=OR`

StatePath

This property is used to set the path where the user state migration data will be stored, which can be a UNC path, a local path, or a relative path. The OSDStateStorePath property takes precedence over the **StatePath** or UserDataLocation property when those properties are also specified.

> **Note**
>
> This property is provided for backward compatibility with previous versions of MDT. Use the OSDStateStorePath property instead.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	❌	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
Path	The path where the user state migration data will be stored, which can be a UNC path, a local path, or a relative path

Expand table

Example
`[Settings] Priority=Default [Default] StatePath=\\fs1\Share\Replace ComputerBackupLocation=\\fs1\Share\ComputerBackup\Client01`

StoredProcedure

The name of the stored procedure used when performing a database query that returns property values from columns in the table or view. The stored procedure is located in the database specified in the **Database** property. The computer running SQL Server is specified in the **SQLServer** property. The instance of SQL Server on the computer is specified in the **Instance** property. The name of the stored procedure is specified in the **StoredProcedure** property.

For more information about using a stored procedure to query a SQL Server database, see the section, "Deploying Applications Based on Earlier Application Versions", in the MDT document *Microsoft Deployment Toolkit Samples Guide*.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	✅	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
stored_procedure	The name of the stored procedure used to query the SQL Server database

Expand table

Example
`[Settings] Priority=DynamicPackages, Default [Default] OSInstall=YES [DynamicPackages] SQLDefault=DB_DynamicPackages [DB_DynamicPackages] SQLServer=SERVER1 Database=MDTDB StoredProcedure=RetrievePackages Parameters=MacAddress SQLShare=Logs Instance=MDT2013 Port=1433 Netlib=DBNMPNTW`

SupportsHyperVRole

Specifies whether the processor resources on the target computer can support the Hyper-V server role in Windows Server. This property is True if the value for the following properties is set to **TRUE**:

- **SupportsNX**
- **SupportsVT**

- **Supports64Bit**

  Each of the previous properties is set using information from the **CPUID** interface. For further information collected about VMs and information returned from the **CPUID** interface, see the following properties:

- **IsHypervisorRunning**
- **IsVM**
- **SupportsNX**
- **SupportsVT**
- **Supports64Bit**
- **VMPlatform**

> **Note**
>
> This property is dynamically set by the MDT scripts and is not configured in CustomSettings.ini or the MDT DB. Treat this property as read only.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	❌	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
TRUE	The processor resources of the target computer can support the Hyper-V server role in Windows Server.
FALSE	The processor resources of the target computer cannot support the Hyper-V server role in Windows Server.

Expand table

Example
None

SupportsNX

Specifies whether the processor resources on the target computer support the No Execute (NX) technology. The NX technology is used in processors to segregate areas of memory for use by either storage of processor instructions (code) or for storage of data. This property is set using information from the **CPUID** interface.

For further information collected about VMs and information returned from the **CPUID** interface, see the following properties:

- **IsHypervisorRunning**
- **IsVM**
- **SupportsHyperVRole**
- **SupportsVT**
- **Supports64Bit**
- **VMPlatform**

> **Note**
>
> This property is dynamically set by the MDT scripts and is not configured in CustomSettings.ini or the MDT DB. Treat this property as read only.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	❌	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
TRUE	The processor resources of the target computer support NX technology.
FALSE	The processor resources of the target computer do not support NX technology.

Expand table

Example
None

SupportsVT

Specifies whether the processor resources on the target computer support the Virtualization Technology (VT) feature. VT is used to support current virtualized environments, such as Hyper-V. This property is set using information from the CPUID interface.

For further information collected about VMs and information returned from the CPUID interface, see the following properties:

- **IsHypervisorRunning**
- **IsVM**
- **SupportsHyperVRole**
- **SupportsNX**
- **Supports64Bit**
- **VMPlatform**

> **Note**
>
> This property is dynamically set by the MDT scripts and is not configured in CustomSettings.ini or the MDT DB. Treat this property as read only.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅

Component	Configured By		Scenario	Property Is Applicable
CustomSettings.ini	❌	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
TRUE	The processor resources of the target computer support VT technology.
FALSE	The processor resources of the target computer do not support VT technology.

Expand table

Example
None

Supports64Bit

Specifies whether the processor resources on the target computer support Windows 64-bit operating systems. Most modern virtualization environments require 64-bit processor architecture. This property is set using information from the **CPUID** interface.

For further information collected about VMs and information returned from the **CPUID** interface, see the following properties:

- **IsHypervisorRunning**
- **IsVM**
- **SupportsHyperVRole**
- **SupportsNX**
- **SupportsVT**
- **VMPlatform**

> ⓘ **Note**
>
> This property is dynamically set by the MDT scripts and is not configured in CustomSettings.ini or the MDT DB. Treat this property as read only.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	❌	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
TRUE	The processor resources of the target computer support a Windows 64-bit operating system.
FALSE	The processor resources of the target computer do not support a Windows 64-bit operating system.

Expand table

Example
None

SysVolPath

Specifies the fully qualified, non-UNC path to a directory on a fixed disk of the local computer.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
path	Specifies the fully qualified, non-UNC path to a directory on a fixed disk of the local computer

Expand table

Example
`[Settings] Priority=Default [Default] SysVolPath=%DestinationLogicalDrive%\Windows\Sysvol`

Table

The name of the table or view to be used in performing a database query that returns property values from columns in the table or view. The query is based on parameters specified in the **Parameters** and **ParameterCondition** properties. The table or view is located in the database specified in the **Database** property. The computer running SQL Server is specified in the **SQLServer** property. The instance of SQL Server on the computer is specified in the **Instance** property.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	✅	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
table_name	The name of the table or view to be queried for property values

Expand table

Example
`[Settings] Priority=Computers, Default [Default] OSInstall=YES ScanStateArgs=/v:5 /o /c LoadStateArgs=/v:5 /c /lac [Computers] SQLServer=NYC-SQL-01 SQLShare=SQL$ Database=MDTDB Instance=MDT2010 Table=Computers Parameters=SerialNumber, AssetTag ParameterCondition=OR`

TaskSequenceID

Identifies the operating system task sequence to be deployed to the target computer. The task sequence ID is created on the Task Sequences node in the Deployment Workbench. The **TaskSequenceID** property allows alphanumeric characters, hyphens (-), and underscores (_). The **TaskSequenceID** property cannot be blank or contain spaces.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
task_sequence_id	Identifier of the operating system task sequence defined in the Deployment Workbench for the target operating system being deployed Note: Be sure to use the **TaskSequenceID** specified in the Deployment Workbench UI, not the GUID of the **TaskSequenceID**.

Expand table

Example
`[Settings] Priority=Default [Default] TaskSequenceID=BareMetal`

TaskSequenceName

Specifies the name of the task sequence being run.

> **Note**
>
> This property is dynamically set by the MDT scripts and is not configured in CustomSettings.ini or the MDT DB. Treat this property as read only.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	❌	|		
MDT DB	❌	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
task_sequence_name	Name of the task sequence being run, such as Deploy *Windows 8.1 to Reference Computer*

Expand table

Example
None

TaskSequenceVersion

Specifies the version of the task sequence being run.

> **Note**
>
> This property is dynamically set by the MDT scripts and is not configured in CustomSettings.ini or the MDT DB. Treat this property as read only.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	❌	|		
MDT DB	❌	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
task_sequence_version	Version of the task sequence being run, such as *1.00*

Expand table

Example
None

TimeZoneName

The time zone in which the target computer is located. This value is inserted into the appropriate configuration settings in Unattend.xml.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
time_zone_name	The text value that indicates the time zone where the target computer is located

Expand table

Example
`[Settings] Priority=Default [Default] TimeZoneName=Pacific Standard Time DeployRoot=\\NYC-AM-FIL-01\Distribution$ ResourceRoot=\\NYC-AM-FIL-01\Resource$ UDShare=\\NYC-AM-FIL-01\MigData$ UDDir=%OSDComputerName% SLShare=\\NYC-AM-FIL-01\Logs$ UDProfiles=Administrator, User-01, ExtranetUser UserDataLocation=NONE`

ToolRoot

Specifies the UNC path to the Tools*proc_arch* folder (where *proc_arch* is the processor architecture of the currently running operating system and can have a value of **x86** or **x64**), which is immediately beneath the root of the folder structure specified in the **DeployRoot** property. The Tools*proc_arch* folder contains utilities that MDT uses during the deployment process.

> **Note**
>
> This property is dynamically set by the MDT scripts and is not configured in CustomSettings.ini or the MDT DB. Treat this property as read only.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	❌	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
path	The UNC or local path to the Tools*proc_arch* folder (where *proc_arch* is the processor architecture of the currently running operating system and can have a value of **x86** or **x64**) immediately beneath the root of the folder structure specified by the **DeployRoot** property

Expand table

Example
None

TPMOwnerPassword

The TPM password (also known as the *TPM administration password*) for the owner of the target computer. The password can be saved to a file or stored in AD DS.

> **Note**

If the TPM ownership is already set or TPM ownership is not allowed, then the **TPMOwnerPassword** property is ignored. If the TPM password is needed and the **TPMOwnerPassword** property is not provided, the TPM password is set to the local Administrator password.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
password	The TPM password for the owner of the target computer

Expand table

Example
`[Settings] Priority=Default [Default] BDEDriveLetter=S: BDEDriveSize=2000 BDEInstall=TPMKey BDERecoveryKey=TRUE BDEKeyLocation=C: TPMOwnerPassword=<complex_password> BackupShare=\\NYC-AM-FIL-01\Backup$ BackupDir=%OSDComputerName% DeployRoot=\\NYC-AM-FIL-01\Distribution$ ResourceRoot=\\NYC-AM-FIL-01\Resource$ UDShare=\\NYC-AM-FIL-01\MigData$ UDDir=%OSDComputerName%`

UDDir

The folder in which the user state migration data is stored. This folder exists beneath the network shared folder specified in **UDShare**.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		

Component	Configured By		Scenario	Property Is Applicable
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
folder	The name of the folder that exists beneath the network shared folder

Expand table

Example
`[Settings] Priority=Default [Default] DeployRoot=\\NYC-AM-FIL-01\Distribution$ ResourceRoot=\\NYC-AM-FIL-01\Resource$ UDShare=\\NYC-AM-FIL-01\MigData$ UDDir=%OSDComputerName% SLShare=\\NYC-AM-FIL-01\Logs$ UDProfiles=Administrator, User-01, ExtranetUser UserDataLocation=NONE SkipCapture=NO`

UDProfiles

A comma-delimited list of user profiles that need to be saved by Scanstate.exe during the State Capture Phase.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
user_profiles	The list of user profiles to be saved, separated by commas

Expand table

Example
`[Settings] Priority=Default [Default] DeployRoot=\\NYC-AM-FIL-01\Distribution$ ResourceRoot=\\NYC-AM-FIL-01\Resource$ UDShare=\\NYC-AM-FIL-01\MigData$ UDDir=%OSDComputerName% SLShare=\\NYC-AM-FIL-01\Logs$ UDProfiles=Administrator, User-01, ExtranetUser UserDataLocation=NONE SkipCapture=NO`

UDShare

The network share where user state migration data is stored.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
UNC_path	The UNC path to the network share where user state migration data is stored

Expand table

Example
`[Settings] Priority=Default [Default] DeployRoot=\\NYC-AM-FIL-01\Distribution$ ResourceRoot=\\NYC-AM-FIL-01\Resource$ UDShare=\\NYC-AM-FIL-01\MigData$ UDDir=%OSDComputerName% SLShare=\\NYC-AM-FIL-01\Logs$ UDProfiles=Administrator, User-01, ExtranetUser UserDataLocation=NONE SkipCapture=NO`

UILanguage

The default language to be used with the target operating system. If not specified, the **Deployment Wizard** uses the language configured in the image being deployed.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
UI_language	The default language for the operating system on the target computer

Expand table

Example
`[Settings] Priority=Default [Default] UserLocale=en-us UILanguage=en-us KeyboardLocale=0409:00000409`

UserDataLocation

The location in which USMT stores user state migration data.

> ⊗ **Caution**
>
> This property value must be specified in uppercase letters so that the deployment scripts can properly read it.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
blank	If **UserDataLocation**is not specified or is left blank, the Deployment Wizard will default to using the AUTO behavior.
UNC_path	The UNC path to the network shared folder where the user state migration data is stored.
AUTO	The deployment scripts store the user state migration data on a local hard disk if space is available. Otherwise, the user state migration data is saved to a network location, which is specified in the **UDShare** and **UDDir** properties.
NETWORK	The user state migration data is stored in the location designated by the **UDShare** and **UDDir** properties.
NONE	The user state migration data is not saved.

Expand table

Example
`[Settings] Priority=Default [Default] OSInstall=YES ScanStateArgs=/v:5 /o /c LoadStateArgs=/v:5 /c /lac DoCapture=YES BackupShare=\\NYC-AM-FIL-01\Backup$ BackupDir=%OSDComputerName% UserDataLocation=NETWORK DeployRoot=\\NYC-AM-FIL-01\Distribution$ ResourceRoot=\\NYC-AM-FIL-01\Resource$ UDShare=\\NYC-AM-FIL-01\MigData$ UDDir=%OSDComputerName%`

UserDomain

The domain in which a user's credentials (specified in the **UserID** property) reside.

> ⓘ Note
>
> For a completely automated LTI deployment, provide this property in both CustomSettings.ini and BootStrap.ini. However, note that storing the user credentials in these files stores the credentials in clear text and therefore is not secure.

Expand table

Component	Configured By	Scenario	Property Is Applicable
BootStrap.ini	✅	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅		

Component	Configured By		Scenario	Property Is Applicable
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
domain	The name of the domain where the user account credentials reside

Expand table

Example
`[Settings] Priority=Default [Default] OSInstall=YES ScanStateArgs=/v:5 /o /c LoadStateArgs=/v:5 /c /lac DeployRoot=\\NYC-AM-FIL-01\Distribution$ ResourceRoot=\\NYC-AM-FIL-01\Resource$ UserDataLocation=NONE UserDomain=WOODGROVEBANK UserID=NYC Help Desk Staff UserPassword=<complex_password>`

UserID

The user credentials for accessing network resources.

> **Note**
>
> For a completely automated LTI deployment, provide this property in both CustomSettings.ini and BootStrap.ini. However, note that storing the user credentials in these files stores the credentials in clear text and therefore is not secure.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	✅	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
user_id	The name of the user account credentials used to access the network resources

Expand table

Example
`[Settings] Priority=Default [Default] OSInstall=YES ScanStateArgs=/v:5 /o /c LoadStateArgs=/v:5 /c /lac DeployRoot=\\NYC-AM-FIL-01\Distribution$ ResourceRoot=\\NYC-AM-FIL-01\Resource$ UserDataLocation=NONE UserDomain=WOODGROVEBANK UserID=NYC-HelpDesk UserPassword=<complex_password>`

UserLocale

The user locale to be used with the target operating system. If not specified, the **Deployment Wizard** uses the user locale configured in the image being deployed.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
user_locale	The locale for the user on the target computer. The value is specified as a text value (en-us).

Expand table

Example 1
`[Settings] Priority=Default [Default] UserLocale=en-us KeyboardLocale=0409:00000409`

Expand table

Example 2
`[Settings] Priority=Default [Default] UserLocale=en-us KeyboardLocale=en-us`

UserPassword

The password for user credentials specified in the **UserID** property.

> **Note**
>
> For a completely automated LTI deployment, provide this property in both CustomSettings.ini and BootStrap.ini. However, note that storing the user credentials in these files stores the credentials in clear text and therefore is not secure.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	✅	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
user_password	The password for the user account credentials

Expand table

Example
`[Settings] Priority=Default [Default] UserDataLocation=NONE UserDomain=WOODGROVEBANK UserID=NYC-HelpDesk UserPassword=<complex_password>`

USMTConfigFile

The USMT configuration XML file that should be used when running **Scanstate** and **Loadstate**.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
USMTConfigFile	The name of the XML configuration file that should be used when running Scanstate.exe and Loadstate.exe

Expand table

Example
`[Settings] Priority=Default [Default] OSInstall=YES ScanStateArgs=/v:5 /o /c LoadStateArgs=/v:5 /c /lac DeployRoot=\\NYC-AM-FIL-01\Distribution$ ResourceRoot=\\NYC-AM-FIL-01\Resource$ UDShare=\\NYC-AM-FIL-01\MigData$ UDDir=%OSDComputerName% SLShare=\\NYC-AM-FIL-01\Logs$ USMTMigFiles1=MigApp.xml USMTMigFiles2=MigUser.xml USMTMigFiles3=MigSys.xml USMTMigFiles4=MigCustom.xml USMTConfigFile=USMTConfig.xml UserDataLocation=NONE`

USMTLocal

This property specifies whether the USMT user state information is stored locally on the target computer. This property is primarily used by the ZTIUserState.wsf and ZTIBackup.wsf scripts to indicate that the **Request State Store** and **Release State Store** task sequence steps for Configuration Manager deployments are skipped. For more information, see the OSDStateStorePath property.

> **Note**
>
> This property should only be used in the circumstance described in the OSDStateStorePath property).

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	❌
CustomSettings.ini	❌	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
TRUE	The USMT user state information is stored locally on the target computer, and the **Request State Store** and **Release State Store** task sequence steps are skipped.
FALSE	The USMT user state information is not stored locally on the target computer, and the **Request State Store** and **Release State Store** task sequence steps are performed.

Expand table

Example

```
[Settings] Priority=Default [Default] OSInstall=YES ScanStateArgs=/v:5 /o /c
LoadStateArgs=/v:5 /c /lac DeployRoot=\\NYC-AM-FIL-01\Distribution$ ResourceRoot=\\NYC-AM-
FIL-01\Resource$ UDShare=\\NYC-AM-FIL-01\MigData$ UDDir=%OSDComputerName% SLShare=\\NYC-AM-
FIL-01\Logs$ USMTLocal=TRUE USMTMigFiles001=MigApp.xml USMTMigFiles002=MigUser.xml
USMTMigFiles003=MigSys.xml USMTMigFiles004=MigCustom.xml UserDataLocation=NONE
```

USMTMigFiles

A list of files in XML format that are used by USMT (Scanstate.exe) to identify user state migration information to be saved. When this property is not specified, the ZTIUserState.wsf script uses MigApp.xml, MigUser.xml, and MigSys.xml. Otherwise, ZTIUserState.wsf uses the files explicitly referenced in this property. The **USMTMigFiles** property has a numeric suffix (for example, **USMTMigFiles001** or **USMTMigFiles002**).

> ⓘ **Note**
>
> Use this property to specify the XML files to be used by Scanstate.exe instead of using the **/I** parameter in the **ScanStateArgs** property. This prevents the ZTIUserState.wsf script from potentially duplicating the same list of XML files.

> ⓘ **Note**

This property name can be specified using single-digit nomenclature (**USMTMigFiles1**) or triple-digit nomenclature (**USMTMigFiles001**).

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	❌	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
USMTMigFile	The name of the .xml file to be used as input for Scanstate.exe, on separate lines. If not specified, the default is MigApp.xml, MigUser.xml, and MigSys.xml. Note: If this value is specified, the default files (MigApp.xml, MigUser.xml, and MigSys.xml) must also be added to the list if these files are to be included.

Expand table

Example
`[Settings] Priority=Default [Default] OSInstall=YES ScanStateArgs=/v:5 /o /c LoadStateArgs=/v:5 /c /lac DeployRoot=\\NYC-AM-FIL-01\Distribution$ ResourceRoot=\\NYC-AM-FIL-01\Resource$ UDShare=\\NYC-AM-FIL-01\MigData$ UDDir=%OSDComputerName% SLShare=\\NYC-AM-FIL-01\Logs$ USMTMigFiles001=MigApp.xml USMTMigFiles002=MigUser.xml USMTMigFiles003=MigSys.xml USMTMigFiles004=MigCustom.xml UserDataLocation=NONE`

USMTOfflineMigration

This property determines whether MDT uses USMT to perform an offline user state migration. In an offline migration, the capture is performed in Windows PE instead of the existing operating system.

Offline migration is using USMT is performed for:

- UDI always, regardless of the setting of the **USMTOfflineMigration** property

- ZTI only for the MDT Refresh Computer deployment scenario and only when the **USMTOfflineMigration** property is set to **"TRUE"**

  > **Note**
  >
  > You cannot perform USMT offline user state migration in the MDT New Computer deployment scenario using ZTI.

- LTI for the:

  1. MDT New Computer deployment scenario using the **Move Data and Settings** wizard page in the Deployment Wizard
  2. MDT Refresh Computer deployment scenario and only when the **USMTOfflineMigration** property is set to **"TRUE"**

  For more information about using MDT and USMT to perform an offline user state migration, see "Configure USMT Offline User State Migration".

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
TRUE	MDT uses USMT to perform an offline user state migration.
Any other value	MDT does not perform an offline user state migration. Instead, user state migration is captured in the existing operating system. This is the default value.

Expand table

Example
`[Settings] Priority=Default [Default] OSInstall=YES SkipUserData=YES USMTOfflineMigration=TRUE DoNotFormatAndPartition=YES OSDStateStorePath=\\WDG-MDT-01\StateStore$`

UUID

The Universal Unique Identifier (UUID) stored in the System Management BIOS of the target computer.

The format for UUID is a 16-byte value using hexadecimal digits in the following format: *12345678-1234-1234-1234-123456789ABC*. Use this property to create a subsection that contains settings targeted to a specific computer.

> **Note**
>
> This property is dynamically set by MDT scripts and cannot have its value set in CustomSettings.ini or the MDT DB. Treat this property as read only. However, you can use this property within CustomSettings.ini or the MDT DB, as shown in the following examples, to aid in defining the configuration of the target computer.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	❌	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
UUID	The UUID of the target computer

Expand table

Example
None

ValidateDomainCredentialsUNC

This property is used to specify a UNC path to a network shared folder that is used to validate the credentials provided for joining the target computer to a domain. The

credentials being validated are specified in the **DomainAdmin**, **DomainAdminDomain**, and **DomainAdminPassword** properties.

> ⓘ Note
>
> Ensure that no other properties in MDT use the server sharing the folder in this property. Using a server that is already referenced by other MDT properties could result in improper validation of the credentials.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
unc_path	Specifies the fully qualified UNC path to a network shared folder

Expand table

Example
`[Settings] Priority=Default [Default] ValidateDomainCredentialsUNC=\\wdg-fs-01\Source$`

VHDCreateDiffVHD

This property is used to specify the name of a differencing VHD (also known as a *child VHD*) file. A differencing VHD is similar to a dynamically expanding VHD but contains only the modified disk blocks of the associated parent VHD. The parent VHD is read only, so you must modify the differencing VHD. The differencing VHD file is created in the same folder as the parent VHD file, so only the file name is specified in this property. This property is only valid for the MDT New Computer deployment scenario.

> ⓘ Note

All parent VHD files created by MDT are stored in the VHD folder in the root of the parent drive.

This property is commonly set using a task sequence step created using the **Create Virtual Hard Disk (VHD)** task sequence type. You can override the value the **Create Virtual Hard Disk (VHD)** task sequence step sets by configuring this property in CustomSettings.ini.

> ⓘ **Note**
>
> To configure this property in CustomSettings.ini, you must add this property to the **Properties** line in CustomSettings.ini.

For related properties that are used with VHD files, see:

- **VHDCreateFileName**
- **VHDCreateSizeMax**
- **VHDCreateSource**
- **VHDCreateType**
- **VHDDisks**
- **VHDInputVariable**
- **VHDOutputVariable**
- **VHDTargetDisk**

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	❌	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
filename	Specifies the name of the differencing VHD file, which is located in the same folder as the parent VHD file The differencing VHD file cannot have the same name as the parent VHD file.
RANDOM	Automatically generates a random name for the differencing VHD file, which is located in the same folder as the parent VHD file

Expand table

Example
`[Settings] Priority=Default [Default] VHDCreateDiffVHD=Win7Diff_C.vhd VHDInputVariable=VHDTargetDisk`

VHDCreateFileName

This property is used to specify the name of a VHD file. The type of VHD file is based on the value of the **VHDCreateType** property. The property only includes the file name, not the path to the file name, and is valid only for the MDT New Computer deployment scenario.

> **Note**
>
> The VHD files created by MDT are stored in the VHD folder in the root of the parent drive.

This property is commonly set using a task sequence step created using the **Create Virtual Hard Disk (VHD)** task sequence type. You can override the value the **Create Virtual Hard Disk (VHD)** task sequence step sets by configuring this property in CustomSettings.ini.

> **Note**
>
> To configure this property in CustomSettings.ini, you must add this property to the **Properties** line in CustomSettings.ini.

For related properties that are used with VHD files, see:

- **VHDCreateDiffVHD**

- VHDCreateSizeMax
- VHDCreateSource
- VHDCreateType
- VHDDisks
- VHDInputVariable
- VHDOutputVariable
- VHDTargetDisk

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	❌	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
file_name	Specifies the name of the VHD file
RANDOM	Automatically generates a random name for the VHD file, which is located in the VHD folder in the root of the parent drive
Blank	Same a **RANDOM**

Expand table

Example
`[Settings] Priority=Default [Default] VHDCreateSizeMax=130048 VHDCreateType=EXPANDABLE VHDCreateFileName=Win7_C.vhd VHDInputVariable=VHDTargetDisk`

VHDCreateSizeMax

This property is used to specify the maximum size of a VHD file in megabytes (MB). The size of the VHD file at creation time is based on the type of VHD file being created. For

more information, see the VHDCreateType property. This property is valid only for the MDT New Computer deployment scenario.

> **Note**
>
> If this property is not specified, the default value for the maximum size of a VHD file is 90% of the available disk space on the parent disk.

This property is commonly set using a task sequence step created using the **Create Virtual Hard Disk (VHD)** task sequence type. You can override the value that the **Create Virtual Hard Disk (VHD)** task sequence step sets by configuring this property in CustomSettings.ini.

> **Note**
>
> To configure this property in CustomSettings.ini, you must add this property to the **Properties** line in CustomSettings.ini.

For related properties that are used with VHD files, see:

- **VHDCreateDiffVHD**
- **VHDCreateFileName**
- **VHDCreateSource**
- **VHDCreateType**
- **VHDDisks**
- **VHDInputVariable**
- **VHDOutputVariable**
- **VHDTargetDisk**

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	❌	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
size	The maximum size of the VHD file specified in MB. For example, 130,048 MB equals 127 GB. The default value is 90% of the available disk space on the parent disk.

Expand table

Example
`[Settings] Priority=Default [Default] VHDCreateSizeMax=130048 VHDCreateType=FIXED VHDCreateFileName=Win7_C.vhd VHDInputVariable=VHDTargetDisk`

VHDCreateSource

This property is used to specify the name of a VHD file that is used as a template (source) for creating a new VHD file. You can specify the file name using a UNC path, local path, relative path, or just the file name. If just the file name is specified, then MDT attempts to find the VHD file on the target computer. This property is valid only for the MDT New Computer deployment scenario.

This property is commonly set using a task sequence step created using the **Create Virtual Hard Disk (VHD)** task sequence type. You can override the value that the **Create Virtual Hard Disk (VHD)**task sequence step sets by configuring this property in CustomSettings.ini.

> ⓘ **Note**
>
> To configure this property in CustomSettings.ini, you must add this property to the **Properties** line in CustomSettings.ini.

For related properties that are used with VHD files, see:

- **VHDCreateDiffVHD**
- **VHDCreateFileName**
- **VHDCreateSizeMax**
- **VHDCreateType**
- **VHDDisks**

- **VHDInputVariable**
- **VHDOutputVariable**
- **VHDTargetDisk**

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	❌	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
name	The file name, which can be specified using a UNC path, local path, relative path, or just the file name. If just the file name is specified, then MDT attempts to find the VHD file on the target computer.

Expand table

Example
`[Settings] Priority=Default [Default] VHDCreateSizeMax=130048 VHDCreateSource=\\wdg-mdt-01\vhds\win7_template.vhd VHDCreateType=FIXED VHDCreateFileName=Win7_C.vhd VHDInputVariable=VHDTargetDisk`

VHDCreateType

This property is used to specify the type of VHD file that is specified in the **VHDCreateFileName** property and can be one of the following VHD file types:

- **Fixed VHD file**. For this VHD type, the size of the VHD specified at creation is allocated and does not change automatically after creation. For example, if you create a 24-gigabyte (GB) fixed VHD file, the file will be approximately 24 GB in size (with some space used for the internal VHD structure) regardless of how much information is stored in the VHD file.
- **Dynamically expanding VHD file**. For this VHD type, only a small percentage of the size of the VHD specified at creation time is allocated. Then, the VHD file

continues to grow as more and more information is stored in it. However, the VHD file cannot grow beyond the size specified at creation. For example, if you create a 24 GB dynamically expanding VHD, it will be small at creation. However, as information is stored in the VHD file, the file will continue to grow but never exceed the maximum size of 24 GB.

This property is only valid for the MDT New Computer deployment scenario.

> ⓘ **Note**
>
> The maximum size of the VHD file is specified in the **VHDCreateSizeMax** property.

This property is commonly set using a task sequence step created using the **Create Virtual Hard Disk (VHD)** task sequence type. You can override the value that the **Create Virtual Hard Disk (VHD)** task sequence step sets by configuring this property in CustomSettings.ini.

> ⓘ **Note**
>
> To configure this property in CustomSettings.ini, you must add this property to the **Properties** line in CustomSettings.ini.

For related properties that are used with VHD files, see:

- **VHDCreateDiffVHD**
- **VHDCreateFileName**
- **VHDCreateSizeMax**
- **VHDCreateSource**
- **VHDDisks**
- **VHDInputVariable**
- **VHDOutputVariable**
- **VHDTargetDisk**

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	❌	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
EXPANDABLE	Creates a fixed VHD file
FIXED	Creates a dynamically expanding VHD file

Expand table

Example
`[Settings] Priority=Default [Default] VHDCreateSizeMax=130048 VHDCreateType=EXPANDABLE VHDCreateFileName=Win7_C.vhd VHDInputVariable=VHDTargetDisk`

VHDDisks

This property contains a list of the physical drive numbers assigned to VHD files separated by spaces. Each time a VHD file is created, MDT adds the disk index of the newly created disk to this property using the **Index** property of the **Win32_DiskDrive** WMI class.

> ⓘ **Note**
>
> This property is dynamically set by the MDT scripts and is not configured in CustomSettings.ini or the MDT DB. Treat this property as read only.

This property is commonly set using a task sequence step created using the **Create Virtual Hard Disk (VHD)** task sequence type. You can override the value that the **Create Virtual Hard Disk (VHD)** task sequence step sets by configuring this property in CustomSettings.ini.

> ⓘ **Note**

To configure this property in CustomSettings.ini, you must add this property to the **Properties** line in CustomSettings.ini.

For related properties that are used with VHD files, see:

- **VHDCreateDiffVHD**
- **VHDCreateFileName**
- **VHDCreateSizeMax**
- **VHDCreateSource**
- **VHDCreateType**
- **VHDInputVariable**
- **VHDOutputVariable**
- **VHDTargetDisk**

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	❌	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
index1 index2 index3	A list of the physical drive numbers assigned to the VHD files separated by spaces—for example, *1 2 5*.

Expand table

Example
None

VHDInputVariable

This property contains a variable that contains the drive on the target computer where the VHD files will be created. MDT creates the VHD files in the VHD folder in the root of this drive.

> ⓘ Note
>
> If this property is omitted, MDT attempts to create the VHD files in the VHD folder in the root of the first system drive.

This property is commonly set using a task sequence step created using the **Create Virtual Hard Disk (VHD)** task sequence type. You can override the value that the **Create Virtual Hard Disk (VHD)** task sequence step sets by configuring this property in CustomSettings.ini.

> ⓘ Note
>
> To configure this property in CustomSettings.ini, you must add this property to the **Properties** line in CustomSettings.ini.

For related properties that are used with VHD files, see:

- **VHDCreateDiffVHD**
- **VHDCreateFileName**
- **VHDCreateSizeMax**
- **VHDCreateSource**
- **VHDCreateType**
- **VHDDrives**
- **VHDOutputVariable**
- **VHDTargetDisk**

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		

Component	Configured By		Scenario	Property Is Applicable
MDT DB	❌	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
variable	Variable that contains the drive letter on the target computer where the VHD files will be created. MDT creates the VHD files in the VHD folder in the root of this drive. For example, if this property has a value of **VHDTargetDisk**, the **VHDTargetDisk** property contains the drive letter (such as *H*).

Expand table

Example
`VHDCreateSizeMax=130048 VHDCreateType=EXPANDABLE VHDCreateFileName=Win7_C.vhd VHDInputVariable=VHDTargetDisk`

VHDOutputVariable

This property contains a variable that contains the physical drive number that was assigned to the newly created VHD file. Each time a VHD file is created, MDT sets this property to the disk index of the newly created disk using the **Index** property of the **Win32_DiskDrive** WMI class.

This property is commonly set using a task sequence step created using the **Create Virtual Hard Disk (VHD)** task sequence type. You can override the value that the **Create Virtual Hard Disk (VHD)** task sequence step sets by configuring this property in CustomSettings.ini.

> ⓘ Note
>
> To configure this property in CustomSettings.ini, you must add this property to the **Properties** line in CustomSettings.ini.

For related properties that are used with VHD files, see:

- **VHDCreateDiffVHD**
- **VHDCreateFileName**

- VHDCreateSizeMax
- VHDCreateSource
- VHDCreateType
- VHDDisks
- VHDInputVariable
- VHDTargetDisk

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	❌	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
Variable	Variable will contains the physical drive number assigned to the newly created VHD file. For example, if this property has a value of **OSDDiskIndex**, the **OSDDiskIndex** property will contain the physical drive number assigned to the newly created VHD file (such as *4*).

Expand table

Example
None

VHDTargetDisk

Specifies the drive on the target computer where the VHD is to be created. This property is later referenced in the VHDInputVariable property.

> ⓘ Note

This property is dynamically set by the MDT scripts and is not configured in CustomSettings.ini or the MDT DB. Treat this property as read only.

For related properties that are used with VHD files, see:

- **VHDCreateDiffVHD**
- **VHDCreateFileName**
- **VHDCreateSizeMax**
- **VHDCreateSource**
- **VHDCreateType**
- **VHDDisks**
- **VHDInputVariable**
- **VHDOutputVariable**

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	❌	|		
MDT DB	❌	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
Disk	Specifies the drive where the VHD is to be created

Expand table

Example
None

VMHost

Specifies the name of the Hyper-V host running the VM where MDT is running. This property is available only when the Hyper-V Integration Components are installed and running.

> **Note**
>
> This property is dynamically set by the MDT scripts and is not configured in CustomSettings.ini or the MDT DB. Treat this property as read only.

Table 4 lists the Windows operating systems that MDT supports and their corresponding Hyper-V Integration Components support.

Table 4. Windows Operating Systems and Hyper-V Integration Components Support

Expand table

Operating system	Hyper-V Integration Components
Windows PE	Integration Components are unavailable.
Windows 7	Available by default in Enterprise, Ultimate, and Professional editions.
Windows Server 2008 R2	Available by default in all editions.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	❌	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
Name	The name of the Hyper-V host running the VM where MDT is running

Expand table

Example
None

VMName

Specifies the name of the VM where MDT is running. This property is only available when the Hyper-V Integration Components are installed and running.

Table 5 lists the Windows operating systems supported by MDT and their corresponding Hyper-V Integration Components support.

> **Note**
>
> This property is dynamically set by the MDT scripts and is not configured in CustomSettings.ini or the MDT DB. Treat this property as read only.

Table 5. Windows Operating Systems and Hyper-V Integration Components Support

Expand table

Operating system	Hyper-V Integration Components
Windows PE	Integration Components are unavailable.
Windows 7	Available by default in Enterprise, Ultimate, and Professional editions.
Windows Server 2008 R2	Available by default in all editions.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	❌	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
name	The name of the VM where MDT is running

Expand table

Example
None

VMPlatform

Specifies specific information about the virtualization environment for the target computer when the target computer is a VM. The VM platform is determined by using WMI.

> **Note**
>
> This property is dynamically set by the MDT scripts and is not configured in CustomSettings.ini or the MDT DB. Treat this property as read only.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	❌	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
Hyper-V	Hyper-V
VirtualBox	Virtual Box
VMware	VMware virtualization platform
Xen	Citrix Xen Server

Expand table

Example
None

VRefresh

The vertical refresh rate for the monitor on the target computer. The vertical refresh rate is specified in Hertz. In the example, the value **60** indicates that the vertical refresh rate of the monitor is 60 Hz. This value is inserted into the appropriate configuration settings in Unattend.xml.

> **Note**
>
> The default values (in the Unattend.xml template file) are 1,024 pixels horizontal resolution, 768 pixels vertical resolution, 32-bit color depth, and 60 Hz vertical refresh rate.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
refresh_rate	The vertical refresh rate for the monitor on the target computer in Hertz

Expand table

Example
`[Settings] Priority=Default [Default] BitsPerPel=32 VRefresh=60 XResolution=1024 YResolution=768`

VSSMaxSize

This property is used to pass a value to the **maxsize** parameter of the **vssadmin resize shadowstorage** command in the **Vssadmin** command. The **maxsize** parameter is used to specify the maximum amount of space on the target volume that can be used for storing shadow copies. For more information on the **maxsize** parameter, see Vssadmin resize shadowstorage.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description

maxsize_value	Specifies the maximum amount of space that can be used for storing shadow copies. The value can be specified in bytes or as a percentage of the target volume.
	
	To specify the value:
	
	- In bytes, the value must be 300 MB or greater and accept the following suffixes: KB, MB, GB, TB, PB and EB. You can also use B, K, M, G, T, P, and E as suffixes—for example:
	
	`VSSMaxSize=60G`
	
	- As a percentage, use the % character as the suffix to the numeric value—for example:
	
	`VSSMaxSize=20%`
	
	Note:
	
	If a suffix is not supplied, the default suffix is bytes. For example, `VSSMaxSize=1024` indicates that the **VSSMaxSize** will be set to 1,024 bytes.
	
	If the value is set to **UNBOUNDED**, then there is no limit placed on the amount of storage space that can be used—for example:
	
	`VSSMaxSize=UNBOUNDED`

Expand table

Example
`[Settings] Priority=Default [Default] VSSMaxSize=25%`

WDSServer

The computer running Windows Deployment Services that is used for installing Windows Deployment Services images. The default value is the server running Windows Deployment Services from which the image was initiated.

> **Note**
>
> This property is dynamically set by the MDT scripts and is not configured in CustomSettings.ini or the MDT DB. Treat this property as read only.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	❌	|		
MDT DB	❌	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
WDS_server	The name of the computer running Windows Deployment Services

Expand table

Example
None

WindowsSource

MDT uses this property to set the location of the sources\sxs folder in a network shared folder that contains the operating system source files. This property is used when:

- MDT is running a custom task sequence or deploying a custom image
- MDT is installing roles or features in Windows 8 and Windows Server 2012
- The computer does not have access to the Internet

  When the situation described in the bulleted list above occurs, MDT may be unable to find the operating system source files locally, and the installation will attempt to download the files from the Internet. Because the computer does not have Internet access, the process will fail. Setting this property to the appropriate value helps prevent this problem from occurring.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	❌	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
folder_unc	A UNC path to the Sources\sxs folder for the operating system being deployed. Note: The UNC path must include the Sources\sxs folder.

Expand table

Example
`[Settings] Priority=Default [Default] WindowsSource=%DeployRoot%\Operating Systems\Windows 8\Sources\sxs`

WipeDisk

Specifies whether the disk should be wiped. If **WipeDisk** is TRUE, the ZTIWipeDisk.wsf script will clean the disk using the **Format** command. The **Format** command is not the most "secure" way of wiping the disk.

Securely wiping the disk should be done so in a manner that follows the U.S. Department of Defense standard 5220.22-M, which states, "To clear magnetic disks, overwrite all locations three times (first time with a character, second time with its complement, and the third time with a random character)."

When MDT wipes the disk, it uses the **Format** command with the **/P:3** switch, which instructs **Format** to zero every sector on the volume and to perform the operation three times. There is no way to tell the **Format** command to use a particular character or a random character.

> **Note**
>
> If the disk must be securely wiped, a non-Microsoft secure disk wipe tool should be added to the task sequence using the **Run Command Line** task sequence step.

> **Caution**
>
> This property value must be specified in uppercase letters so that the deployment scripts can properly read it.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
TRUE	If **WipeDisk** is set to **TRUE**, the Win32_DiskPartition at DiskIndex 0 and Index 0 will be formatted.
FALSE	The disk will not be formatted.

Expand table

Example
`[Settings] Priority=Default [Default] WipeDisk=TRUE`

WizardSelectionProfile

Profile name used by the wizard for filtering the display of various items.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
profile_name	Profile name used by the wizard for filtering the display of various items

Expand table

Example
`[Settings] Priority=Default [Default] WizardSelectionProfile=SelectTaskSequenceOnly`

WSUSServer

This is the name of the Windows Server Update Services (WSUS) server that the target computer should use when scanning for, downloading, and installing updates.

For more information about what script uses this property, see ZTIWindowsUpdate.wsf.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		

Component	Configured By		Scenario	Property Is Applicable
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
server_name	The name of the WSUS server, specified in HTTP format

Expand table

Example
`[Settings] Priority=Default [Default] WSUSServer=https://WSUSServerName[Settings] Priority=Default [Default] WSUSServer=https://WSUSServerName`

WUMU_ExcludeKB

The list of Windows Update/Microsoft Update software updates to ignore (by associated Knowledge Base articles).

Deployment project team members will want to periodically review the list of updates being installed by the ZTIWindowsUpdate.wsf script to verify that each update meets the project's needs and expectations. All updates are logged and recorded in the ZTIWindowsUpdate.log file, which is generated during deployment. Each update will indicate its status as INSTALL or SKIP and lists the UpdateID, the update name, and the QNumber associated with each update. If an update needs to be excluded, that update should be added to the CustomSettings.ini file (for LTI deployments).

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	❌	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
WUMU_ExcludeKB	The list of Windows Update/Microsoft Update software updates to ignore by QNumber

Expand table

Example
`[Settings] Priority=Default [Default] WUMU_ExcludeKB1=925471`

WUMU_ExcludeID

The list of Windows Update/Microsoft Update software updates to ignore (by associated update ID).

Deployment project team members will want to periodically review the list of updates being installed by the ZTIWindowsUpdate.wsf script to verify that each update meets the project's needs and expectations. All updates are logged and recorded in the ZTIWindowsUpdate.log file, which is generated during deployment. Each update will indicate its status as INSTALL or SKIP and lists the UpdateID, the update name, and the QNumber associated with each update. If an update should be excluded, that update should be added to the CustomSettings.ini file (for LTI deployments).

For example, if the installation of the Windows Malicious Software Removal Tool should be excluded, look up the line in the ZTIWindowsUpdate.log that shows where the update was identified and installed, and then select the UpdateID number. For example, the UpdateID number for the Windows Malicious Software Removal Tool is adbe6425-6560-4d40-9478-1e35b3cdab4f.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	❌	|	ZTI (Configuration Manager)	❌

Expand table

Value	Description
WUMU_ExcludeID	The list of Windows Update/Microsoft Update software updates to ignore, by UpdateID number

Expand table

Example
`[Settings] Priority=Default [Default] WUMU_ExcludeID1={adbe6425-6560-4d40-9478-1e35b3cdab4f}`
`[Settings] Priority=Default [Default] WUMU_ExcludeID1={adbe6425-6560-4d40-9478-1e35b3cdab4f}`

XResolution

The horizontal resolution of the monitor on the target computer, specified in pixels. In the example, the value **1024** indicates the horizontal resolution of the monitor is 1,024 pixels. This value is inserted into the appropriate configuration settings in Unattend.xml.

> **Note**
>
> The default values (in the Unattend.xml template file) are 1,024 pixels horizontal resolution, 768 pixels vertical resolution, 32-bit color depth, and 60 Hz vertical refresh rate.

Expand table

Component	Configured By	|	Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
horizontal_resolution	The horizontal resolution of the monitor on the target computer in pixels

Expand table

Example
`[Settings] Priority=Default [Default] BitsPerPel=32 VRefresh=60 XResolution=1024 YResolution=768[Settings] Priority=Default [Default] BitsPerPel=32 VRefresh=60 XResolution=1024 YResolution=768`

YResolution

The vertical resolution of the monitor on the target computer, specified in pixels. In the example, the value **768** indicates the vertical resolution of the monitor is 768 pixels. This value gets inserted into the appropriate configuration settings in Unattend.xml.

> **Note**
>
> The default values (in the Unattend.xml template file) are 1,024 pixels horizontal resolution, 768 pixels vertical resolution, 32-bit color depth, and 60 Hz vertical refresh rate.

Expand table

Component	Configured By		Scenario	Property Is Applicable
BootStrap.ini	❌	|	LTI (Stand-alone MDT)	✅
CustomSettings.ini	✅	|		
MDT DB	✅	|	ZTI (Configuration Manager)	✅

Expand table

Value	Description
vertical_resolution	The vertical resolution of the monitor on the target computer in pixels

Expand table

Example
`[Settings] Priority=Default [Default] BitsPerPel=32 VRefresh=60 XResolution=1024 YResolution=768[Settings] Priority=Default [Default] BitsPerPel=32 VRefresh=60 XResolution=1024 YResolution=768`

Providing Properties for Skipped Deployment Wizard Pages

Table 6 lists the individual Deployment Wizard pages, the property to skip the corresponding wizard page, and the properties that must be configured when skipping the wizard page.

If the **SkipWizard** property is used to skip all the Deployment Wizard pages, provide all the properties in the **Configure these properties** column. For examples of various deployment scenarios that skip Deployment Wizard pages, see the section, "Fully Automated LTI Deployment Scenario", in the MDT document *Microsoft Deployment Toolkit Samples Guide*.

> **Note**
>
> In instances where the **Configure These Properties** column is blank, no properties need to be configured when skipping the corresponding wizard page.

Table 6. Deployment Wizard Pages

Expand table

Skip this wizard page	Using this property	Configure these properties
Welcome	SkipBDDWelcome	
Specify credentials for connecting to network shares	Skipped by providing properties in next column	- UserID - UserDomain - UserPassword
Task Sequence	SkipTaskSequence	- TaskSequenceID
Computer Details	SkipComputerName, SkipDomainMembership	- OSDComputerName - JoinWorkgroup -or- - JoinDomain - DomainAdmin

Skip this wizard page	Using this property	Configure these properties
User Data	SkipUserData	- UDDir - UDShare - UserDataLocation
Move Data and Settings	SkipUserData	- UDDir - UDShare - UserDataLocation
User Data (Restore)	SkipUserData	- UDDir - UDShare - UserDataLocation
Computer Backup	SkipComputerBackup	- BackupDir - BackupShare - ComputerBackupLocation
Product Key	SkipProductKey	- ProductKey -or- - OverrideProductKey
Language Packs	SkipPackageDisplay	LanguagePacks
Locale and Time	SkipLocaleSelection, SkipTimeZone	- KeyboardLocale - UserLocale - UILanguage - TimeZoneName
Roles and Features	SkipRoles	- OSRoles - OSRoleServices - OSFeatures
Applications	SkipApplications	Applications
Administrator Password	SkipAdminPassword	AdminPassword

Skip this wizard page	Using this property	Configure these properties
Local Administrators	SkipAdminAccounts	- Administrators
Capture Image	SkipCapture	- ComputerBackupLocation
Bitlocker	SkipBitLocker	- BDEDriveLetter - BDEDriveSize - BDEInstall - BDEInstallSuppress - BDERecoveryKey - TPMOwnerPassword - OSDBitLockerStartupKeyDrive - OSDBitLockerWaitForEncryption
Ready to begin	SkipSummary	-
Operating system deployment completed successfully	SkipFinalSummary	-
Operating system deployment did not complete successfully	SkipFinalSummary	-

Related articles

- Task Sequence Steps.
- Scripts.
- Support Files.
- Utilities.
- MDT Windows PowerShell Cmdlets.
- Tables and Views in the MDT DB.
- Windows 7 Feature Dependency Reference.
- UDI Reference.

Feedback

Was this page helpful? Yes No

Provide product feedback

Scripts

Article • 02/12/2024

The scripts used in LTI and ZTI deployments reference properties that determine the process steps and configuration settings used during the deployment process. Use this reference section to help it determine the correct scripts to include in actions and the valid arguments to provide when running each script. The following information is provided for each script:

- **Name**.Specifies the name of the script.
- **Description**.Provides a description of the purpose of the script and any pertinent information regarding script customization.
- **Input**. Indicates the files used for input to the script.
- **Output**.Indicates the files created or modified by the script.
- **References**.Indicates other scripts or configuration files that are referenced by the script.
- **Location**.Indicates the folder where the script can be found. In the information for the location, the following variables are used:
    - **program_files**. This variable points to the location of the Program Files folder on the computer where MDT is installed.
    - **distribution**. This variable points to the location of the Distribution folder for the deployment share.
    - **platform**. This variable is a placeholder for the operating system platform (x86 or x64).
- **Use**.Provides the commands and options that you can specify.
- **Arguments and description**. Indicate the valid arguments to be specified for the script and a brief description of what each argument means.
- **Properties**.The properties referenced by the script.

BDD_Autorun.wsf

This script displays a dialog box that indicates the user inserted deployment media created by the MDT process (such as a bootable DVD or a removable hard disk). The

message is displayed for 15 seconds. If no action is taken, the script starts LiteTouch.vbs.

For more information about LiteTouch.vbs, see LiteTouch.vbs.

Expand table

Value	Description
Input	**Environment variables**. Contains the property values, custom property values, database connections, deployment rules, and other information required by the scripts to complete the deployment process
Output	None
References	**LiteTouch.vbs**. Initiates LTI
Location	*distribution*\Scripts
Use	None

Arguments

Expand table

Value	Description
None	None

Properties

Expand table

Name	Read	Write
None		

BDD_Welcome_ENU.xml

This XML file contains the script code and HTML layout for the **Welcome to Windows Deployment** page that is displayed at the start of the Deployment Wizard. This XML file is read by Wizard.hta, which runs the wizard pages embedded in this XML file.

Expand table

Value	Description
Input	None
Output	None
References	- **NICSettings_Definition_ENU.xml**. Allows the user to provide configuration settings for network adapters - **Wizard.hta**. Displays the Deployment Wizard pages - **WPEUtil.exe**. Initializes Windows PE and network connections; initiates LTI
Location	*distribution*\Tools*platform*
Use	`mshta.exeWizard.hta BDD_Welcome_ENU.xml`

Arguments

Expand table

Value	Description
None	None

Properties

Expand table

Name	Read	Write
KeyboardLocalePE	-	
WelcomeWizardCommand		-
WizardComplete		-

Credentials_ENU.xml

This XML file contains the script code and HTML layout for the **Specify credentials for connecting to network shares** wizard page in the Deployment Wizard. This XML file is read by Wizard.hta, which runs the wizard pages embedded in this XML file.

> **Note**

This wizard page is only displayed if there is a failure while validating the predefined user credentials.

Expand table

Value	Description
Input	None
Output	None
References	**Credentials_scripts.vbs**. Contains user credential support functions
Location	*distribution*\Scripts
Use	`mshta.exe Wizard.hta /NotWizard /definition:Credentials_ENU.xml [/ValidateAgainstDomain:domain	/ValidateAgainstUNCPath:uncpath] </DoNotSave> </LeaveShareOpen>`

Arguments

Expand table

Value	Description
None	None

Properties

Expand table

Name	Read	Write
None		

Credentials_scripts.vbs

This script parses the arguments that were provided when loading the Credentials_ENU.xml file into the Deployment Wizard. It also performs user credential validation. This script is read by the Credentials_ENU.xml file.

For more information about Credentials_ENU.xml, see the corresponding topic in Credentials_ENU.xml.

Expand table

Value	Description
Input	None
Output	Event messages are written to these log files: - **Credentials_scripts.log**. Log file that contains events generated by this script - **BDD.log**. Log file that contains events generated by all MDT scripts
References	None
Location	*distribution*\Scripts
Use	`<script language="VBScript" src="Credentials_scripts.vbs"/>`

Arguments

Expand table

Value	Description
None	None

Properties

Expand table

Name	Read	Write
UserCredentials		-
UserDomain	-	

DeployWiz_Definition_ENU.xml

This XML file contains the script code and HTML layout for each wizard page in the Deployment Wizard. This file is read by Wizard.hta, which runs the wizard pages embedded in this XML file. This .xml file contains the following wizard pages:

- **Welcome**
- **Specify credentials for connecting to network shares**

- Task Sequence
- Computer Details
- User Data
- Move Data and Settings
- User Data (Restore)
- Computer Backup
- Product Key
- Language Packs
- Locale and Time
- Roles and Features
- Applications
- Administrator Password
- Local Administrators
- Capture Image
- BitLocker
- Ready to Begin

Expand table

Value	Description
Input	None
Output	None
References	- **DeployWiz_Initialization.vbs**. Includes support functions and subroutines used by the script - **DeployWiz_Validation.vbs**. Includes support functions and subroutines used by the script - **ZTIBackup.wsf**. Creates a backup of the target computer - **ZTIPatches.wsf**. Installs updates (language packs, security updates, and so on)

Value	Description
	- **ZTIUserState.wsf**. Initializes user state migration to capture and restore user state on the target computer
Location	*distribution*\Scripts
Use	None

Arguments

Expand table

Value	Description
None	None

Properties

Expand table

Name	Read	Write
DeploymentMethod	-	
DeploymentType	-	
DoCapture	-	
ImageBuild	-	
ImageFlags	-	
IsBDE	-	
IsServerOS	-	
JoinDomain	-	
OSDComputerName	-	
OSVersion	-	
SkipAdminAccounts	-	
SkipAdminPassword	-	
SkipApplications	-	

Name	Read	Write
SkipBitLocker	-	
SkipCapture	-	
SkipComputerBackup	-	
SkipComputerName	-	
SkipDomainMembership	-	
SkipLocaleSelection	-	
SkipPackageDisplay	-	
SkipProductKey	-	
SkipRoles	-	
SkipSummary	-	
SkipTaskSequence	-	
SkipTimeZone	-	
SkipUserData	-	
TaskSequenceTemplate	-	
UserDomain	-	
UserID	-	
UserPassword	-	
USMTOfflineMigration	-	

DeployWiz_Initialization.vbs

This script initializes the pages in the **Deployment Wizard** (stored in DeployWiz_Definition_ENU.xml). It also contains functions and subroutines that the Deployment Wizard calls during an LTI deployment.

Expand table

Value	Description
Input	- **DomainOUList.xml**. Contains a list of domain OUs

Value	Description
	- **ListOfLanguages.xml** - **LocationServer.xml**. Contains a list of available deployment shares - **Environment variables**. Contains the list of property values, custom properties, database connections, deployment rules, and other information that the scripts require to complete the deployment process; the environment variables are populated by ZTIGather.wsf
Output	Event message are written to these log files: - **DeployWiz_Initialization.log**. Log file that contains events generated by this script - **BDD.log**. Log file that contains events generated by all MDT scripts
References	**ZTIApplications.wsf**. Initiates application installation
Location	*distribution*\Scripts
Use	`<script language="VBScript" src="DeployWiz_Initialization.vbs"/>`

Arguments

Expand table

Value	Description
None	None

Properties

Expand table

Name	Read	Write
Architecture	-	
Applications	-	
BackupDir	-	
BackupFile	-	
BackupShare	-	
BDEInstall	-	

Name	Read	Write
BDEKeyLocation	-	
BDERecoveryKey	-	
BDEWaitForEncryption	-	
CapableArchitecture	-	
ComputerBackupLocation	-	
CustomWizardSelectionProfile	-	
DeploymentType	-	
DeployRoot	-	
DomainAdmin	-	
DomainAdminDomain	-	
DomainAdminPassword	-	
DomainOUs	-	
ImageBuild	-	
ImageFlags	-	
ImageLanguage	-	
ImageLanguage001	-	
ImageProcessor	-	
IsServerOS	-	
KeyboardLocale	-	
KeyboardLocale_Edit	-	
LanguagePacks	-	
LanguagePacks001	-	
LocalDeployRoot	-	
MandatoryApplications	-	
OSDComputerName	-	
OSCurrentBuild	-	

Name	Read	Write
OSDBitLockerCreateRecoveryPassword	-	
OSDBitLockerMode	-	
OSDBitLockerStartupKeyDrive	-	
OSDBitLockerWaitForEncryption	-	
OSSKU	-	
OSVersion	-	
OverrideProductKey	-	
ProductKey	-	
SkipCapture	-	
SkipDomainMembership	-	
TaskSequenceID	-	
TimeZoneName	-	
TSGUID	-	
UDDir	-	
UDShare	-	
UILanguage	-	
UserDataLocation	-	
UserDomain	-	
UserID	-	
UserLocale	-	
UserPassword	-	
WizardSelectionProfile	-	

DeployWiz_Validation.vbs

This script initializes and validates the information typed in the pages of the Deployment Wizard (stored in DeployWiz_Definition_ENU.xml). This script contains functions and subroutines that the Deployment Wizard calls during an LTI deployment.

Expand table

Value	Description
Input	- **OperatingSystems.xml**. Contains the list of operating systems available for deployment - **Environment variables**. Contains the list of property values, custom properties, database connections, deployment rules, and other information required by the scripts to complete the deployment process; the environment variables are populated by ZTIGather.wsf
Output	None
References	- **Credentials_ENU.xml**. Prompts the user for credentials that will be used when connecting to network resources - **ZTIGather.wsf**. Gathers properties and processing rules
Location	*distribution*\Scripts
Use	`<script language="VBScript" src="DeployWiz_Validation.vbs"/>`

Arguments

Expand table

Value	Description
None	None

Properties

Expand table

Name	Read	Write
Architecture	-	
DeploymentType	-	-
DeployTemplate		-
ImageBuild	-	
ImageProcessor	-	-

Name	Read	Write
OSVersion	-	
TaskSequenceID		-
TSGUID	-	
UserCredentials	-	
UserDomain		-
UserID		-
UserPassword		-

LiteTouch.vbs

This script is called by the Deployment Wizard to initiate LTI. The script:

- Removes the C:\MININT folder (if it exists)
- Checks that the target computer meets the requirements for running the Deployment Wizard by calling ZTIPrereq.vbs
- Starts the Deployment Wizard by running LiteTouch.wsf

Expand table

Value	Description
Input	None
Output	None
References	- **BDDRun.exe** - **ZTIPrereq.vbs**. Used to determine whether the target computer meets the prerequisites for deploying a new operating system - **LiteTouch.wsf**. The script responsible for controlling the LTI deployment process
Location	*distribution*\Scripts
Use	`cscript LiteTouch.vbs </debug:value>`

Arguments

Expand table

Value	Description
/debug:*value*	Outputs the event messages to the console and to the .log files. If the value specified in value is: - **TRUE**, event messages are sent to the console and the .log files - **FALSE**, event messages are sent only to the .log files (this is the behavior when the argument is not provided)

Properties

Expand table

Name	Read	Write
None		

LiteTouch.wsf

This script is called by LiteTouch.vbs and is responsible for controlling the LTI deployment process. This includes:

- Running the Deployment Wizard
- Running the LTI deployment process by using the appropriate task sequence file

Expand table

Value	Description
Input	- ***task_sequence_file*.xml**. Contains the tasks and sequence of tasks for the LTI deployment process - **Environment variables**. Contains the list of property values, custom properties, database connections, deployment rules, and other information required by the scripts to complete the deployment process; the environment variables are populated by ZTIGather.wsf
Output	- **LiteTouch.log**. Log file that contains events that this script generates - **BDD.log**. Log file that contains events that all MDT scripts generate

Value	Description
References	- **BDD_Welcome_ENU.xml**. Displays the Deployment Wizard **Welcome** page for LTI deployment - **DeployWiz_Definition_ENU.xml**. Displays the Deployment Wizard pages for LTI deployment - **Diskpart.exe**. Utility that allows the automated management of disks, partitions, and volumes - **LTICleanup.wsf**. Performs cleanup tasks after deployment finishes - **LTICopyScripts.wsf**. Copies the deployment scripts to a local hard drive on the target computer - **MSHTA.exe**. HTML application host - **RecEnv.exe**. If this utility exists, the user is prompted to determine whether to launch Windows Recovery Environment. - **Regsvr32.exe**. Registers files (.dll, .exe, .ocx, and so on) with the operating system - **Summary_Definition_ENU.xml**. Displays the summary results for the LTI deployment - **TsmBootStrap.exe**. Task sequence Bootstrap utility - **Wizard.hta**. Displays the Deployment Wizard pages - **WPEUtil.exe**. Initializes Windows PE and network connections; initiates LTI - **ZTIGather.wsf**. Gathers properties and processing rules - **ZTIPrereq.vbs**. Checks that the target computer meets the requirements for running the Deployment Wizard - **ZTINICConfig.wsf**. Configures activated network adapters - **ZTIUtility.vbs**. Includes support functions and subroutines the script uses
Location	*distribution*\Scripts
Use	`BDDRun.exe "wscript.exe <ScriptDirectory>\LiteTouch.wsf </debug:value>"`

Arguments

Expand table

Value	Description
/debug:*value*	Outputs the event messages to the console and to the .log files. If the value specified in value is: - **TRUE**, event messages are sent to the console and the .log files - **FALSE**, event messages are sent only to the .log files (this is the behavior when the argument is not provided)
/Start	Creates a shortcut in the new operating system that runs once the shell starts

Properties

Expand table

Name	Read	Write
_DoNotCleanLiteTouch	-	
_SMSTSPackageName		-
AdminPassword	-	
Architecture	-	-
BootPE	-	-
ComputerBackupLocation		-
ComputerName	-	
DeployDrive	-	-
DeploymentMethod	-	-
DeploymentType	-	-
DeployRoot	-	-
DestinationLogicalDrive		-
DomainAdmin		-
DomainAdminDomain		-
DomainAdminPassword		-

Name	Read	Write
FinishAction	-	
HostName	-	
IsServerCoreOS	-	
JoinDomain	-	
JoinWorkgroup	-	-
KeyboardLocalePE	-	
LTISuspend	-	
OSDAdapterCount	-	
OSDComputerName	-	-
Phase	-	-
ResourceDrive	-	-
ResourceRoot	-	-
RetVal		-
SkipBDDWelcome	-	
SkipFinalSummary	-	-
SkipWizard	-	
SMSTSLocalDataDrive		-
TaskSequenceID	-	
TimeZoneName		-
UserDataLocation	-	-
UserDomain	-	
UserID	-	
UserPassword	-	
WelcomeWizardCommand	-	
WizardComplete	-	

LTIApply.wsf

This script is responsible for installing a Windows PE image to the target computer. The Windows PE image is used to collect information about the target computer and to run the deployment tasks on the target computer.

Expand table

Value	Description
Input	**Environment variables**. Contains the property values, custom property values, database connections, deployment rules, and other information the scripts require to complete the deployment process
Output	- **LTIApply.log**. Log file that contains events that this script generates - **LTIApply_wdsmcast.log**. Log file that contains events that the Wdsmcast utility generates - **BDD.log**. Log file that contains events that all MDT scripts generate
References	- **CMD.exe**. Allows the running of command-line tools - **Bootsect.exe**. Applies a boot sector to the hard disk - **ImageX.exe**. A utility used to create and manage WIM files - **ZTIBCDUtility.vbs**. Includes utility functions used when performing Boot Manager tasks - **ZTIConfigFile.vbs**. Includes routines for processing XML files - **ZTIDiskUtility.vbs**. Includes support functions and subroutines the script uses - **ZTIUtility.vbs**. Includes support functions and subroutines the script uses - **Wdsmcast.exe**. A utility that target computers use to join a multicast transmission
Location	*distribution*\Scripts
Use	`cscript LTIApply.wsf </pe> </post> </debug:value>`

Arguments

Expand table

Value	Description
/pe	Uses the process for installing the Windows PE image on the target computer
/post	Cleans up unnecessary files after the installation of an image
/debug:*value*	Outputs the event messages to the console and to the .log files; if the value specified in value is: - **TRUE**, event messages are sent to the console and the .log files - **FALSE**, event messages are sent only to the .log files (this is the behavior when the argument is not provided)

Properties

Expand table

Name	Read	Write
Architecture	-	
BootPE		-
DeployRoot	-	
DestinationLogicalDrive	-	-
OSGUID	-	
OSCurrentVersion	-	
OSVersion	-	
ImageBuild	-	
ImageFlags	-	
ImageProcessor	-	
ISBDE	-	
SourcePath		-
TaskSequenceID	-	
UserDomain	-	
UserID	-	
UserPassword	-	

Name	Read	Write
WDSServer	-	

LTICleanup.wsf

This script removes any files or configuration settings (such as scripts, folders, registry entries, or automatic logon configuration settings) from the target computer after the deployment process finishes.

Expand table

Value	Description
Input	**Environment variables**. Contains the list of property values, custom properties, database connections, deployment rules, and other information that the scripts require to complete the deployment process. The environment variables are populated by ZTIGather.wsf.
Output	- **LTICleanup.log**. Log file that contains events that this script generates - **BDD.log**. Log file that contains events that all MDT scripts generate
References	- **Bootsect.exe**. Applies a boot sector to the hard disk - **Net.exe**. Performs network management tasks - **RegSvr32.exe**. Registers files (.dll, .exe, .ocx, and so on) with the operating system - **ZTIBCDUtility.vbs**. Includes utility functions used when performing Boot Manager tasks - **ZTIUtility.vbs**. Includes support functions and subroutines the script uses
Location	*distribution*\Scripts
Use	`cscript LTICleanup.wsf </debug:value>`

Arguments

Expand table

Value	Description
/debug:*value*	Outputs the event messages to the console and to the .log files. If the value specified in value is:

Value	Description
	- **TRUE**, event messages are sent to the console and the .log files - **FALSE**, event messages are sent only to the .log files (this is the behavior when the argument is not provided)

Properties

Expand table

Name	Read	Write
_DoNotCleanLiteTouch	-	
DeployRoot	-	
DestinationLogicalDrive	-	
OSVersion	-	

LTICopyScripts.wsf

This script copies the deployment scripts for the LTI and ZTI deployment processes to a local hard drive on the target computer.

Expand table

Value	Description
Input	- **Summary_Definition_ENU.xml**. Displays the summary results for the LTI deployment - **Environment variables**. Contains the property values, custom property values, database connections, deployment rules, and other information that the scripts require to complete the deployment process
Output	- **LTICopyScripts.log**. Log file that contains events that this script generates - **BDD.log**. Log file that contains events that all MDT scripts generate
References	**ZTIUtility.vbs**. Includes support functions and subroutines the script uses
Location	*distribution*\Scripts
Use	`cscript LTICopyScripts.wsf </debug:value>`

Arguments

Expand table

Value	Description
/debug:*value*	Outputs the event messages to the console and to the .log files. If the value specified in value is: - **TRUE**, event messages are sent to the console and the .log files - **FALSE**, event messages are sent only to the .log files (this is the behavior when the argument is not provided)

Properties

Expand table

Name	Read	Write
None		

LTIGetFolder.wsf

This script displays a dialog box that allows the user to browses to a folder. The selected folder path is stored in the FOLDERPATH environment variable.

Expand table

Value	Description
Input	**Environment variables**. Contains the list of property values, custom properties, database connections, deployment rules, and other information that the scripts require to complete the deployment process. The environment variables are populated by ZTIGather.wsf.
Output	None
References	- **ZTIUtility.vbs**. Includes support functions and subroutines that the script uses - **WizUtility.vbs**. Includes support functions and subroutines that the UI uses (such as wizard pages)
Location	- *distribution*\Scripts

Value	Description
	- *program_files*\Microsoft Deployment Toolkit\Scripts
Use	`cscript LTIGetFolder.wsf </debug:value>`

Arguments

Expand table

Value	Description
/debug:value	Outputs the event messages to the console and to the .log files. If the value specified in value is: - **TRUE**, event messages are sent to the console and the .log files - **FALSE**, event messages are sent only to the .log files (this is the behavior when the argument is not provided)

Properties

Expand table

Name	Read	Write
DefaultFolderPath	-	
FolderPath		-

LTIOEM.wsf

This script is used by an OEM during an LTI OEM scenario to copy the contents of a media deployment share to the target computer's hard disk to prepare it for duplication.

Expand table

Value	Description
Input	**Environment variables**. Contains the list of property values, custom properties, database connections, deployment rules, and other information that the scripts require to complete the deployment process. The environment variables are populated by ZTIGather.wsf.

Value	Description
Output	- **LTIOEM.log**. Log file that contains events that this script generates - **BDD.log**. Log file that contains events that all MDT scripts generate
References	- **RoboCopy.exe**. File and folder copy tool - **ZTIUtility.vbs**. Includes support functions and subroutines that the script uses
Location	*distribution*\Scripts
Use	`cscript LTIOEM.wsf </BITLOCKER	/BDE> </debug:value>`

Arguments

Expand table

Value	Description
/debug:*value*	Outputs the event messages to the console and to the .log files. If the value specified in value is: - **TRUE**, event messages are sent to the console and the .log files - **FALSE**, event messages are sent only to the .log files (this is the behavior when the argument is not provided)
/BITLOCKER	Enables BitLocker
/BDE	Enables BitLocker

Properties

Expand table

Name	Read	Write
_DoNotCleanLiteTouch		-
DeployDrive	-	
DeployRoot	-	
TSGUID	-	

LTISuspend.wsf

This script suspends a task sequence to allow manual tasks to be performed. When this script runs, it creates a **Resume Task Sequence** shortcut on the user's desktop that allows the user to restart the task sequence after all manual tasks are completed.

> **Note**
>
> This script is only supported while in the full operating system.

Expand table

Value	Description
Input	**Environment variables**. Contains the list of property values, custom properties, database connections, deployment rules, and other information the scripts require to complete the deployment process. The environment variables are populated by ZTIGather.wsf.
Output	- **LTISuspend.log**. Log file that contains events that this script generates - **BDD.log**. Log file that contains events that all MDT scripts generate
References	- **LiteTouch.wsf**. Controls the LTI deployment process - **LTICopyScripts.wsf**. Copies the deployment scripts to a local hard drive on the target computer - **ZTIUtility.vbs**. Includes support functions and subroutines that the script uses
Location	*distribution*\Scripts
Use	`cscript LTISuspend.wsf </debug:value>`

Arguments

Expand table

Value	Description
/debug:*value*	Outputs the event messages to the console and to the .log files. If the value specified in value is: - **TRUE**, event messages are sent to the console and the .log files

Value	Description
	- **FALSE**, event messages are sent only to the .log files (this is the behavior when the argument is not provided)
/Resume	-

Properties

Expand table

Name	Read	Write
LTISuspend		-
SMSTSRebootRequested		-

LTISysprep.wsf

This script prepares the target computer for running Sysprep, runs Sysprep on the target computer, and then verifies that Sysprep ran successfully.

Expand table

Value	Description
Input	**Environment variables**. Contains the list of property values, custom properties, database connections, deployment rules, and other information that the scripts require to complete the deployment process. The environment variables are populated by ZTIGather.wsf.
Output	- **LTISysprep.log**. Log file that contains events that this script generates - **BDD.log**. Log file that contains events that all MDT scripts generate
References	- **Expand.exe**. Expands compressed files - **Sysprep.exe**. Prepares computers for duplication - **ZTIConfigFile.vbs**. Contains routines for processing XML files - **ZTIUtility.vbs**. Includes support functions and subroutines that the script uses
Location	*distribution*\Scripts
Use	`cscript LTISysprep.wsf </debug:value>`

Arguments

Expand table

Value	Description
/debug:*value*	Outputs the event messages to the console and to the .log files. If the value specified in value is: - **TRUE**, event messages are sent to the console and the .log files - **FALSE**, event messages are sent only to the .log files (This is the behavior when the argument is not provided.)

Properties

Expand table

Name	Read	Write
Architecture	-	
DeployRoot	-	
DestinationLogicalDrive	-	
DoCapture	-	
OSCurrentBuild	-	
OSDAnswerFilePath	-	
OSGUID	-	
SourcePath	-	-
TaskSequenceID	-	

NICSettings_Definition_ENU.xml

This XML file contains the script code and HTML layout for the **Configure Static IP Network Settings** wizard page in the Deployment Wizard. During an LTI deployment, Wizard.hta reads this file and runs the embedded wizard page that prompts for the required network addressing configuration. If no static IP addressing configuration is supplied, the deployment scripts will default to using DHCP to obtain the required network configuration.

Expand table

Value	Description
Input	None
Output	None
References	**ZTINICUtility.vbs**. Includes support functions and subroutines that the script uses
Location	*distribution*\Scripts
Use	None

Arguments

Expand table

Value	Description
None	None

Properties

Expand table

Name	Read	Write
OSDAdapterxDNSServerList		-
OSDAdapterxDNSSuffix		-
OSDAdapterxGateways		-
OSDAdapterxIPAddressList		-
OSDAdapterxMacAddress		-
OSDAdapterxSubnetMask		-
OSDAdapterxWINSServerList		-
OSDAdapterCount		-

Note

Thexin the property names listed above is a placeholder for a zero-based array that contains network adapter information.

Summary_Definition_ENU.xml

This XML file contains the script code and HTML layout for the **Deployment Summary** wizard page in the Deployment Wizard. During an LTI deployment, Wizard.hta reads this file and runs the embedded wizard page that displays the summary results for the LTI deployment. This XML file contains the following wizard pages:

- **Success**. Notification regarding the successful completion of the deployment tasks
- **Failure**. Notification regarding the failure to successfully complete the deployment tasks

Expand table

Value	Description
Input	None
Output	None
References	**Summary_Scripts.vbs**. Includes support functions and subroutines that the wizard pages embedded in this XML file use
Location	*distribution*\Scripts
Use	None

Arguments

Expand table

Value	Description
None	None

Properties

Expand table

Name	Read	Write
SkipFinalSummary	-	
RetVal	-	

Summary_scripts.vbs

This script is called by the **Summary** wizard page of the Deployment Wizard. It contains functions and subroutines used for initialization and validation.

Expand table

Value	Description
Input	**Environment variables**. Contains the property values, custom property values, database connections, deployment rules, and other information that the scripts require to complete the deployment process
Output	Event message are written to these log files: - **Summary_scripts.log**. Log file that contains events that this script generates - **BDD.log**. Log file that contains events that all MDT scripts generate
References	None
Location	*distribution*\Scripts
Use	`<script language="VBScript" src="Summary_Scripts.vbs"/>`

Arguments

Expand table

Value	Description
None	None

Properties

Expand table

Name	Read	Write
DeploymentType	-	
RetVal	-	

Wizard.hta

This Hypertext Application displays the Deployment Wizard pages.

Expand table

Value	Description
Input	**Environment variables**. Contains the list of property values, custom properties, database connections, deployment rules, and other information that the scripts require to complete the deployment process. The environment variables are populated by ZTIGather.wsf.
Output	- **Wizard.log**. Log file that contains events that this script generates - **BDD.log**. Log file that contains events that all MDT scripts generate
References	- **LTIGetFolder.wsf**. Script file that initiates a **BrowseForFolder** dialog box - **ZTIConfigFile.vbs**. Includes routines for processing XML files - **ZTIUtility.vbs**. Includes support functions and subroutines that the script uses - **WizUtility.vbs**. Includes support functions and subroutines that the script uses
Location	- *distribution*\Scripts - *program_files*\Microsoft Deployment Toolkit\Scripts
Use	`mshta.exe Wizard.hta </definition:filename> </NotWizard> </debug:value>`

Arguments

Expand table

Value	Description
/debug:*value*	Outputs the event messages to the console and to the .log files. If the value specified in value is: - **TRUE**, event messages are sent to the console and the .log files

Value	Description
	- **FALSE**, event messages are sent only to the .log files (This is the behavior when the argument is not provided.)
/NotWizard	Used to bypass wizard page prompts
/Definition:*filename*	Specifies the XML file that is to be loaded into the wizard

Properties

Expand table

Name	Read	Write
Definition	-	
DefaultFolderPath		-
FolderPath	-	
WizardComplete		-

WizUtility.vbs

This script contains functions and subroutines that the various Deployment Wizard scripts reference.

Expand table

Value	Description
Input	**Environment variables**. Contains the property values, custom property values, database connections, deployment rules, and other information that the scripts require to complete the deployment process
Output	- **WizUtility.log**. Log file that contains events that this script generates - **BDD.log**. Log file that contains events that all MDT scripts generate
References	**LTIGetFolder.wsf**. Script file that initiates a **BrowseForFolder**dialog box
Location	- *distribution*\Scripts - *program_files*\Microsoft Deployment Toolkit\Scripts

Value	Description
Use	`<script language="VBScript" src="WizUtility.vbs"/>`

Arguments

Expand table

Value	Description
None	None

Properties

Expand table

Name	Read	Write
DefaultFolderPath		-
DefaultDestinationDisk	-	
DefaultDestinationIsDirty	-	
DefaultDestinationPartition	-	
DeploymentType	-	
DestinationDisk	-	
FolderPath	-	
OSVersion	-	
UserDomain	-	
UserCredentials		-

ZTIApplications.wsf

This script initiates an installation of applications that have been configured in the Applications node in Deployment Workbench. This script will not attempt to install any application that:

- Does not support the target computer's platform type

- Does not support the target computer's processor type
- Has an uninstall entry in the registry under **HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall**

> **Note**
>
> If the listed application has any dependent applications defined, this script attempts to install those dependent applications before installing the listed application.

Expand table

Value	Description
Input	**Environment variables**. Contains the property values, custom property values, database connections, deployment rules, and other information that the scripts require to complete the deployment process
Output	- **ZTIApplications.log**. Log file that contains events that this script generates - **BDD.log**. Log file that contains events that all MDT scripts generate
References	- **ZTIConfigFile.vbs**. Includes routines for processing XML files - **ZTIUtility.vbs**. Includes support functions and subroutines that the script uses - **BDDRun.exe**. Runs a command that requires user interaction
Location	*distribution*\Scripts
Use	`cscript ZTIApplications.wsf </debug:value>`

Arguments

Expand table

Value	Description
/debug:*value*	Outputs the event messages to the console and to the .log files. If the value specified in value is: - **TRUE**, event messages are sent to the console and the .log files - **FALSE**, event messages are sent only to the .log files (this is the behavior when the argument is not provided)

Properties

Expand table

Name	Read	Write
ApplicationGUID	-	
ApplicationSuccessCodes	-	
DependentApplications	-	
DeploymentMethod	-	
InstalledApplications	-	-
ResourceDrive	-	
ResourceRoot	-	-
SMSTSRebootRequested		-
SMSTSRetryRequested		-

ZTIAppXmlGen.wsf

This script generates an XML file—ZTIAppXmlGen.xml—to use when automatically capturing user data (documents) associated with installed applications. It does so through the **HKEY_CLASSES_ROOT\Software\Classes** registry key and captures any applications that:

- Are not associated with one of these file extensions: .mp3, .mov, .wma, .wmv, .chm, .evt, .evtx, .exe, .com, or .fon
- Are not associated with Microsoft Office, such as the 2007 Office system or Microsoft Office 2003.
- Have a valid open handler listed at **HKEY_CLASSES_ROOT\application\shell\open\command**

Expand table

Value	Description
Input	**Environment variables**. Contains the property values, custom property values, database connections, deployment rules, and other information that the scripts require to complete the deployment process

Value	Description
Output	- **ZTIAppXmlGen.xml**.Contains a list of applications installed on the target computer - **ZTIAppXmlGen.log**. Log file that contains events that this script generates - **BDD.log**. Log file that contains events that all MDT scripts generate
References	**ZTIUtility.vbs**. Includes support functions and subroutines that the script uses
Location	*distribution*\Scripts
Use	`cscript ZTIAppXmlGen.wsf </debug:value>`

Arguments

Expand table

Value	Description
/debug:*value*	Outputs the event messages to the console and to the .log files. If the value specified in value is: - **TRUE**, event messages are sent to the console and the .log files - **FALSE**, event messages are sent only to the .log files (This is the behavior when the argument is not provided.)

Properties

Expand table

Name	Read	Write
DeploymentMethod	-	
DeploymentType	-	
ImageBuild	-	
OSCurrentVersion	-	
USMTMigFiles	-	-

ZTIAuthorizeDHCP.wsf

This script uses the Netsh tool to configure the target computer so that it is an authorized DHCP server in AD DS.

For more information about authorizing DHCP servers, see Netsh commands for DHCP.

Expand table

Value	Description
Input	**Environment variables**. Contains the property values, custom property values, database connections, deployment rules, and other information that the scripts require to complete the deployment process
Output	- **ZTIAuthorizeDHCP.log**. Log file that contains events that this script generates - **BDD.log**. Log file that contains events that all MDT scripts generate
References	- **Netsh.exe**. A utility used to automate the configuration of networking components - **ZTIUtility.vbs**. Includes support functions and subroutines that the script uses
Location	*distribution*\Scripts
Use	`cscript ZTIAuthorizeDHCP.wsf </debug:value>`

Arguments

Expand table

Value	Description
/debug:*value*	Outputs the event messages to the console and to the .log files. If the value specified in value is: - **TRUE**, event messages are sent to the console and the .log files - **FALSE**, event messages are sent only to the .log files (This is the behavior when the argument is not provided.)

Properties

Expand table

Name	Read	Write
IPAddress	-	

ZTIBackup.wsf

This script performs a backup of the target computer using the ImageX utility. The backup is stored in the location specified in the BackupDir and BackupShare properties.

Expand table

Value	Description
Input	**Environment variables**. Contains the property values, custom property values, database connections, deployment rules, and other information that the scripts require to complete the deployment process
Output	- **ZTIBackup.log**. Log file that contains events that this script generates - **ZTIBackup_imagex.log**. Log file that contains events that ImageX generates - **BDD.log**. Log file that contains events that all MDT scripts generate
References	- **ImageX.exe**. A utility used to create and manage WIM files - **ZTIBCDUtility.vbs**. Includes utility functions used when performing Boot Manager tasks - **ZTIDiskUtility.vbs**. Includes support functions and subroutines that the script uses - **ZTIUtility.vbs**. Includes support functions and subroutines that the script uses
Location	*distribution*\Scripts
Use	`cscript ZTIBackup.wsf </debug:value>`

Arguments

Expand table

Value	Description
/debug:*value*	Outputs the event messages to the console and to the .log files. If the value specified in value is: - **TRUE**, event messages are sent to the console and the .log files - **FALSE**, event messages are sent only to the .log files (this is the behavior when the argument is not provided)

Properties

Expand table

Name	Read	Write
BackupDir	-	
BackupDisk	-	
BackupDrive	-	
BackupFile	-	
BackupPartition	-	
BackupScriptComplete		-
BackupShare	-	
ComputerBackupLocation	-	
DeploymentMethod	-	
DeploymentType	-	
DestinationLogicalDrive	-	-
DoCapture	-	
ImageBuild	-	
ImageFlags	-	
OSDStateStorePath	-	
Phase	-	
TaskSequenceID	-	
USMTLocal	-	

ZTIBCDUtility.vbs

This script contains utility functions that some MDT scripts use when performing Boot Manager tasks.

Expand table

Value	Description
Input	**Environment variables**. Contains the property values, custom property values, database connections, deployment rules, and other information that the scripts require to complete the deployment process
Output	None
References	**BCDEdit.exe**. A tool for editing the Windows boot configuration
Location	- *distribution*\Scripts - *program_files*\Microsoft Deployment Toolkit\Scripts
Use	`<script language="VBScript" src="ZTIBCDUtility.vbs"/>`

Arguments

Expand table

Value	Description
None	None

Properties

Expand table

Name	Read	Write
None		

ZTIBde.wsf

This script installs and configures BitLocker on the target computer. BitLocker configuration is limited to New Computer scenarios that have hard disks configured with a single partition.

> **Note**
>
> For ZTI and UDI deployments, the **UILanguage** property must be set in CustomSettings.ini or in the MDT DB, because ZTIBde.wsf tries to read the locale from the **UILanguage** property.

Expand table

Value	Description
Input	**Environment variables**. Contains the property values, custom property values, database connections, deployment rules, and other information that the scripts require to complete the deployment process
Output	- **ZTIBde.log**. Log file that contains events that this script generates - **ZTIBdeFix_diskpart.log**. Log file that contains events that the Diskpart tool generates - **BDD.log**. Log file that contains events that all MDT scripts generate
References	- **CMD.exe**. Allows running of command-line tools - **Defrag.exe**. Defragments the hard disk - **Diskpart.exe**. Utility that allows for the automated management of disks, partitions, and volumes - **ServerManagerCmd.exe** - **ZTIDiskUtility.vbs**. Includes support functions and subroutines that the script uses - **ZTIOSRole.wsf**. Installs server roles - **ZTIUtility.vbs**. Includes support functions and subroutines that the script uses
Location	*distribution*\Scripts
Use	`cscript ZTIBde.wsf </debug:value>`

Arguments

Expand table

Value	Description
/debug:*value*	Outputs the event messages to the console and to the .log files. If the value specified in value is: - **TRUE**, event messages are sent to the console and the .log files - **FALSE**, event messages are sent only to the .log files (this is the behavior when the argument is not provided)

Properties

Expand table

Name	Read	Write
AdminPassword	-	
BDEDriveLetter	-	-
BDEDriveSize	-	
BDEInstall	-	
BDEInstallSuppress	-	
BDEKeyLocation	-	
BDEPin	-	
BDERecoveryKey	-	
BDESecondPass	-	-
BdeWaitForEncryption	-	
BitlockerInstalled	-	-
DeploymentMethod	-	
ISBDE	-	
OSDBitLockerCreateRecoveryPassword	-	
OSDBitLockerMode	-	
OSDBitLockerStartupKey	-	
OSDBitLockerStartupKeyDrive	-	
OSDBitLockerTargetDrive	-	
OSDBitLockerWaitForEncryption	-	
OSCurrentBuild	-	
OSCurrentVersion	-	
OSFeatures	-	-
OSRoles	-	-
OSRoleServices	-	-

Name	Read	Write
OSVersion	-	
SMSTSRebootRequested	-	-
SMSTSRetryRequested		-
TPMOwnerPassword	-	

ZTIBIOSCheck.wsf

This script checks the BIOS on the target computer, and then looks at a list of BIOSes that are incompatible with Windows. The list of incompatible BIOSes is stored in the ZTIBIOSCheck.xml file.

If the BIOS on the target computer is listed in the ZTIBIOSCheck.xml file, then the script returns a status that indicates the BIOS is incompatible with Windows and the deployment process should be terminated. For information on populating the list of incompatible BIOSes, see ZTIBIOSCheck.xml.

Expand table

Value	Description
Input	- **ZTIBIOSCheck.xml**. Contains a list of BIOSes that are known to be incompatible with Windows - **Environment variables**. Contains the property values, custom property values, database connections, deployment rules, and other information that the scripts require to complete the deployment process
Output	- **ZTIBIOSCheck.log**. Log file that contains events that this script generates - **BDD.log**. Log file that contains events that all MDT scripts generate
References	**ZTIUtility.vbs**. Includes support functions and subroutines that the script uses
Location	*distribution*\Scripts
Use	`cscript ZTIBIOSCheck.wsf </debug:value>`

Arguments

Expand table

Value	Description
/debug:*value*	Outputs the event messages to the console and to the .log files. If the value specified in value is: - **TRUE**, event messages are sent to the console and the .log files - **FALSE**, event messages are sent only to the .log files (this is the behavior when the argument is not provided)

Properties

Expand table

Name	Read	Write
None		

ZTICoalesce.wsf

Configuration Manager requires packages to be numbered sequentially starting with **PACKAGES001**, with no gaps in the number sequence. Otherwise, installation will fail.

This script allows you to define and name variables using identifying information about the program to run—for example, **ComputerPackages100**, **ComputerPackages110**, or **CollectionPackages150**. Then, when this script is run, Configuration Manager finds all variables that match a pattern (for example, all variable names that contain the string **Packages**) and builds a sequential list, without gaps, using the base name **PACKAGES**.

For example, if the following variables were defined (using computer variables, collection variables, or in CustomSettings.ini or the MDT DB, for example):

- **ComputerPackages100=XXX00001:Program**
- **ComputerPackages110=XXX00002:Program**
- **CollectionPackages150=XXX00003:Program**
- **Packages001=XXX00004:Program**

  After the script runs, the list would be:

- **PACKAGES001=XXX00004:Program**
- **PACKAGES002=XXX00001:Program**

- **PACKAGES003=XXX00002:Program**
- **PACKAGES004=XXX00003:Program**

  Configuration Manager would then be able to run all four programs.

Expand table

Value	Description
Input	**Environment variables**. Contains the property values, custom property values, database connections, deployment rules, and other information that the scripts require to complete the deployment process
Output	- **ZTICoalesce.log**. Log file that contains events that this script generates - **BDD.log**. Log file that contains events that all MDT scripts generate
References	**ZTIUtility.vbs**. Includes support functions and subroutines that the script uses
Location	*distribution*\Scripts
Use	`cscript ZTICoalesce.wsf </debug:value>`

Arguments

Expand table

Value	Description
/debug:*value*	Outputs the event messages to the console and to the .log files. If the value specified in value is: - **TRUE**, event messages are sent to the console and the .log files - **FALSE**, event messages are sent only to the .log files (This is the behavior when the argument is not provided.)
/CoalesceDigits:*value*	Specifies the number of digits that need to be provided when creating the numbering sequence. For example, a value of: - **2** would create PACKAGE03 - **3** would create PACKAGE003 The default value if this argument is not provided is **3**.

Properties

Expand table

Name	Read	Write
CoalescePattern	-	
CoalesceTarget	-	

ZTIConfigFile.vbs

This script contains common routines for processing MDT XML files.

Expand table

Value	Description
Input	**Environment variables**. Contains the property values, custom property values, database connections, deployment rules, and other information that the scripts require to complete the deployment process
Output	- **ZTIConfigFile.log**. Log file that contains events that this script generates - **BDD.log**. Log file that contains events that all MDT scripts generate
References	Net.exe
Location	*distribution*\Scripts
Use	`<script language="VBScript" src="ZTIConfigFile.vbs"/>`

Arguments

Expand table

Value	Description
None	None

Properties

Expand table

Name	Read	Write
IsSafeForWizardHTML	-	
MandatoryApplications	-	
SkipGroupSubFolders	-	

ZTIConfigure.wsf

This script configures the Unattend.xml file with the property values specified earlier in the MDT deployment process. The script configures the appropriate file based on the operating system being deployed.

This script reads the ZTIConfigure.xml file to determine how to update the Unattend.xml file with the appropriate values specified in the deployment properties. The ZTIConfigure.xml file contains the information to translate properties to settings in the Unattend.xml file.

Expand table

Value	Description
Input	- **ZTIConfigure.xml**. Contains a list of property values (specified earlier in the deployment process) and their corresponding configuration settings - **Environment variables**. Contains the property values, custom property values, database connections, deployment rules, and other information that the scripts require to complete the deployment process
Output	- **ZTIConfigure.log**. Log file that contains events that this script generates - **BDD.log**. Log file that contains events that all MDT scripts generate
References	**ZTIUtility.vbs**. Includes support functions and subroutines that the script uses
Location	*distribution*\Scripts
Use	`cscript ZTIConfigure.wsf </debug:value>`

Arguments

Expand table

Value	Description
/debug:*value*	Outputs the event messages to the console and to the .log files. If the value specified in value is: - **TRUE**, event messages are sent to the console and the .log files - **FALSE**, event messages are sent only to the .log files (This is the behavior when the argument is not provided.)

Properties

Expand table

Name	Read	Write
ComputerName	-	-
DeploymentType	-	
DeploymentMethod	-	
DeployRoot	-	
DestinationLogicalDrive	-	
DomainAdminDomain	-	
ImageBuild	-	
OSDAnswerFilePath	-	
OSDAnswerFilePathSysprep	-	
OSDComputerName	-	
Phase	-	
TaskSequenceID	-	

ZTIConfigureADDS.wsf

Expand table

Value	Description
Input	**Environment variables**. Contains the property values, custom property values, database connections, deployment rules, and other information that the scripts

Value	Description
	require to complete the deployment process
Output	- **ZTIConfigureADDS.log**. Log file that contains events that this script generates - **BDD.log**. Log file that contains events that all MDT scripts generate
References	- **Dcpromo.exe**. Installs and removes AD DS - **Net.exe**. Performs network management tasks - **ZTIUtility.vbs**. Includes support functions and subroutines that the script uses
Location	*distribution*\Scripts
Use	`cscript ZTIConfigureADDS.wsf </debug:value>`

Arguments

Expand table

Value	Description
/debug:*value*	Outputs the event messages to the console and to the .log files. If the value specified in value is: - **TRUE**, event messages are sent to the console and the .log files - **FALSE**, event messages are sent only to the .log files (This is the behavior when the argument is not provided.)

Properties

Expand table

Name	Read	Write
ADDSLogPath	-	
ADDSPassword	-	
ADDSUserDomain	-	
ADDSUserName	-	
AutoConfigDNS	-	

Name	Read	Write
ChildName	-	
ConfirmGC	-	
DatabasePath	-	
DomainLevel	-	
DomainNetBiosName	-	
ForestLevel	-	
NewDomain	-	
NewDomainDNSName	-	
OSVersion	-	
ParentDomainDNSName	-	
ReplicaOrNewDomain	-	-
ReplicaDomainDNSName	-	
ReplicationSourceDC	-	
SafeModeAdminPassword	-	
SiteName	-	
SysVolPath	-	

ZTIConfigureDHCP.wsf

This script configures DHCP on the target computer.

> ⓘ **Note**
>
> DHCP should already be installed on the target computer before running this script.

Expand table

Value	Description
Input	**Environment variables**. Contains the property values, custom property values, database connections, deployment rules, and other information that the scripts

Value	Description
	require to complete the deployment process
Output	- **ZTIConfigureDHCP.log**. Log file that contains events that this script generates - **BDD.log**. Log file that contains events that all MDT scripts generate
References	- **Netsh.exe**. A utility that permits automating the configuration of networking components - **ZTIUtility.vbs**. Includes support functions and subroutines that the script uses
Location	*distribution*\Scripts
Use	`cscript ZTIConfigureDHCP.wsf </debug:value>`

Arguments

Expand table

Value	Description
/debug:*value*	Outputs the event messages to the console and to the .log files. If the value specified in value is: - **TRUE**, event messages are sent to the console and the .log files - **FALSE**, event messages are sent only to the .log files (This is the behavior when the argument is not provided.)

Properties

Expand table

Name	Read	Write
DHCPScopesxDescription	-	
DHCPScopesxEndIP	-	
DHCPScopesxExcludeStartIP	-	
DHCPScopesxExcludeEndIP	-	
DHCPScopesxIP	-	
DHCPScopesxName	-	

Name	Read	Write
DHCPScopesxOptionRouter	-	
DHCPScopesxOptionDNSDomainName	-	
DHCPScopesxOptionDNSServer	-	
DHCPScopesxOptionLease	-	
DHCPScopesxOptionNBTNodeType	-	
DHCPScopesxOptionPXEClient	-	
DHCPScopesxOptionWINSServer	-	
DHCPScopesxStartIP	-	
DHCPScopesxSubnetmask	-	
DHCPServerOptionDNSDomainName	-	
DHCPServerOptionDNSServer	-	
DHCPServerOptionNBTNodeType	-	
DHCPServerOptionPXEClient	-	
DHCPServerOptionRouter	-	
DHCPServerOptionWINSServer	-	

> **Note**
>
> The *x*in the properties listed here is a placeholder for a zero-based array that contains DHCP configuration information.

ZTIConfigureDNS.wsf

This script configures DNS on the target computer. To perform the actual configuration tasks, the script uses the Dnscmd utility.

For more information about Dnscmd.exe, see Dnscmd Overview.

> **Note**
>
> DNS should already be installed on the target computer before running this script.

Expand table

Value	Description
Input	**Environment variables**. Contains the property values, custom property values, database connections, deployment rules, and other information that the scripts require to complete the deployment process
Output	- **ZTIConfigureDNS.log**. Log file that contains events that this script generates - **BDD.log**. Log file that contains events that all MDT scripts generate
References	- **Dnscmd.exe**. Assists administrators with DNS management - **ZTIUtility.vbs**. Includes support functions and subroutines that the script uses
Location	*distribution*\Scripts
Use	`cscript ZTIConfigureDNS.wsf </debug:value>`

Arguments

Expand table

Value	Description
/debug:*value*	Outputs the event messages to the console and to the .log files. If the value specified in value is: - **TRUE**, event messages are sent to the console and the .log files - **FALSE**, event messages are sent only to the .log files (This is the behavior when the argument is not provided.)

Properties

Expand table

Name	Read	Write
DNSServerOptionDisableRecursion	-	
DNSServerOptionBINDSecondaries	-	
DNSServerOptionFailOnLoad	-	
DNSServerOptionEnableRoundRobin	-	

Name	Read	Write
DNSServerOptionEnableNetmaskOrdering	-	
DNSServerOptionEnableSecureCache	-	
DNSServerOptionNameCheckFlag	-	
DNSZonesxName	-	
DNSZonesxType	-	
DNSZonesxMasterIP	-	
DNSZonesxDirectoryPartition	-	
DNSZonesxFileName	-	
DNSZonesxScavenge	-	
DNSZonesxUpdate	-	

> **Note**
>
> The *x*in the properties listed here is a placeholder for a zero-based array that contains DNS configuration information.

ZTIConnect.wsf

The MDT deployment process uses this script to authenticate with a server computer (such as a computer running SQL Server or another server that has a shared network folder). When this script is run, it validates that a connection can be created to the network shared folder specified in the **/uncpath** argument.

Expand table

Value	Description
Input	**Environment variables**. Contains the property values, custom property values, database connections, deployment rules, and other information that the scripts require to complete the deployment process
Output	- **ZTIConnect.log**. Log file that contains events that this script generates - **BDD.log**. Log file that contains events that all MDT scripts generate
References	**ZTIUtility.vbs**. Includes support functions and subroutines that the script uses

Value	Description
Location	*distribution*\Scripts
Use	`cscript ZTIConnect.wsf /UNCPath:<uncpath> </debug:value>`

Arguments

Expand table

Value	Description
/UNCPath:uncpath	Specifies a fully qualified UNC path to a network shared folder
/debug:*value*	Outputs the event messages to the console and to the .log files; if the value specified in value is: - **TRUE**, event messages are sent to the console and the .log files - **FALSE**, event messages are sent only to the .log files (This is the behavior when the argument is not provided.)

Properties

Expand table

Name	Read	Write
None		

ZTICopyLogs.wsf

Copy the Smsts.log and BDD.log files to a subfolder beneath the share that the **SLShare** property specifies. The subfolder takes the name that **OSDComputerName**, **_SMSTSMachineName**, or **HostName** specifies.

Expand table

Value	Description
Input	**Environment variables**. Contains the property values, custom property values, database connections, deployment rules, and other information that the scripts require to complete the deployment process

Value	Description
Output	- **ZTICopyLogs.log**. Log file that contains events that this script generates - **BDD.log**. Log file that contains events that all MDT scripts generate
References	**ZTIUtility.vbs**. Includes support functions and subroutines that the script uses
Location	*distribution*\Scripts
Use	`cscript ZTICopyLogs.wsf </debug:value>`

Arguments

Expand table

Value	Description
/debug: ***value***	Outputs the event messages to the console and to the .log files. If the value specified in value is: - **TRUE**, event messages are sent to the console and the .log files - **FALSE**, event messages are sent only to the .log files (This is the behavior when the argument is not provided.)

Properties

Expand table

Name	Read	Write
None		

ZTIDataAccess.vbs

This script contains common routines for database access.

Expand table

Value	Description
Input	**Environment variables**. Contains the property values, custom property values, database connections, deployment rules, and other information that the scripts

Value	Description
	require to complete the deployment process
Output	- **ZTIDataAccess.log**. Log file that contains events that this script generates - **BDD.log**. Log file that contains events that all MDT scripts generate
References	None
Location	*distribution*\Scripts
Use	`<script language="VBScript" src="ZTIDataAccess.vbs"/>`

Arguments

Expand table

Value	Description
None	None

Properties

Expand table

Name	Read	Write
_SMSTSReserved1	-	
_SMSTSReserved2	-	
RulesFile	-	
UserDomain	-	-
UserID	-	-
UserPassword	-	-

ZTIDisableBDEProtectors.wsf

If BitLocker is enabled, this script suspends the BitLocker protectors configured on the system.

Expand table

Value	Description
Input	**Environment variables**. Contains the property values, custom property values, database connections, deployment rules, and other information that the scripts require to complete the deployment process
Output	- **ZTIDisableBDEProtectors.log**. Log file that contains events that this script generates - **BDD.log**. Log file that contains events that all MDT scripts generate
References	**ZTIUtility.vbs**. Includes support functions and subroutines that the script uses
Location	*distribution*\Scripts
Use	`cscript ZTIDisableBDEProtectors.wsf </debug:value>`

Arguments

Expand table

Value	Description
/debug:*value*	Outputs the event messages to the console and to the .log files. If the value specified in value is: - **TRUE**, event messages are sent to the console and the .log files - **FALSE**, event messages are sent only to the .log files (This is the behavior when the argument is not provided.)

Properties

Expand table

Name	Read	Write
ImageBuild	-	
ISBDE		-
OSCurrentBuild	-	
OSCurrentVersion	-	

Name	Read	Write
OSVersion	-	

ZTIDiskpart.wsf

This script creates the disk partitions on the target computer by calling the Diskpart utility. The parameters used to configure the disk are specified by the Task Sequencer or in CustomSettings.ini. ZTIDiskpart.wsf is primarily run in New Computer scenarios. The process works like this:

1. The MDT deployment process runs the ZTIDiskpart.wsf script based on the steps and sequence of steps in the Task Sequencer.
2. ZTIDiskpart.wsf starts the Diskpart utility and sends it the required configuration commands.
3. ZTIDiskpart.wsf runs Diskpart.exe and provides a .txt file as a command-line parameter.
4. The disk is initially cleaned by sending Diskpart the **CLEAN** command.
5. If this is the first disk and no disk configuration has been specified by the Task Sequencer or in CustomSettings.ini, a single partition is created to store the operating system. However, if a disk configuration has been specified, the disk will be configured according to the specified configuration.
6. If BitLocker is to be enabled, space is reserved at the end of the first disk.
7. All format commands are queued until after Diskpart has finished. If not explicitly specified by the Task Sequencer or in CustomSettings.ini, ZTIDiskpart.wsf performs a quick format of drive C using the following command: `FORMAT C: /FS:NTFS /V:OSDisk /Q /Y`.
8. ZTIDiskpart.wsf copies the ZTIDiskpart_diskpart.log and BDD.log files from the RAM disk back to the hard drive.

   Customize the disk configuration of the target computer by providing the required information in the Task Sequencer or in CustomSettings.ini.

   For more information about configuring disks, see the MDT document *Using the Microsoft Deployment Toolkit*.

Expand table

Value	Description
Input	**Environment variables**. Contains the property values, custom property values, database connections, deployment rules, and other information that the scripts require to complete the deployment process
Output	- **ZTIDiskpart.log**. Log file that contains events that this script generates - **BDD.log**. Log file that contains events that all MDT scripts generate
References	- **Diskpart.exe**. Utility that allows for the automated management of disks, partitions, and volumes - **Format.com**. Formats the hard disk - **ZTIDiskUtility.vbs**. Includes support functions and subroutines that the script uses - **ZTIUtility.vbs**. Includes support functions and subroutines that the script uses
Location	*distribution*\Scripts
Use	`cscript ZTIDiskpart.wsf </debug:value>`

Arguments

Expand table

Value	Description
/debug:*value*	Outputs the event messages to the console and to the .log files. If the value specified in value is: - **TRUE**, event messages are sent to the console and the .log files - **FALSE**, event messages are sent only to the .log files (This is the behavior when the argument is not provided.)

Properties

Expand table

Name	Read	Write
BDEDriveLetter	-	

Name	Read	Write
BDEDriveSize	-	
BDEInstall	-	
DeployDrive	-	
DeploymentType	-	
DestinationDisk	-	
DestinationLogicalDrive		-
DoNotCreateExtraPartition	-	
ImageBuild	-	
OSDDiskIndex	-	
OSDDiskpartBiosCompatibilityMode	-	-
OSDDiskType	-	
OSDPartitions	-	
OSDPartitionStyle	-	
SMSTSLocalDataDrive		-
VolumeLetterVariable	-	

ZTIDiskUtility.vbs

This script contains disk-related functions and subroutines that the various scripts in the MDT deployment process call.

Expand table

Value	Description
Input	None
Output	- **ZTIDiskUtility.log**. Log file that contains events that this script generates - **BDD.log**. Log file that contains events that all MDT scripts generate
References	- **BcdBoot.exe**. Configures the system partition

Value	Description
	- **DiskPart.exe**. Utility that allows for the automated management of disks, partitions, and volumes
Location	*distribution*\Scripts
Use	`<script language="VBScript" src="ZTIDiskUtility.vbs"/>`

Arguments

Expand table

Value	Description
None	None

Properties

Expand table

Name	Read	Write
DestinationLogicalDrive	-	
UILanguage	-	-

ZTIDomainJoin.wsf

During the State Restore deployment phase, this script verifies that the computer is joined to a domain and recovers from failed attempts to join a domain.

Expand table

Value	Description
Input	**Environment variables**. Contains the property values, custom property values, database connections, deployment rules, and other information that the scripts require to complete the deployment process
Output	- **ZTIDomainJoin.log**. Log file that contains events that this script generates - **BDD.log**. Log file that contains events that all MDT scripts generate

Value	Description
References	- **LTISuspend.wsf** - **ZTIUtility.vbs**. Includes support functions and subroutines that the script uses
Location	*distribution*\Scripts
Use	`cscript ZTIDomainJoin.wsf </debug:value>`

Arguments

Expand table

Value	Description
/debug: ***value***	Outputs the event messages to the console and to the .log files. If the value specified in value is: - **TRUE**, event messages are sent to the console and the .log files - **FALSE**, event messages are sent only to the .log files (This is the behavior when the argument is not provided.)
/DomainErrorRecovery: ***value***	Attempts to join the computer to the domain. If the value specified in value is: - **AUTO**. Retry the domain join process. Restart and retry. This is the default script behavior. - **FAIL**. Stops all processing. All task sequence processing stops. - **MANUAL**. Stop processing; allows the user to manually join the computer to the domain.

Properties

Expand table

Name	Read	Write
DomainAdmin	-	
DomainAdminDomain	-	
DomainAdminPassword	-	

Name	Read	Write
DomainErrorRecovery	-	
DomainJoinAttempts	-	-
JoinDomain	-	
JoinWorkgroup	-	
LTISuspend		-
MachineObjectOU	-	
SMSTSRebootRequested		-
SMSTSRetryRequested		-

ZTIDrivers.wsf

This script installs additional device drivers onto the target computer before initiating the configuration of the operating system. This script reads the Drivers.xml file and copies the list of device driver files in the Drivers.xml file (created by and managed in the Drivers node in the Deployment Workbench) to the target computer.

Expand table

Value	Description
Input	**Environment variables**. Contains the property values, custom property values, database connections, deployment rules, and other information that the scripts require to complete the deployment process
Output	- **PnpEnum.xml**. Contains a list of all devices installed on the target computer - **ZTIDrivers.log**. Log file that contains events that this script generates - **BDD.log**. Log file that contains events that all MDT scripts generate
References	- **Attrib.exe**. Sets file and folder attributes - **CMD.exe. Allows running of command-line tools** - **Microsoft.BDD.PnpEnum.exe**. Utility that enumerates Plug and Play devices - **Reg.exe**. The console registry tool for reading and modifying registry data - **ZTIConfigFile.vbs**. Includes routines for processing XML files

Value	Description
	- **ZTIUtility.vbs**. Includes support functions and subroutines that the script uses
Location	*distribution*\Scripts
Use	`cscript ZTIDrivers.wsf </debug:value>`

Arguments

Expand table

Value	Description
/debug:*value*	Outputs the event messages to the console and to the .log files. If the value specified in value is: - **TRUE**, event messages are sent to the console and the .log files - **FALSE**, event messages are sent only to the .log files (This is the behavior when the argument is not provided.)

Properties

Expand table

Name	Read	Write
Architecture	-	
CustomDriverSelectionProfile	-	
DeploymentMethod	-	
DeploymentType	-	
DestinationLogicalDrive	-	-
DoCapture	-	
DriverPaths	-	
DriverSelectionProfile	-	
ImageBuild	-	
InstallFromPath	-	

Name	Read	Write
OSDAnswerFilePath	-	
OSDAnswerFilePathSysPrep	-	
OSDPlatformArch	-	
Phase	-	
ResourceRoot	-	

ZTIExecuteRunbook.wsf

This script runs Orchestrator runbooks on the target computer. An Orchestrator *runbook* is the sequence of activities that orchestrate actions on computers and networks. You can initiate Orchestrator runbooks in MDT using the Execute Runbook task sequence step type, which in turn runs this script.

Expand table

Value	Description
Input	Environment variables contain the property values, custom property values, database connections, deployment rules, and other information that the scripts require to complete the deployment process.
Output	- BDD.log contains events that all MDT scripts generate. - Return status of the runbook completion. - Return parameters from the runbook output.
References	- ZTIUtility.vbs includes support functions and subroutines that the script uses.
Location	*distribution*\Scripts
Use	`cscript ZTIExecuteRunbook.wsf </debug:value>`

Arguments

Expand table

Value	Description
/debug:*value*	Outputs the event messages to the console and to the .log files. If the value specified in value is:

Value	Description
	- **TRUE**, event messages are sent to the console and the .log files - **FALSE**, event messages are sent only to the .log files (This is the behavior when the argument is not provided.)

Properties

Expand table

Name	Read	Write
OrchestratorServer	-	
RunbookName	-	
RunbookID	-	
RunbookParameterMode	-	
RunbookParametersxParameterID	-	
RunbookParametersxParameterValue	-	
RunbookOutputParameters Note: If a runbook returns output parameters, a task sequence variable is created for each parameter and the return value of the parameter is assigned to the task sequence variable.		-

This script creates the task sequence variables listed in the following table for internal script use. Do not set these task sequence variables in CustomSettings.ini or in the MDT DB.

Expand table

Name	Description
OrchestratorServer	Name of the server running Orchestrator specified in **Orchestrator Server** in the Execute Runbook task sequence step
RunbookName	Name of the runbook specified in **Runbook** in the Execute Runbook task sequence step

Name	Description
RunbookID	Identifier assigned to the runbook on the Orchestrator server
RunbookParametersxParameterID	Identifier assigned to a specific runbook parameter on the Orchestrator server
RunbookParametersxParameterName	Name assigned to a specific runbook parameter on the Orchestrator server
RunbookParametersxParameterValue	Value assigned to a specific runbook parameter on the Orchestrator server

ZTIGather.wsf

This script gathers the properties and processing rules that control the deployment process. The properties and rules (also known as *local properties*) are explicitly defined in this script and contained in the ZTIGather.xml file, in the CustomSettings.ini file, and in the MDT DB (created in the Database node in the Deployment Workbench).

Expand table

Value	Description
Input	**Environment variables**. Contains the property values, custom property values, database connections, deployment rules, and other information that the scripts require to complete the deployment process
Output	- **ZTIGather.log**. Log file that contains events that this script generates - **BDD.log**. Log file that contains events that all MDT scripts generate
References	- **Wpeutil.exe**. Initializes Windows PE and network connections; initiates LTI - **ZTIDataAccess.vbs**. Contains routines for database access - **ZTIUtility.vbs**. Includes support functions and subroutines that the script uses
Location	*distribution*\Scripts
Use	`cscript ZTIGather.wsf </debug:value> </localonly> </inifile:ini_file_name>`

Arguments

Expand table

Value	Description
/debug:*value*	Outputs the event messages to the console and to the .log files. If the value specified in value is: - **TRUE**, event messages are sent to the console and the .log files - **FALSE**, event messages are sent only to the .log files (This is the behavior when the argument is not provided.)
/localonly	Returns only information about the target computer and the current operating system installed on the target computer; does not parse the input .ini file (specified in the **/inifile** argument); returns properties and rules specified in the .ini file If not specified, the script returns information about the target computer and the currently installed operating system; parses the .ini file
/inifile:ini_file_name	Name and path of the input .ini file that contains the properties and rules used in the deployment processIf not specified, the script uses the default value in CustomSettings.ini

Properties

Expand table

Name	Read	Write
All	-	-

ZTIGroups.wsf

This script captures and restores the local group membership on the target computer. This script is called with the**/capture** argument to back up the group membership from the target computer before deploying the operating system. The **CaptureGroups** property contains the list of groups that script backs up. The script is called with the**/restore** argument to restore the group membership after the operating system is deployed. When performing a restore operation, it restores the membership of all groups that were backed up when the script was run using the **/capture** argument.

> **Note**
>
> When restoring group membership, the script does not create any destination groups that do not already exist on the target computer. Therefore, be sure to

include all required groups in the reference computer when building the image file.

Expand table

Value	Description
Input	**Environment variables**. Contains the property values, custom property values, database connections, deployment rules, and other information that the scripts require to complete the deployment process
Output	- **ZTIGroups.log**. Log file that contains events that this script generates - **BDD.log**. Log file that contains events that all MDT scripts generate
References	**ZTIUtility.vbs**. Includes support functions and subroutines that the script uses
Location	*distribution*\Scripts
Use	`cscript ZTIGroups.wsf </debug:value> </backup> </restore>`

Arguments

Expand table

Value	Description
/debug:*value*	Outputs the event messages to the console and to the .log files. If the value specified in value is: - **TRUE**, event messages are sent to the console and the .log files - **FALSE**, event messages are sent only to the .log files (This is the behavior when the argument is not provided.)
/capture	Backs up the group membership of the local groups on the target computer as specified in the **CaptureGroups** property
/restore	Restores the group membership to the local groups backed up earlier in the deployment process

Properties

Expand table

Name	Read	Write
CaptureGroups	-	
Groups	-	-
HostName	-	

ZTILangPacksOnline.wsf

This script installs language packs for Windows operating systems. The script is expecting the language pack CAB files in a folder structure containing at least one folder.

Expand table

Value	Description
Input	**Environment variables**. Contains the property values, custom property values, database connections, deployment rules, and other information that the scripts require to complete the deployment process
Output	- **ZTILangPacksOnline.log**. Log file that contains events that this script generates - **BDD.log**. Log file that contains events that all MDT scripts generate
References	- **CMD.exe**. Allows running of command-line tools - **Lpksetup.exe**. The Language Pack Setup tool used to add or remove language packs - **ZTIUtility.vbs**. Includes support functions and subroutines that the script uses
Location	*distribution*\Scripts
Use	`cscript ZTILangPacksOnline.wsf </debug:value>`

Arguments

Expand table

Value	Description
/debug:*value*	Outputs the event messages to the console and to the .log files. If the value specified in value is:

Value	Description
	- **TRUE**, event messages are sent to the console and the .log files - **FALSE**, event messages are sent only to the .log files (This is the behavior when the argument is not provided.)

Properties

Expand table

Name	Read	Write
Architecture	-	
OSVersion	-	

ZTIModifyVol.wsf

This script modifies a volume to set the GPT ID and attributes for utility volumes, which is necessary for creating Windows RE partitions on computers with UEFI. This script needs to be called when deploying to computers with UEFI for these situations:

- LTI deployments where custom partition (volume) structures are being created, such as creating five partitions instead of the standard four partitions that are typically created for use with UEFI
- All ZTI and UDI deployments

> **Note**
>
> This script is intended to be called only when creating partitions structures for use with UEFI. This script should not be called when creating partition structures to be used in deployments without UEFI.

Expand table

Value	Description
Input	**Environment variables**. Contains the property values, custom property values, database connections, deployment rules, and other information that the scripts require to complete the deployment process

Value	Description
Output	BDD.log contains events that all MDT scripts generate.
References	ZTIUtility.vbs includes support functions and subroutines that the script uses.
Location	*distribution*\Scripts
Use	`cscript ZTIModifyVol.wsf /UtilityVol:value </debug:value>`

Arguments

Expand table

Value	Description
/UtilityVol:*value*	Provides the drive letter of the volume that needs to be configured for a Windows RE Tools partition for use with computers with UEFI (for example, "E:")
/debug:*value*	Outputs the event messages to the console and to the .log files. If the value specified in value is: - **TRUE**, event messages are sent to the console and the .log files - **FALSE**, event messages are sent only to the .log files (This is the behavior when the argument is not provided.)

Properties

Expand table

Name	Read	Write
UtilityVol	-	

ZTIMoveStateStore.wsf

This script moves the captured user state and backup files to C:\Windows\Temp\StateStore.

> **Note**
>
> This script is run only when deploying images using Configuration Manager.

Expand table

Value	Description
Input	**Environment variables**. Contains the property values, custom property values, database connections, deployment rules, and other information that the scripts require to complete the deployment process
Output	- **ZTIMoveStateStore.log**. Log file that contains events that this script generates - **BDD.log**. Log file that contains events that all MDT scripts generate
References	**ZTIUtility.vbs**. Includes support functions and subroutines that the script uses
Location	*distribution*\Scripts
Use	`cscript ZTIMoveStateStore.wsf </debug:value>`

Arguments

Expand table

Value	Description
/debug:*value*	Outputs the event messages to the console and to the .log files. If the value specified in value is: - **TRUE**, event messages are sent to the console and the .log files - **FALSE**, event messages are sent only to the .log files (This is the behavior when the argument is not provided.)

Properties

Expand table

Name	Read	Write
None		

ZTINextPhase.wsf

This script updates the **Phase** property to the next phase in the deployment process. The Task Sequencer uses these phases to determine the sequence in which each task must be completed. The **Phase** property includes the following values:

- **VALIDATION**. Identify that the target computer is capable of running the scripts necessary to complete the deployment process.
- **STATECAPTURE**. Save any user state migration data before deploying the new target operating system.
- **PREINSTALL**. Complete any tasks that need to be done (such as creating new partitions) before the target operating system is deployed.
- **INSTALL**. Install the target operating system on the target computer.
- **POSTINSTALL**. Complete any tasks that need to be done before restoring the user state migration data. These tasks customize the target operating system before starting the target computer the first time after deployment (such as installing updates or adding drivers).
- **STATERESTORE**. Restore the user state migration data saved during the State Capture Phase.

  For more information about the **Phase** property, see Phase.

Expand table

Value	Description
Input	**Environment variables**. Contains the property values, custom property values, database connections, deployment rules, and other information that the scripts require to complete the deployment process
Output	- **ZTINextPhase.log**. Log file that contains events that this script generates - **BDD.log**. Log file that contains events that all MDT scripts generate
References	**ZTIUtility.vbs**. Includes support functions and subroutines that the script uses
Location	*distribution*\Scripts
Use	`cscript ZTINextPhase.wsf </debug:value>`

Arguments

Expand table

Value	Description
/debug:*value*	Outputs the event messages to the console and to the .log files. If the value specified in value is:

Value	Description
	- **TRUE**, event messages are sent to the console and the .log files - **FALSE**, event messages are sent only to the .log files (This is the behavior when the argument is not provided.)

Properties

Expand table

Name	Read	Write
DeploymentMethod	-	
Phase	-	-

ZTINICConfig.wsf

This script configures activated network adapters with values that ZTIGather.wsf captured based on the properties listed in the CustomSettings.ini file or the MDT DB (created in the Database node in the Deployment Workbench).

Expand table

Value	Description
Input	**Environment variables**. Contains the property values, custom property values, database connections, deployment rules, and other information that the scripts require to complete the deployment process
Output	- **ZTINICConfig.log**. Log file that contains events that this script generates - **BDD.log**. Log file that contains events that all MDT scripts generate
References	- **ZTIUtility.vbs**. Includes support functions and subroutines that the script uses - **ZTINicUtility.vbs**. Includes support functions and subroutines that the script uses
Location	*distribution*\Scripts
Use	`cscript ZTINicConfig.wsf </debug:value> </ForceCapture> </RestoreWithinWinPE>`

Arguments

Expand table

Value	Description
/debug:*value*	Outputs the event messages to the console and to the .log files. If the value specified in value is: - **TRUE**, event messages are sent to the console and the .log files - **FALSE**, event messages are sent only to the .log files (This is the behavior when the argument is not provided.)
/ForceCapture	If there are any local networking adapters with static IP addresses saved, this script captures those settings and saves them to the local environment—for example, C:\MININT\SMSOSD\OSDLogs\Variables.dat. This script can be useful in capturing static IP settings for a large number of computers for automation.
/RestoreWithinWinPE	When specified, applies any saved static IP network settings to the local computer, when appropriate; used for internal processing only.

Properties

Expand table

Name	Read	Write
DeployDrive	-	-
DeploymentMethod	-	
DeploymentType	-	
DeployRoot	-	
OSDAdapterCount	-	-
OSGuid	-	
OSDMigrateAdapterSettings	-	
Phase	-	

ZTINICUtility.vbs

This script contains network adapter-related functions and subroutines that the various scripts in the MDT deployment process call.

Expand table

Value	Description
Input	None
Output	None
References	- **CMD.exe**. Allows running of command-line tools - **Netsh.exe**. A utility used to automate the configuration of networking components
Location	*distribution*\Scripts
Use	`<script language="VBScript" src="ZTINicUtility.vbs"/>`

Arguments

Expand table

Value	Description
None	None

Properties

Expand table

Name	Read	Write
OSDAdapter*AdapterIndexAdapterName*	-	-

> **Note**
>
> *AdapterIndex*in this property is a placeholder for a zero-based array that contains network adapter information.

ZTIOSRole.wsf

This script installs server roles for target computers that are running Windows operating systems. The script reads the **OSRoles**, **OSRoleServices**, and **OSFeatures** properties to determine what should be installed.

> **Note**
>
> This script is intended to be called only by the **Install Roles and Features** and**Uninstall Roles and Features** task sequence steps. Calling this script directly is not supported.

Expand table

Value	Description
Input	**Environment variables**. Contains the property values, custom property values, database connections, deployment rules, and other information that the scripts require to complete the deployment process
Output	- **ZTIOSRole.log**. Log file that contains events that this script generates - **BDD.log**. Log file that contains events that all MDT scripts generate
References	- **CMD.exe**. Allows running of command-line tools - **OCSetup.exe**. Adds to or removes Windows optional components - **ServerManagerCmd.exe**. Installs, configures, and manages Windows Server roles and features - **Sysocmgr.exe**. Adds or removes Windows components - **ZTIUtility.vbs**. Includes support functions and subroutines that the script uses
Location	*distribution*\Scripts
Use	`cscript ZTIOSRole.wsf </debug:value>`

Arguments

Expand table

Value	Description
/debug:*value*	Outputs the event messages to the console and to the .log files. If the value specified in value is: - **TRUE**, event messages are sent to the console and the .log files - **FALSE**, event messages are sent only to the .log files (This is the behavior when the argument is not provided.)

Value	Description
/Uninstall	If provided, this argument indicates that the roles and features will be uninstalled. If not provided, the script assumes the roles and features will be installed.

Properties

Expand table

Name	Read	Write
IsServerCoreOS	-	
OSFeatures	-	
OSRoles	-	
OSRoleServices	-	
OSVersion	-	
SMSTSRebootRequested		-

ZTIPatches.wsf

This script installs updates (language packs, security updates, and so on) that are listed in the Packages.xml file. The script self-terminates if the deployment is not in one of the following states:

- **Phase** equals **PREINSTALL**
- **DeploymentMethod** equals **SCCM**

  The script starts Pkgmgr if **DeploymentMethod** equals **SCCM**.

Expand table

Value	Description
Input	**Environment variables**. Contains the property values, custom property values, database connections, deployment rules, and other information that the scripts require to complete the deployment process
Output	- **ZTIPatches.log**. Log file that contains events that this script generates - **BDD.log**. Log file that contains events that all MDT scripts generate

Value	Description
References	- **Expand.exe**. Expands compressed files - **Pkgmgr.exe**. Installs or updates Windows Vista offline - **ZTIConfigFile.vbs**. Includes routines for processing XML files - **ZTIUtility.vbs**. Includes support functions and subroutines that the script uses
Location	*distribution*\Scripts
Use	`cscript ZTIPatches.wsf </debug:value>`

Arguments

Expand table

Value	Description
/debug:*value*	Outputs the event messages to the console and to the .log files. If the value specified in value is: - **TRUE**, event messages are sent to the console and the .log files - **FALSE**, event messages are sent only to the .log files (This is the behavior when the argument is not provided.)

Properties

Expand table

Name	Read	Write
Architecture	-	
CustomPackageSelectionProfile	-	
DeployRoot	-	
DeploymentMethod	-	
DeploymentType	-	
DestinationLogicalDrive	-	
LanguagePacks	-	

Name	Read	Write
OSDAnswerFilePath	-	
OSDPlatformArch	-	
PackageSelectionProfile	-	
Phase	-	
ResourceRoot	-	

ZTIPowerShell.wsf

This script runs a Windows PowerShell script using a custom Windows PowerShell host.

Expand table

Value	Description
Input	**Environment variables**. Contains the property values, custom property values, database connections, deployment rules, and other information that the scripts require to complete the deployment process
Output	- **ZTIPowerShell.log**. Log file that contains events that this script generates - **BDD.log**. Log file that contains events that all MDT scripts generate - **Return code**. The numeric value returned by the Windows PowerShell script after completion, which indicates the completion status of the script.
References	- **Microsoft.BDD.TaskSequencePSHost.exe**. Custom Windows PowerShell host used to run the Windows PowerShell script.
Location	*distribution*\Scripts
Use	`cscript ZTIPowerShell.wsf`

Arguments

Expand table

Value	Description
None	

Properties

Expand table

Name	Read	Write
None		

ZTIPrereq.vbs

This script verifies that the target computer has the prerequisite software installed and that it is functional. The checks the script performs are:

- Determine whether the Windows Script version is equal to or greater than version 5.6.
- Verify that errors do not occur when object references are instantiated to Wscript.Shell, Wscript.Network, Scripting.FileSystemObject MSXML2.DOMDocument, and the Process environment.

  If any one of the checks fails, an error is raised and the script exits the **ValidatePrereq** procedure.

Expand table

Value	Description
Input	None
Output	None
References	None
Location	*distribution*\Scripts
Use	`None`

Arguments

Expand table

Value	Description
None	None

Properties

Expand table

Name	Read	Write
None		

ZTISCCM.wsf

This script initializes ZTI when deploying using Configuration Manager. The script performs the following procedure:

1. If debugging is activated, the script creates the OSD.Debug file.
2. The script configures these properties:
    - **ScriptRoot**is set to the parent folder of the currently running script.
    - **DeployRoot** is set to the parent folder of **ScriptRoot**.
    - **ResourceRoot** is set to **DeployRoot**.
    - **DeploySystemDrive** is set to **C:**.
    - **DeploymentMethod** is set to **SCCM**.
3. When **DeployRoot**contains **:**:
    - The **DeployRoot** folder is copied to **_SMSTSMDataPath**\WDPackage
    - **ScriptRoot** is set to **_SMSTSMDataPath**\WDPackage\Scripts
    - **DeployRoot** is set to the parent folder of **ScriptRoot**
    - **ResourceRoot** is set to **DeployRoot**
4. When **Phase** is **NULL**:
    - If the %SystemDrive% environment variable is **X:**, then **DeploymentType**is set to **NEWCOMPUTER** and **Phase** is set to **PREINSTALL**.
      Otherwise,**DeploymentType** is set to **REPLACE** and **Phase** is set to **VALIDATION**.
    - If the **OldComputer**.tag file exists in the parent folder of the current running script, **DeploymentType** is set to **REPLACE** and **Phase** is set to **VALIDATION**.

Otherwise,**DeploymentType** is set to **REFRESH** and **Phase** is set to **VALIDATION**.

For more information about these properties, see the Properties article.

Expand table

Value	Description
Input	**Environment variables**. Contains the property values, custom property values, database connections, deployment rules, and other information that the scripts require to complete the deployment process
Output	- **ZTISCCM.log**. Log file that contains events that this script generates - **BDD.log**. Log file that contains events that all MDT scripts generate
References	**ZTIUtility.vbs**. Includes support functions and subroutines that the script uses
Location	*distribution*\Scripts
Use	`cscript ZTISCCM.wsf </debug:value>`

Arguments

Expand table

Value	Description
/debug:*value*	Outputs the event messages to the console and to the .log files. If the value specified in value is: - **TRUE**, event messages are sent to the console and the .log files - **FALSE**, event messages are sent only to the .log files (This is the behavior when the argument is not provided.)

Properties

Expand table

Name	Read	Write
_SMSTSMDataPath	-	
Architecture	-	

Name	Read	Write
BDDPackageID	-	-
DeploymentMethod	-	-
DeploymentType	-	-
DeployRoot	-	-
Phase	-	-
ResourceRoot	-	-
ScriptRoot	-	-
ToolRoot	-	-

ZTISetVariable.wsf

This script sets the specified global task sequence variable that corresponds to the name contained in **VariableName** to the value contained in **VariableValue**.

Expand table

Value	Description
Input	**Environment variables**. Contains the property values, custom property values, database connections, deployment rules, and other information that the scripts require to complete the deployment process
Output	- **ZTISetVariable.log**. Log file that contains events that this script generates - **BDD.log**. Log file that contains events that all MDT scripts generate
References	**ZTIUtility.vbs**. Includes support functions and subroutines that the script uses
Location	*distribution*\Scripts
Use	`cscript ZTISetVariable.wsf </debug:value>`

Arguments

Expand table

Value	Description
/debug:*value*	Outputs the event messages to the console and to the .log files. If the value specified in value is: - **TRUE**, event messages are sent to the console and the .log files - **FALSE**, event messages are sent only to the .log files (This is the behavior when the argument is not provided.)

Properties

Expand table

Name	Read	Write
VariableName	-	
VariableValue	-	

ZTITatoo.wsf

This script tattoos the target computer with identification and version information. The script performs the following procedure:

1. Locate and copy the ZTITatoo.mof file to the %SystemRoot%\System32\Wbem folder. Any preexisting ZTITatoo.mof that exists at the destination will be deleted before starting the copy operation.

2. Mofcomp.exe will be run using the following command:

   ```
   exe
   ```

   ```
   %SystemRoot%\System32\Wbem\Mofcomp.exe -autorecover
   %SystemRoot%\System32\Wbem\ZTITatoo.mof.
   ```

3. For all deployment methods (LTI, ZTI, and UDI), these deployment details are written for all deployment methods to the registry at **HKEY_LOCAL_MACHINE\Software\Microsoft\Deployment 4**:

   - **Deployment Method** is set to the deployment method being used and can be set to **LTI**, **ZTI**, or **UDI**, depending on the deployment method being performed.

- **Deployment Source** is set to the source for the deployment and can be set to **OEM**, **MEDIA**, or the value in the **DeploymentMethod** property.
- **Deployment Type** is set to the **DeploymentType** property.
- **Deployment Timestamp** is set to the current date in WMI date format.
- **Deployment Toolkit Version** is set to the **Version** property.

4. For LTI deployments, these deployment details are written to the registry at **HKEY_LOCAL_MACHINE\Software\Microsoft\Deployment 4**:
    - **Task Sequence ID** is set to the **TaskSequenceID**property.
    - **Task Sequence Name** is set to the **TaskSequenceName** property.
    - **Task Sequence Version** is set to the **TaskSequenceVersion** property.
5. For all Configuration Manager deployments (ZTI and UDI for Configuration Manager), these deployment details are written to the registry at **HKEY_LOCAL_MACHINE\Software\Microsoft\Deployment 4**:
    - OSD Package ID is set to the **_SMSTSPackageID** task sequence variable.
    - **OSD Program Name** is always set to "*****".
    - **OSD Advertisement ID** is set to the **_SMSTSAdvertID** task sequence variable.
6. For LTI deployments where an image is being captured, these deployment details are written to the registry at **HKEY_LOCAL_MACHINE\Software\Microsoft\Deployment 4**:
    - **Capture Method** is set to the deployment method being used and can be set to **LTI**, **ZTI**, or **UDI**, depending on the deployment method being performed.
    - **Capture Timestamp** is set to the current date in WMI date format.
    - **Capture Toolkit Version** is set to the **Version** property.
    - **Capture Task Sequence ID** is set to the **TaskSequenceID**property.
    - **Capture Task Sequence Name** is set to the **TaskSequenceName** property.
    - **Capture Task Sequence Version** is set to the **TaskSequenceVersion** property.
7. For all Configuration Manager deployments (ZTI and UDI for Configuration Manager) in which an image is being captured, these deployment details are

written to the registry at **HKEY_LOCAL_MACHINE\Software\Microsoft\Deployment 4**:

- **Capture OSD Package ID** is set to the **_SMSTSPackageID** task sequence variable.
- **Capture OSD Program Name** is always set to "*****".
- **Capture OSD Advertisement ID** is set to the **_SMSTSAdvertID**task sequence variable.

> **Note**
>
> This script is not designed to run on Windows PE.

Expand table

Value	Description
Input	**Environment variables**. Contains the property values, custom property values, database connections, deployment rules, and other information that the scripts require to complete the deployment process
Output	- **ZTITatoo.log**. Log file that contains events that this script generates - **BDD.log**. Log file that contains events that all MDT scripts generate
References	- **Mofcomp.exe**. Command-line .mof file compiler - **ZTIUtility.vbs**. Includes support functions and subroutines that the script uses
Location	*distribution*\Scripts
Use	`cscript ZTITatoo.wsf </debug:value>`

Arguments

Expand table

Value	Description
/debug:*value*	Outputs the event messages to the console and to the .log files. If the value specified in value is: - **TRUE**, event messages are sent to the console and the .log files

Value	Description
	- **FALSE**, event messages are sent only to the .log files (This is the behavior when the argument is not provided.)

Properties

Expand table

Name	Read	Write
_SMSTSAdvertID	-	
_SMSTSPackageID	-	
_SMSTSSiteCode	-	
DeploymentMethod	-	
DeploymentType	-	
Version	-	
TaskSequenceID	-	
TaskSequenceName	-	
TaskSequenceVersion	-	

ZTIUserState.wsf

This script initializes USMT to capture and restore user state on the target computer.

Expand table

Value	Description
Input	**Environment variables**. Contains the property values, custom property values, database connections, deployment rules, and other information that the scripts require to complete the deployment process
Output	- **ZTIUserState.log**. Log file that contains events that this script generates - **BDD.log**. Log file that contains events that all MDT scripts generate
References	- **CMD.exe**. Allows running of command-line tools - **Loadstate.exe**. Deposits user state data on a target computer

Value	Description
	- **Msiexec.exe**. Manages the installation of .msi-based applications - **Scanstate.exe**. Collects user data and settings - **USMT Application Files** - **ZTIUtility.vbs**. Includes support functions and subroutines that the script uses
Location	*distribution*\Scripts
Use	`cscript ZTIUserState.wsf </debug:value>`

Arguments

Expand table

Value	Description
/debug:value	Outputs the event messages to the console and to the .log files. If the value specified in value is: - **TRUE**, event messages are sent to the console and the .log files - **FALSE**, event messages are sent only to the .log files (This is the behavior when the argument is not provided.)
/Capture	-
/Estimate	-
/Restore	-

Properties

Expand table

Name	Read	Write
Architecture	-	
DeploymentMethod	-	
DeploymentType	-	
DestinationLogicalDrive	-	

Name	Read	Write
ImageBuild	-	
ImageSize	-	
ImageSizeMultiplier	-	
InstallFromPath	-	
IsServerOS	-	
LoadStateArgs	-	
OSCurrentVersion	-	
OSDMigrateAdditionalCaptureOptions	-	-
OSDMigrateAdditionalRestoreOptions	-	-
OSDPackagePath	-	
OSDStateStorePath		-
OSVersion	-	
ScanStateArgs	-	
StatePath	-	-
UDDir	-	
UDProfiles	-	
UDShare	-	
UserDataLocation	-	-
USMTConfigFile	-	
USMTEstimate	-	-
USMTLocal		-
USMTMigFiles	-	

ZTIUtility.vbs

This script contains utility functions that most of the MDT scripts use.

Expand table

Value	Description
Input	**Environment variables**. Contains the property values, custom property values, database connections, deployment rules, and other information that the scripts require to complete the deployment process
Output	None
References	- **Credentials_ENU.xml**. Prompts the user for credentials that will be used when connecting to network resources - **IPConfig.exe**. Displays all current TCP/IP network configuration values and refreshes DHCP and DNS settings - **MSHTA.exe**. HTML application host - **Regsvr32.exe**. Registers files (.dll, .exe, .ocx, and so on) with the operating system - **Xcopy.exe**. Copies files and directories, including subdirectories
Location	- *distribution*\Scripts - *program_files*\Microsoft Deployment Toolkit\Scripts
Use	`<script language="VBScript" src="ZTIUtility.vbs"/>`

Arguments

Expand table

Value	Description
None	None

Properties

Expand table

Name	Read	Write
_SMSTSAdvertID	-	
_SMSTSCurrentActionName	-	
_SMSTSCustomProgressDialogMessage	-	
_SMSTSInstructionTableSize	-	

Name	Read	Write
_SMSTSLogPath	-	
_SMSTSMachineName	-	
_SMSTSNextInstructionPointer	-	
_SMSTSOrgName	-	
_SMSTSPackageID	-	
_SMSTSPackageName	-	
_SMSTSPackagePath	-	
_SMSTSReserved1	-	
_SMSTSReserved2	-	
Architecture	-	
AssetTag	-	
ComputerName	-	
Debug	-	-
DeploymentMethod	-	
DeployRoot	-	
DestinationDisk	-	-
DestinationLogicalDrive	-	-
DestinationPartition	-	-
EventShare	-	
HostName	-	
ImageBuild	-	-
ImageFlags		-
ImageIndex		-
ImageLanguage		-
ImageProcessor		-
ImageSize		-

Name	Read	Write
InstallFromPath		-
JoinDomain	-	
LogPath	-	-
MacAddress	-	
OSCurrentVersion	-	
OSDAdvertID	-	
OSDAnswerFilePath	-	-
OSDAnswerFilePathSysprep	-	-
OSDComputerName	-	-
OSDPackageID	-	
OSDPackagePath	-	
OSDTargetSystemDrive	-	
OSGUID		-
OSSKU	-	
OSVersion	-	
Phase	-	
Processor_Architecture	-	
ResourceRoot	-	
SLShare	-	
SLShareDynamicLogging	-	
TaskSequenceID	-	
TaskSequenceName		-
TaskSequenceVersion		-
UDDir	-	
UDShare	-	
UserDomain	-	-

Name	Read	Write
UserID	-	-
UserPassword	-	-
UUID	-	
Version **Note:** This variable is an internal variable that represents the version of MDT.	-	-
WDSServer	-	

ZTIValidate.wsf

This script ensures that it is safe for the deployment to continue by validating the condition of the target computer. The script processes are:

- If **DeploymentType** equals REFRESH and the target computer is a server, the script exits.
- If **OSInstall** exists and is not equal to YES, the script exits.
- Verify that the minimum amount of RAM exists on the target computer; if not, the script exits.
- Verify that the processor meets the minimum required speed; if not, the script exits.
- Verify that the hard disk size meets the minimum size requirements; if not, the script exits.
- Verify that the target computer's operating system is installed on drive C; if not, the script exits.
- If **DeploymentType** = **REFRESH**, verify that drive C is not compressed by running `Compact /u C:\`.

Expand table

Value	Description
Input	**Environment variables**. Contains the property values, custom property values, database connections, deployment rules, and other information that the scripts require to complete the deployment process

Value	Description
Output	- **ZTIValidate.log**. Log file that contains events that this script generates - **BDD.log**. Log file that contains events that all MDT scripts generate
References	- **Compact.exe**. Displays or alters the compression of files on NTFS file system partitions - **ZTIUtility.vbs**. Includes support functions and subroutines that the script uses
Location	*distribution*\Scripts
Use	`cscript ZTIValidate.wsf </debug:value>`

Arguments

Expand table

Value	Description
/debug:*value*	Outputs the event messages to the console and to the .log files. If the value specified in value is: - **TRUE**, event messages are sent to the console and the .log files - **FALSE**, event messages are sent only to the .log files (This is the behavior when the argument is not provided.)

Properties

Expand table

Name	Read	Write
DeploymentType	-	
DestinationLogicalDrive	-	-
ImageBuild	-	
ImageMemory	-	
ImageProcessorSpeed	-	
ImageSize	-	

Name	Read	Write
ImageSizeMultiplier	-	
IsServerOS	-	
Memory	-	
OSDPackagePath	-	
OSInstall	-	
ProcessorSpeed	-	
SMSTSLocalDataDrive		-
VerifyOS	-	

ZTIVHDCreate.wsf

This script is used to create a virtual hard disk (.vhd or .avhd) file on the target computer and mount the .vhd file as a disk. Then, other portions of the LTI deployment process deploy the Windows operating system and applications to the newly created virtual hard disk. The script processes are as follows:

- The **Class_Initialize** method is used to initialize the **VHDInputVariable** variable.
- Validate that **VHDCreateSource** is defined and locates the source .vhd file (if specified).
- Generate a random .vhd file name if **VHDCreateFilename** equals RANDOM or "" (null).
- Verify that the folder exists where the .vhd file (specified in **VHDCreateFileName**) is to be created.
- Create the .vhd file using the values in **VHDCreateSizePercent**, **VHDCreateSizeMax**, and **VHDCreateType**.
- Create a differencing disk (if specified) using the value in **VHDCreateDiffVHD**.
- The newly created .vhd file and the optional differencing disk are mounted.
- The disk number of the mounted virtual hard disk is returned.

Expand table

Value	Description
Input	**Environment variables**. Contains the property values, custom property values, database connections, deployment rules, and other information that the scripts require to complete the deployment process
Output	- **ZTIVHDCreate.log**. Log file that contains events that this script generates - **BDD.log**. Log file that contains events that all MDT scripts generate
References	- **ZTIDiskUtility.vbs**. Includes support functions and subroutines the script uses - **ZTIUtility.vbs**. Includes support functions and subroutines that the script uses
Location	*distribution*\Scripts
Use	`cscript ZTIVHDCreate.wsf </debug:value>`

Arguments

Expand table

Value	Description
/debug:*value*	Outputs the event messages to the console and to the .log files. If the value specified in value is: - **TRUE**, event messages are sent to the console and the .log files - **FALSE**, event messages are sent only to the .log files (This is the behavior when the argument is not provided.)

Properties

Expand table

Name	Read	Write
VHDCreateDiffVHD	-	
VHDCreateFileName	-	
VHDCreateSizeMax	-	
VHDCreateSource	-	
VHDCreateType	-	

Name	Read	Write
VHDDisks		-
VHDInputVariable	-	
VHDOutputVariable	-	

ZTIWindowsUpdate.wsf

This script downloads and installs updates from computers on a corporate network that are running WSUS, Windows Update, or Microsoft Update using the Windows Update Agent (WUA) application programming interface (API). By default, this feature is disabled in each task sequence and must be manually activated to run.

Most enterprises will already have teams and infrastructures in place to update newly deployed computers over the corporate network. This process involves tracking the latest set of patches, drivers, and updates available for each desktop configuration and determining which updates should be downloaded and installed for each configuration. If the organization already has an established process, this script might not be necessary. This script was designed to fill a need for deployment teams that might not have established processes, yet want to ensure that target computers are updated when deployed.

This script automatically scans the target computer and downloads a wide range of updates that are found to be applicable. Among these are:

- Windows service packs
- Non-Microsoft drivers that were placed on Windows Update
- The latest hotfix updates
- Microsoft Office updates
- Microsoft Exchange Server and SQL Server updates
- Microsoft Visual Studio® updates
- Some non-Microsoft application updates

> **Tip**
>
> Many hardware manufacturers have placed their drivers on Windows Update. These drivers no longer need to be maintained in the Out-of-Box Drivers directory.

Experiment by removing drivers from the distribution share to see which ones are available on Windows Update. Note that if the drivers are not included with Windows by default, do not remove networking or storage drivers, because the operating system will require user input.

MDT supports the ability to deploy an updated version of WUA as part of the operating system deployment. This helps ensure that target computers are running the correct version of WUA when they are deployed. It also helps eliminate the need to connect to the Internet and download the latest version of WUA after deployment.

MDT can also configure WUA to collect updates from computers on the corporate network that are running WSUS instead of connecting to Microsoft Updates over the Internet. MDT can optionally configure WUA to use a specific computer running WSUS using the **WSUSServer** property.

For additional information and for WUA deployment instructions, see How to Install the Windows Update Agent on Client Computers.

Obtain the latest version of the WUA stand-alone installer for:

- x86 versions (WindowsUpdateAgent30-x86.exe) at https://go.microsoft.com/fwlink/?LinkID=100334
- x64 version (WindowsUpdateAgent30-x64.exe) at https://go.microsoft.com/fwlink/?LinkID=100335

  Windows 7 and later include the most recent version of WUA, so no upgrade is necessary.

  For more information, see Updating Windows Update Agent.

  When enabled in the Task Sequencer, this script runs multiple times while in the State Restore Phase of operating system deployment. It is first run after the operating system has started for the first time. Ensure that the latest updates and service packs are installed before the installation of any applications that might depend on specific updates or service packs being installed on the target computer. For example, an application might be dependent on the latest version of the Microsoft .NET Framework being installed.

  This script also runs after the installation of applications, which ensures that the latest application service packs and updates have been applied. For example, use this script to ensure that the latest updates are applied to Microsoft Office 2010 or the 2007 Office system.

It is possible, during the installation of one or more updates, the target computer will need to be restarted to allow an update installation to finish fully. To ensure that updates are properly installed, if the script detects that the installation of an update requires the target computer to be restarted, the script automatically restarts the target computer and resumes if additional updates have been detected and are pending installation. The script exits if it determines that the target computer is fully up to date. An error will be logged if, while updating the target computer, the script has seven unsuccessful attempts to install the updates and the target computer still requires a restart.

During run time, the script performs the following tasks:

- Configure the target computer to use a WSUS server, if the **WSUSServer** property was specified.
- Verify that the latest version of the WUA is installed on the target computer.
- Search the target computer for applicable updates that are not already installed and that might be typically hidden.
- Each update has an associated **UpdateID** and **QNumber** property:
    - The **UpdateID** property is in GUID form, such as *67da2176-5c57-4614-a514-33abbdd51f67*.
    - The **QNumber** property is a numerical value, such as *987654*.
- The script compares the **UpdateID** and **KBArticle** property values against the list of exclusions specified in the following MDT properties:
    - **WUMU_ExcludeID**. A list of UpdateIDs to exclude; any update with an **UpdateID** found in this list will not be installed.
    - **WUMU_ExcludeKB**. A list of **QNumbers** to exclude; any update with a **QNumber** found in this list will not be installed.
    - In addition, any update that requires user input will be excluded and not installed.
- All updates that require approval of an End User License Agreement (EULA) will automatically be approved by the script. Be sure to manually read and check each EULA before running this script in a production environment.
- The activity for each update is written to the ZTIWindowsUpdate.log file, with the string INSTALL or SKIP if the update has been approved for installation, along with

the UpdateID, a short description of the update, and the QNumber.

- Each update to be installed is downloaded and installed in batches.
- The target computer might require more than one restart during the update installation.

> **Note**
>
> Windows Internet Explorer 7 requires user interaction, so it is not installed using this script.

> **Note**
>
> By default, include **QNumber 925471** in the **WUMU_ExcludeKB** list to prevent Windows Vista Ultimate from installing extra language packs.

Expand table

Value	Description
Input	**Environment variables**. Contains the property values, custom property values, database connections, deployment rules, and other information that the scripts require to complete the deployment process
Output	- **ZTIWindowsUpdate.log**. Log file that contains events that this script generates - **BDD.log**. Log file that contains events that all MDT scripts generate
References	- **Expand.exe**. Expands compressed files - **Net.exe**. Performs network management tasks - **WindowsUpdateAgent30-x86.exe**. Installs WUA - **WindowsUpdateAgent30-x64.exe**. Installs WUA - **ZTIUtility.vbs**. Includes support functions and subroutines that the script uses
Location	*distribution*\Scripts
Use	`cscript ZTIWindowsUpdate.wsf </debug:value> </UpdateCommand:" <IsInstalled=0|1> <IsHidden=0|1>"> </Query:true|false>`

Arguments

Expand table

Value	Description
/debug:*value*	Outputs the event messages to the console and to the .log files. If the value specified in value is: - **TRUE**, event messages are sent to the console and the .log files - **FALSE**, event messages are sent only to the .log files (This is the behavior when the argument is not provided.)
/UpdateCommand:*param*	- **IsInstalled**. Set to **0** to query for updates that are not installed. - **IsHidden**. Set to **0** to query for updates that are hidden.
/Query:*value*	- **True**. Query only for required updates. Do not download and install any binaries. - **False**. Query for and install required updates. Download and install binaries.

> **Note**
>
> When specified, **UpdateCommand** requires at least one option.

> **Note**
>
> If specifying both options for **UpdateCommand**, they must be separated by *and*.

> **Note**
>
> The default value for **UpdateCommand** is **IsInstalled=0** and **IsHidden=0**.

> **Note**
>
> For more information about **UpdateCommand**, see IUpdateSearcher::Search Method.

Properties

Expand table

Name	Read	Write
Architecture	-	
DoCapture	-	
InstalledUpdates		-
MSIT_WU_Count	-	-
NoAutoUpdate_Previous	-	-
SMSTSRebootRequested	-	-
SMSTSRetryRequested	-	-
WSUSServer	-	
WUMU_ExcludeID	-	
WUMU_ExcludeKB	-	

ZTIWipeDisk.wsf

This script formats the target computer's hard disk. The script:

- Exits if **WipeDisk** is not equal to **TRUE**
- Determines the appropriate drive to format
- Formats the drive by calling `cmd /c format <Drive> /fs:ntfs /p:3 /Y` (where `<Drive>` is the drive letter of the hard disk drive to be formatted)

Expand table

Value	Description
Input	**Environment variables**. Contains the property values, custom property values, database connections, deployment rules, and other information that the scripts require to complete the deployment process
Output	- **ZTIWipeDisk.log**. Log file that contains events that this script generates - **BDD.log**. Log file that contains events that all MDT scripts generate
References	- **CMD.exe**. Allows running of command-line tools

Value	Description
	- **Format.com**. Formats the hard disk - **ZTIUtility.vbs**. Includes support functions and subroutines that the script uses
Location	*distribution*\Scripts
Use	`cscript ZTIWipeDisk.wsf </debug:value>`

Arguments

Expand table

Value	Description
/debug:*value*	Outputs the event messages to the console and to the .log files. If the value specified in value is: - **TRUE**, event messages are sent to the console and the .log files - **FALSE**, event messages are sent only to the .log files (This is the behavior when the argument is not provided.)

Properties

Expand table

Name	Read	Write
WipeDisk	-	

Related articles

- Task Sequence Steps.
- Properties.
- Support Files.
- Utilities.
- MDT Windows PowerShell Cmdlets.
- Tables and Views in the MDT DB.
- Windows 7 Feature Dependency Reference.
- UDI Reference.

Feedback

Was this page helpful? Yes No

Provide product feedback

Support Files

Article • 02/12/2024

The utilities and scripts used in LTI and ZTI deployments reference external configuration files to determine the process steps and configuration settings used during the deployment process.

The following information is provided for each utility:

- **Name**. Specifies the name of the file
- **Description**. Provides a description of the purpose of the file
- **Location**. Indicates the folder where the file can be found; in the information for the location, the following variables are used:
  - **program_files**. This variable points to the location of the Program Files folder on the computer where MDT is installed.
  - **distribution**. This variable points to the location of the Distribution folder for the deployment share.
  - **platform**. This variable is a placeholder for the operating system platform (x86 or x64).

ApplicationGroups.xml

> **Note**
>
> This XML file is managed by MDT and should not require modification.

Expand table

Value	Description
Location	*distribution*\Control

Applications.xml

> **Note**

This XML file is managed by MDT and should not require modification.

Expand table

Value	Description
Location	*distribution*\Control

BootStrap.ini

The configuration file used when the target computer is not able to connect to the appropriate deployment share. This situation occurs in the New Computer and the Replace Computer scenarios.

Expand table

Value	Description
Location	*distribution*\Control

CustomSettings.ini

The primary configuration file for the MDT processing rules used in all scenarios.

Expand table

Value	Description
Location	*distribution*\Control

Deploy.xml

> **Note**
>
> This XML file is managed by MDT and should not require modification.

Expand table

Value	Description
Location	*program_files*\Microsoft Deployment Toolkit\Control

DriverGroups.xml

> **Note**
>
> This XML file is managed by MDT and should not require modification.

Expand table

Value	Description
Location	*distribution*\Control

Drivers.xml

> **Note**
>
> This XML file is managed by MDT and should not require modification.

Expand table

Value	Description
Location	*distribution*\Control

LinkedDeploymentShares.xml

> **Note**
>
> This XML file is managed by MDT and should not require modification.

Expand table

Value	Description
Location	*distribution*\Scripts

ListOfLanguages.xml

> **Note**
>
> This XML file is managed by MDT and should not require modification.

Expand table

Value	Description
Location	*distribution*\Scripts

MediaGroups.xml

> **Note**
>
> This XML file is managed by MDT and should not require modification.

Expand table

Value	Description
Location	*distribution*\Scripts

Medias.xml

> **Note**
>
> This XML file is managed by MDT and should not require modification.

Expand table

Value	Description
Location	*distribution*\Scripts

OperatingSystemGroups.xml

> **Note**
>
> This XML file is managed by MDT and should not require modification.

Expand table

Value	Description
Location	*distribution*\Control

OperatingSystems.xml

> **Note**
>
> This XML file is managed by MDT and should not require modification.

Expand table

Value	Description
Location	*distribution*\Control

PackageGroups.xml

> **Note**
>
> This XML file is managed by MDT and should not require modification.

Expand table

Value	Description
Location	*distribution*\Control

Packages.xml

> **Note**
>
> This XML file is managed by MDT and should not require modification.

Expand table

Value	Description
Location	*distribution*\Control

SelectionProfileGroups.xml

> **Note**
>
> This XML file is managed by MDT and should not require modification.

Expand table

Value	Description
Location	*distribution*\Control

SelectionProfiles.xml

> **Note**
>
> This XML file is managed by MDT and should not require modification.

Expand table

Value	Description
Location	*distribution*\Control

ServerManager.xml

> **Note**
>
> This XML file is managed by MDT and should not require modification.

Expand table

Value	Description
Location	*program_files*\Microsoft Deployment Toolkit\Bin

Settings.xml

> **Note**
>
> This XML file is managed by MDT and should not require modification.

Expand table

Value	Description
Location	*distribution*\Control

TaskSequenceGroups.xml

> **Note**
>
> This XML file is managed by MDT and should not require modification.

Expand table

Value	Description
Location	*distribution*\Control

TaskSequences.xml

> **Note**
>
> This XML file is managed by MDT and should not require modification.

Expand table

Value	Description
Location	*distribution*\Control

TS.xml

> **Note**
>
> This XML file is managed by MDT and should not require modification.

Expand table

Value	Description
Location	*distribution*\Control*task_sequence_id*

> **Note**
>
> *Task_sequence_id* is a placeholder for the task sequence ID that was assigned to each task sequence when it was created in the Task Sequences node in the Deployment Workbench.

Wimscript.ini

This .ini file is an ImageX configuration file that contains the list of folders and files that will be excluded from an image. It is referenced by ImageX during the LTI Capture Phase.

For assistance with customizing this file, see the section, "Create an ImageX Configuration File," in the *Windows Preinstallation Environment (Windows PE) User's Guide*.

Expand table

Value	Description
Location	*distribution*\Tools*platform*

ZTIBIOSCheck.xml

This XML file contains metadata about BIOSes for target computers. This file is edited manually and is read by ZTIBIOSCheck.wsf. Extract the necessary information from a target computer to create an entry in this XML file using the Microsoft Visual Basic® Scripting Edition (VBScript) program (ZTIBIOS_Extract_Utility.vbs) that is embedded in this XML file.

Expand table

Value	Description
Location	*distribution*\Scripts

ZTIConfigure.xml

This XML file is used by the ZTIConfigure.wsf script to translate property values (specified earlier in the deployment process) to configure settings in the Unattend.xml file. This file is already customized to make the appropriate translations and should not require further modification.

Expand table

Value	Description
Location	*distribution*\Scripts

ZTIGather.xml

> **Note**

This XML file is preconfigured and should not require modification. Define custom properties in the CustomSettings.ini file or the MDT DB.

Expand table

Value	Description
Location	*distribution*\Scripts

ZTIUserState_config.xml

This XML file is used by the ZTIUserState.wsf script as a default USMT configuration file. This file is used by default if no custom configuration file is specified by the USMTConfigFile property. See the Config.xml File topic in the USMT documentation for more information on syntax and use.

Expand table

Value	Description
Location	*distribution*\Scripts

ZTITatoo.mof

This .mof file, when imported into the WMI repository of the target computer using Mofcomp.exe, creates the **Microsoft_BDD_Info** WMI class. This class contains deployment-related information, such as:

- DeploymentMethod
- DeploymentType
- DeploymentTimestamp
- BuildID
- BuildName
- BuildVersion
- OSDPackageID
- OSDProgramName

- OSDAdvertisementID
- TaskSequenceID
- TaskSequenceName
- TaskSequenceVersion

Expand table

Value	Description
Location	*distribution*\Scripts

Related articles

- Task Sequence Steps.
- Properties.
- Scripts.
- Utilities.
- MDT Windows PowerShell Cmdlets.
- Tables and Views in the MDT DB.
- Windows 7 Feature Dependency Reference.
- UDI Reference.

Feedback

Was this page helpful? Yes No

Provide product feedback

Utilities

Article • 02/12/2024

The scripts used in LTI and ZTI reference utilities that perform specialized tasks supporting the steps used during the deployment process. Use the following information to help determine the correct utilities to include in actions and the valid arguments to provide when running each utility.

The following information is provided for each utility:

- **Name**. Specifies the name of the utility
- **Description**. Provides a description of the purpose of the utility
- **Location**. Indicates the folder where the utility can be found; in the information for the location, the following variables are used:
  - **program_files**. This variable points to the location of the Program Files folder on the computer where MDT is installed.
  - **distribution**. This variable points to the location of the Distribution folder for the deployment share.
  - **platform**. This variable is a placeholder for the operating system platform (x86 or x64).
- **Use**.Provides the commands and options that can be specified
- **Arguments and description**.Indicates the valid arguments to be specified for the utility and a brief description of what each argument means

BCDBoot.exe

BCDBoot is a tool used to quickly set up a system partition or repair the boot environment located on the system partition. The system partition is set up by copying a small set of boot environment files from an installed Windows image. BCDBoot also creates a Boot Configuration Data (BCD) store on the system partition, with a new boot entry that enables Windows to boot to the installed Windows image.

Expand table

Value	Description
Location	Included in the Windows source files

Arguments

Expand table

Value	Description
	See the command-line help provided by this utility.

BDDRun.exe

This utility is run as an action by the Task Sequencer for executables (such as a script or other code) that require user interaction. By default, the task sequence cannot run an executable that requires user interaction. However, this utility allows the Task Sequencer to run an executable that requires user interaction.

The executable that requires user interaction is provided as an argument to this utility. This utility runs the executable in a separate command environment.

> **Note**
>
> This utility can only be used in LTI deployments. ZTI deployments prohibit any user interaction.

Expand table

Value	Description
Location	*distribution*\Tools*platform*
Use	`BDDRun.exe commandline`

Arguments

Expand table

Value	Description
commandline	The command to be run that requires user interaction

> **Note**
>
> Put double quotation marks around any part of the *command-line* portion of the argument that contains blanks. For example: `BDDRun.exe MyAppInstall.exe /destinationdir: "%ProgramFiles%\AppName"`.

Bootsect.exe

Bootsect.exe updates the master boot code for hard disk partitions to switch between BOOTMGR and NTLDR. Use this utility to restore the boot sector on the computer.

For more information on Bootsect.exe, see the section, "Bootsect Command-Line Options," in the *Windows Preinstallation Environment (Windows PE) User's Guide*.

Expand table

Value	Description
Location	*distribution*\Tools*platform*
Use	`bootsect.exe /nt52 C:`

Arguments

Expand table

Value	Description
/Help	Displays the use instructions listed here.
/nt52	Applies the master boot code compatible with NTLDR to **SYS**, **ALL**, or *DriveLetter*. The operating system installed on **SYS**, **ALL**, or *DriveLetter* must be an earlier version of Windows Vista.
/nt60	Applies the master boot code compatible with BOOTMGR to **SYS**, **ALL**, or *DriveLetter*. The operating system installed on **SYS**, **ALL**, or *DriveLetter* must be Windows Vista.
SYS	Updates the master boot code on the system partition used to boot Windows.

Value	Description
All	Updates the master boot code on all partitions. **ALL** does not necessarily update the boot code for each volume. Instead, this option updates the boot code on volumes that can be used as Windows boot volumes, which excludes any dynamic volumes not connected with an underlying disk partition. This restriction is present, because the boot code must be located at the beginning of a disk partition.
DriveLetter	Updates the master boot code on the volume associated with this drive letter. The boot code will not be updated if either (1) **DriveLetter** is not associated with a volume or (2) **DriveLetter** is associated with a volume not connected to an underlying disk partition.
/Force	Forcibly dismounts the volumes during the boot code update. Use this option with caution.

Compact.exe

Displays or alters the compression of files on NTFS file system partitions.

Expand table

Value	Description
Location	Included in the Windows source files

Arguments

Expand table

Value	Description
/C	Compresses the specified files. Directories will be marked so that files added afterward will be compressed.
/V	Decompresses the specified files. Directories will be marked so that files added afterward will not be compressed.
/S	Performs the specified operation on files in the given directory and in all subdirectories. Default dir is the current directory.
/A	Displays files with the hidden or system attributes. These files are omitted by default.
/I	Continues performing the specified operation even after errors have occurred. By default, Compact.exe stops when an error is encountered.

Value	Description
/F	Forces the compress operation on all specified files, even those which are already compressed. Already-compressed files are skipped by default.
/Q	Reports only the most essential information.
filename	Specifies a pattern, file, or directory.

Diskpart.exe

Diskpart is a text-mode command interpreter that allows management of objects (disks, partitions, or volumes) using scripts or direct input in a Command Prompt window.

For more information on Diskpart.exe, see the section, "Diskpart Command-Line Options," in the *Windows Preinstallation Environment (Windows PE) User's Guide*.

Expand table

Value	Description
Location	Included in the Windows PE source files

Arguments

Expand table

Value	Description
	See the guide referenced in the utility description.

Expand.exe

This utility is run to expand (extract) files from compressed files.

Expand table

Value	Description
Location	Included in the Windows source files
Use	`Expand.exe -r wuredist.cab -F:wuRedist.xml %temp%`

Arguments

Expand table

Value	Description
-r	Renames expanded files
-D	Displays the list of files in the source directory
Source	Source file specification (Wildcards can be used.)
-F:*Files*	Name of files to expand from a .cab file
Destination	Destination file | path specification (**Destination** can be a directory. If **Source** is multiple files and **-r** is not specified, **Destination** must be a directory.)

ImageX.exe

ImageX is a command-line utility that enables OEMs and corporations to capture, modify, and apply file-based disk images for rapid deployment. ImageX works with WIM files for copying to a network, or it can work with other technologies that use WIM images, such as Windows Setup and Windows Deployment Services.

For more information about ImageX, see the section, "What is ImageX," in the *Windows Preinstallation Environment (Windows PE) User's Guide*.

Expand table

Value	Description
Location	*distribution*\Tools*platform*

Arguments

Expand table

Value	Description
	See the guide referenced in the utility description.

Microsoft.BDD.PnpEnum.exe

This utility is run to enumerate Plug and Play devices installed on the target computer.

Expand table

Value	Description
Location	*distribution*\Tools*platform*

Arguments

Expand table

Value	Description
None	-

Mofcomp.exe

Mofcomp.exe is the Managed Object Format compiler that parses a file that contains Managed Object Format statements and adds the classes and class instances defined in the file to the WMI repository. Mofcomp.exe provides command-line help on the switch use options.

Expand table

Value	Description
Location	Included in the Windows source files

Arguments

Expand table

Value	Description
	See the command-line help that this utility provides.

Netsh.exe

Netsh.exe is a command-line and scripting utility used to automate the configuration of networking components. For more information about Netsh.exe, see The Netsh

Command-Line Utility.

Expand table

Value	Description
Location	Included in the Windows source files

Arguments

Expand table

Value	Description
	See the command-line help that this utility provides or the information found at the URL listed in the utility description.

Reg.exe

The Console Registry Tool is used to read and modify registry data.

Expand table

Value	Description
Location	Included in the Windows source files

Arguments

Expand table

Value	Description
	See the command-line help that this utility provides.

Regsvr32.exe

This utility is used to register files (.dll, .exe, .ocx, and so on) with the operating system.

Expand table

Value	Description
Location	Included in the Windows source files

Arguments

Expand table

Value	Description
file	The name of the file to register or unregister
/s	Runs the utility in silent mode
/u	Unregisters the file

Wpeutil.exe

The Windows PE utility (Wpeutil) is a command-line utility with which various commands can be run in a Windows PE session. For example, an administrator can shut down or reboot Windows PE, activate or deactivate a firewall, configure language settings, and initialize a network. MDT uses the utility to initialize Windows PE and network connections, and start LTI deployments.

For more information on Wpeutil.exe, see the section, "Wpeutil Command-Line Options," in the *Windows Preinstallation Environment (Windows PE) User's Guide*.

Expand table

Value	Description
Location	Included in the Windows PE source files

Arguments

Expand table

Value	Description
	See the guide referenced in the utility description.

Related articles

- Task Sequence Steps.
- Properties.
- Scripts.
- Support Files.
- MDT Windows PowerShell Cmdlets.
- Tables and Views in the MDT DB.
- Windows 7 Feature Dependency Reference.
- UDI Reference.

Feedback

Was this page helpful? Yes No

Provide product feedback

MDT Windows PowerShell Cmdlets

Article • 02/12/2024

In addition to the Deployment Workbench, MDT deployment shares can be managed using Windows PowerShell cmdlets. The MDT Windows PowerShell cmdlets are included in a Windows PowerShell snap-in—Microsoft.BDD.PSSnapIn—which is included with the installation of MDT.

The MDT cmdlets must be run from a Windows PowerShell console that has the MDT Windows PowerShell snap-in loaded. For more information on how to start a Windows PowerShell console that has the MDT Windows PowerShell snap-in loaded, see "Loading the MDT Windows PowerShell Snap-In".

Table 7 lists the MDT Windows PowerShell cmdlets and provides a brief description of each cmdlet. Each cmdlet is discussed in further detail in a subsequent section.

Table 7. MDT Windows PowerShell Cmdlets

Expand table

Cmdlet	Description
Add-MDTPersistentDrive	Adds a deployment share to the list of MDT persisted drives that can be restored using the Restore-MDTPersistentDrive cmdlet.
Disable-MDTMonitorService	Disables the MDT monitoring services.
Enable-MDTMonitorService	Enables the MDT monitoring services.
Get-MDTDeploymentShareStatistics	Displays the statistics of a deployment share, including the number of entities per major folder in the deployment share.
Get-MDTMonitorData	Displays the MDT monitoring information collected for one or more monitored MTD deployments.
Get-MDTOperatingSystemCatalog	Returns the operating system catalog for a specific operating system. If the operating system catalog does not exist or is out of date, then the operating system catalog is regenerated.
Get-MDTPersistentDrive	Displays the list of deployment shares that can be restored using the Restore-MDTPersistentDrive cmdlet.
Import-MDTApplication	Imports an application into a deployment share.

Cmdlet	Description
Import-MDTDriver	Imports one or more device drivers into a deployment share.
Import-MDTOperatingSystem	Imports one or more operating systems into a deployment share.
Import-MDTPackage	Imports one or more operating system packages into a deployment share.
Import-MDTTaskSequence	Imports a task sequence into a deployment share.
New-MDTDatabase	Creates or upgrades an MDT DB database that is associated with a deployment share.
Remove-MDTMonitorData	Removes one or more MDT monitoring data items from the collected MDT monitoring data in a deployment share.
Remove-MDTPersistentDrive	Removes a deployment share from the list of MDT persisted Windows PowerShell drives that can be restored using the Restore-MDTPersistentDrive cmdlet.
Restore-MDTPersistentDrive	Creates a Windows PowerShell drive for each deployment share in the list of MDT persisted Windows PowerShell drives.
Set-MDTMonitorData	Creates a new or updates an existing MDT monitoring data item in the collected MDT monitoring data in a deployment share.
Test-MDTDeploymentShare	Verifies the integrity of a deployment share.
Test-MDTMonitorData	Verifies that the MDT monitoring services is configured correctly and running.
Update-MDTDatabaseSchema	Updates the MDT DB database schema.
Update-MDTDeploymentShare	Updates a deployment share.
Update-MDTLinkedDS	Replicates content from a deployment share to a linked deployment share.
Update-MDTMedia	Replicates content from a deployment share to a deployment media folder.

Add-MDTPersistentDrive

This section describes the **Add-MDTPersistentDriveWindows PowerShell** cmdlet. Run this cmdlet from a Windows PowerShell console that has the MDT PowerShell snap-in loaded. For more information on how to start a Windows PowerShell console that has

the MDT PowerShell snap-in loaded, see "Loading the MDT Windows PowerShell Snap-In".

Syntax

PowerShell

```
Add-MDTPersistentDrive [-Name] <String> [[-InputObject] <PSObject>]
[<CommonParameters>]
```

Description

This cmdlet adds an existing Windows PowerShell drive created using the **MDTProvider** to a list of drives that are persisted in the Deployment Workbench or in a Windows PowerShell session using the Restore-MDTPersistentDrive cmdlet. This cmdlet is called when you create or open a deployment share in the Deployment Workbench.

> ⓘ **Note**
>
> The list of persisted **MDTProvider** drives is maintained on a per-user based in the user profile.

The list of persisted **MDTProvider** drives can be displayed using the **Get-MDTPersistentDrive** cmdlet.

Parameters

This subsection provides information about the various parameters that can be used with the **Add-MDTPersistentDriveWindows** cmdlet.

-Name <String>

Specifies the name of a Windows PowerShell drive created using the MDT provider and corresponds to an existing deployment share. The name was created using the New-PSDrive cmdlet and specifying the **MDTProvider** in the *PSProvider* parameter.

For more information on how to create a new Windows PowerShell drive using the **MDTProvider** and how to create a deployment share using Windows PowerShell, see the section "Creating a Deployment Share Using Windows PowerShell" in the MDT document, *Microsoft Deployment Toolkit Samples Guide*.

Expand table

Parameter	Value
Required?	True
Position?	2 and Named
Default value	None
Accept pipeline input?	True (ByValue)
Accept wildcard characters?	False

-InputObject <PSObject>

This parameter specifies a Windows PowerShell drive object that was created earlier in the process. Enter a PSObject object, such as one generated by the New-PSDrive cmdlet.

Expand table

Parameter	Value
Required?	False
Position?	3 and Named
Default value	-
Accept pipeline input?	True (ByValue)
Accept wildcard characters?	False

<CommonParameters>

This cmdlet supports the following common parameters: *Verbose, Debug, ErrorAction, ErrorVariable, OutBuffer, OutVariable, WarningAction,* and *WarningVariable*. For more information, see the topic, "about_CommonParameters," which you can access by typing the following command, and then pressing ENTER:

PowerShell

```
Get-Help about_CommonParameters
```

Outputs

This cmdlet outputs a **PSObject** type object for the Windows PowerShell drive object was added to the list of persisted drives.

This cmdlet also outputs a **String** type object if the *Verbose* common parameter is included.

Example 1

PowerShell

```
Add-MDTPersistentDrive -Name DS001
```

Description

This example adds the deployment share with the Windows PowerShell drive name of *DS001* to the list of persisted drives.

Example 2

PowerShell

```
$MDTPSDrive = New-PSDrive -Name "DS001" -PSProvider "MDTProvider" -Root
"C:\DeploymentShare$" -Description "MDT Deployment Share" -NetworkPath
\\WDG-MDT-01\DeploymentShare$ -Verbose
Add-MDTPersistentDrive -InputObject $MDTPSDrive
```

Description

This example adds the Windows PowerShell drive name *DS001*, created by the New-PSDrive cmdlet, to the list of persisted MDT drives using the *$MDTPSDrive* variable.

Example 3

PowerShell

```
New-PSDrive -Name "DS001" -PSProvider "MDTProvider" -Root
"C:\DeploymentShare$" -Description "MDT Deployment Share" -NetworkPath
\\WDG-MDT-01\DeploymentShare$ -Verbose | Add-MDTPersistentDrive -Verbose
```

Description

This example adds the Windows PowerShell drive name *DS001*, created by the New-PSDrive cmdlet, to the list of persisted MDT drives by piping the newly created Windows PowerShell drive object to the **Add-MDTPersistentDrive** cmdlet.

Disable-MDTMonitorService

This section describes the **Disable-MDTMonitorService** Windows PowerShell cmdlet. Run this cmdlet from a Windows PowerShell console that has the MDT PowerShell snap-in loaded. For more information on how to start a Windows PowerShell console that has the MDT PowerShell snap-in loaded, see "Loading the MDT Windows PowerShell Snap-In".

Syntax

```
PowerShell

Disable-MDTMonitorService [<CommonParameters>]
```

Description

This cmdlet disables the MDT monitoring service, which runs on the computer where MDT is installed. The MDT monitoring service collects monitoring information that can be displayed:

- In the Monitoring node in a deployment share in the Deployment Workbench
- Using the Get-MDTMonitorData cmdlet

  The MDT monitoring service can subsequently be enabled using the Enable-MDTMonitorService.

  For more information on the MDT monitoring service, see the section "Monitoring MDT Deployments" in the MDT document, *Using the Microsoft Deployment Toolkit*.

Parameters

This subsection provides information about the various parameters that can be used with the **Disable-MDTMonitorService** cmdlet.

<CommonParameters>

This cmdlet supports the following common parameters: *Verbose, Debug, ErrorAction, ErrorVariable, OutBuffer, OutVariable, WarningAction,* and *WarningVariable*. For more information, see the topic, "about_CommonParameters," which you can accessed by typing the following command, and then pressing ENTER:

```
PowerShell

Get-Help about_CommonParameters
```

Outputs

This cmdlet outputs a **String** type object if the *Verbose* common parameter is included; otherwise, no output is generated.

Example 1

```
PowerShell

Disable-MDTMonitorService
```

Description

This example disables the MDT monitoring service.

Enable-MDTMonitorService

This section describes the **Enable-MDTMonitorService** Windows PowerShell cmdlet. Run this cmdlet from a Windows PowerShell console that has the MDT PowerShell snap-in loaded. For more information on how to start a Windows PowerShell console that has the MDT PowerShell snap-in loaded, see "Loading the MDT Windows PowerShell Snap-In".

Syntax

```
PowerShell

Enable-MDTMonitorService [-EventPort] <Int32> [-DataPort] <Int32>
[<CommonParameters>]
```

Description

This cmdlet enables the MDT monitoring service, which runs on the computer where MDT is installed. The MDT monitoring service collects monitoring information that can be displayed:

- In the Monitoring node in a deployment share in the Deployment Workbench.
- Using the Get-MDTMonitorData cmdlet

  The MDT monitoring service can be disabled using the Disable-MDTMonitorService.

  For more information on the MDT monitoring service, see the section "Monitoring MDT Deployments" in the MDT document, *Using the Microsoft Deployment Toolkit*.

Parameters

This subsection provides information about the various parameters that can be used with the **Enable-MDTMonitorService** cmdlet.

-EventPort <Int32>

This parameter specifies the TCP port used as the event port for the MDT monitoring service.

Expand table

Parameter	Value
Required?	**False**
Position?	**2** and **Named**
Default value	**9800**
Accept pipeline input?	**False**
Accept wildcard characters?	**False**

-DataPort <Int32>

This parameter specifies the TCP port used as the data port for the MDT monitoring service.

Expand table

Parameter	Value
Required?	False
Position?	3 and **Named**
Default value	9801
Accept pipeline input?	False
Accept wildcard characters?	False

<CommonParameters>

This cmdlet supports the following common parameters: *Verbose, Debug, ErrorAction, ErrorVariable, OutBuffer, OutVariable, WarningAction,* and *WarningVariable*. For more information, see the topic, "about_CommonParameters," which you can access by typing the following command, and then pressing ENTER:

PowerShell

```
Get-Help about_CommonParameters
```

Outputs

This cmdlet outputs a **String** type object if the *Verbose* common parameter is included; otherwise, no output is generated.

Example 1

PowerShell

```
Enable-MDTMonitorService
```

Description

This example enables the MDT monitoring service on the local computer using the default value of **9800** for the event port and the value of **9801** for the data port on the MDT monitoring service.

Example 2

```
PowerShell
```

```
Enable-MDTMonitorService -EventPort 7000 -DataPort 7001
```

Description

This example enables the MDT monitoring service on the local computer using the value of **7000** for the event port and the value of **7001** for the data port on the MDT monitoring service.

Get-MDTDeploymentShareStatistics

This section describes the **Get-MDTDeploymentShareStatistics** Windows PowerShell cmdlet. Run this cmdlet from a Windows PowerShell console that has the MDT PowerShell snap-in loaded. For more information on how to start a Windows PowerShell console that has the MDT PowerShell snap-in loaded, see "Loading the MDT Windows PowerShell Snap-In".

Syntax

```
PowerShell
```

```
Get-MDTDeploymentShareStatistics [-Path <String>] [<CommonParameters>]
```

Description

This cmdlet displays the statistics of a deployment share based on the MDTProvder drive that is specified in the *Path* parameter. The statistics include the number of items in the specified deployment share:

- Applications
- Drivers
- Operating Systems
- Packages
- Task Sequences

- Selection Profiles
- Linked Deployment Shares
- MDT Media
- Computers in the MDT DB
- Make and Models in the MDT DB
- Locations in the MDT DB
- Roles in the MDT DB

> **Note**
>
> The values for the statistics that relate to the MDT DB are not populated and always return a value of zero.

Parameters

This subsection provides information about the various parameters that can be used with the **Get-MDTDeploymentShareStatistics** cmdlet.

-Path <String>

This parameter specifies the MDTProvider Windows PowerShell drive for the desired deployment share.

> **Note**
>
> If this parameter is not provided, then the Windows PowerShell working directory must default to a location within the desired MDTProvider Windows PowerShell drive.

Expand table

Parameter	Value
Required?	**False**
Position?	**2** and **Named**

Parameter	Value
Default value	-
Accept pipeline input?	False
Accept wildcard characters?	False

<CommonParameters>

This cmdlet supports the following common parameters: *Verbose, Debug, ErrorAction, ErrorVariable, OutBuffer, OutVariable, WarningAction,* and *WarningVariable*. For more information, see the topic, "about_CommonParameters," which you can access by typing the following command, and then pressing ENTER:

```
PowerShell

Get-Help about_CommonParameters
```

Outputs

This cmdlet outputs a **PSObject** type object that contains the statistics for the deployment share.

Example 1

```
PowerShell

Get-MDTDeploymentShareStatistics -Path DS001:
```

Description

This example returns the deployment share statistics for the deployment share that is specified in the DS001: MDTProvider Windows PowerShell drive.

Example 2

```
PowerShell

cd DS001:
Get-MDTDeploymentShareStatistics
```

Description

This example returns the deployment share statistics for the deployment share that is specified in the DS001: MDTProvider Windows PowerShell drive. Use the **cd** command to set the working directory for Windows PowerShell to the DS001: MDTProvider Windows PowerShell drive.

Get-MDTMonitorData

This section describes the **Get-MDTMonitorData** Windows PowerShell cmdlet. Run this cmdlet from a Windows PowerShell console that has the MDT PowerShell snap-in loaded. For more information on how to start a Windows PowerShell console that has the MDT PowerShell snap-in loaded, see "Loading the MDT Windows PowerShell Snap-In".

Syntax

PowerShell

```
Get-MDTMonitorData [-Path <String>] [-ID <Nullable>] [<CommonParameters>]
```

Description

This cmdlet displays the MDT monitoring data that is being reported to the deployment share that is specified in the *Path* parameter. The following is example output from this cmdlet:

PowerShell

```
Name               : WDG-REF-01
PercentComplete    : 100
Settings           :
Warnings           : 0
Errors             : 0
DeploymentStatus   : 3
StartTime          : 5/23/2012 6:45:39 PM
EndTime            : 5/23/2012 8:46:32 PM
ID                 : 1
UniqueID           : 94a0830e-f2bb-421c-b1e0-6f86f9eb9fa1
CurrentStep        : 88
TotalSteps         : 88
StepName           :
LastTime           : 5/23/2012 8:46:32 PM
DartIP             :
DartPort           :
```

```
DartTicket          :
VMHost              : WDG-HOST-01
VMName              : WDG-REF-01
ComputerIdentities : {}
```

> **Note**
>
> The MDTProvider Windows PowerShell drive that this cmdlet references must exist prior to running this cmdlet.

Parameters

This subsection provides information about the various parameters that you can use with the **Get- MDTMonitorData** cmdlet.

-Path <String>

This parameter specifies the MDTProvider Windows PowerShell drive for the desired deployment share.

> **Note**
>
> If this parameter is not provided, then the Windows PowerShell working directory must default to a location within the desired MDTProvider Windows PowerShell drive.

Expand table

Parameter	Value
Required?	**False**
Position?	**2** and **Named**
Default value	-
Accept pipeline input?	**False**
Accept wildcard characters?	**False**

-ID <Nullable>

This parameter specifies the specific identifier for the deployment of a specific computer. If this parameter is not specified, then all monitoring data for deployments in the deployment share are displayed.

Expand table

Parameter	Value
Required?	**False**
Position?	**3** and **Named**
Default value	-
Accept pipeline input?	**False**
Accept wildcard characters?	**False**

<CommonParameters>

This cmdlet supports the following common parameters: *Verbose, Debug, ErrorAction, ErrorVariable, OutBuffer, OutVariable, WarningAction,* and *WarningVariable*. For more information, see the topic, "about_CommonParameters," which you can access by typing the following command, and then pressing ENTER:

```
PowerShell

Get-Help about_CommonParameters
```

Outputs

This cmdlet outputs a **PSObject** type object for each monitored computer, which contains the monitoring data for the computer.

Example 1

```
PowerShell

Get-MDTMonitorData -Path DS001:
```

Description

This example returns the monitoring data for all deployments in the deployment share that is specified in the DS001: MDTProvider Windows PowerShell drive.

Example 2

PowerShell

```
cd DS001:
Get-MDTMonitorData
```

Description

This example returns the monitoring data for all deployments in the deployment share that is specified in the DS001: MDTProvider Windows PowerShell drive. Use the **cd** command to set the working directory for Windows PowerShell to the DS001: MDTProvider Windows PowerShell drive.

Example 3

PowerShell

```
Get-MDTMonitorData -Path DS001: -ID 22
```

Description

This example returns the monitoring data for the deployment with an ID of 22 in the deployment share that is specified in the DS001: MDTProvider Windows PowerShell drive.

Get-MDTOperatingSystemCatalog

This section describes the **Get-MDTOperatingSystemCatalog** Windows PowerShell cmdlet. Run this cmdlet from a Windows PowerShell console that has the MDT PowerShell snap-in loaded. For more information on how to start a Windows PowerShell console that has the MDT PowerShell snap-in loaded, see "Loading the MDT Windows PowerShell Snap-In".

Syntax

```
```

PowerShell

```
Get-MDTOperatingSystemCatalog [-ImageFile] <String> [-Index] <Int32>
[<CommonParameters>]
```

Description

This cmdlet retrieves or creates an operating system catalog for a custom operating system image so that you can modify the corresponding unattend.xml file using Windows System Image Manager (WSIM). If no operating system catalog is available or if the existing operating system catalog is invalid or out of date, this cmdlet will generate a new operating system catalog.

> **Note**
>
> The process of generating a new operating system catalog may take a long time as the custom operating system image must be mounted, inspected, and unmounted before the operating system catalog creation completes.

Parameters

This subsection provides information about the various parameters that can be used with the **Get-MDTOperatingSystemCatalog** cmdlet.

-ImageFile <String>

This parameter specifies the fully qualified path to the custom operating system image file (.wim file), including the name of the custom operating system image file.

Expand table

Parameter	Value
Required?	**True**
Position?	**2** and **Named**
Default value	-
Accept pipeline input?	**False**
Accept wildcard characters?	**False**

-Index <Int32>

This parameter specifies the index of the desired operating system image within the operating system image file (.wim file).

Expand table

Parameter	Value
Required?	**True**
Position?	**3** and **Named**
Default value	-
Accept pipeline input?	**False**
Accept wildcard characters?	**False**

<CommonParameters>

This cmdlet supports the following common parameters: *Verbose, Debug, ErrorAction, ErrorVariable, OutBuffer, OutVariable, WarningAction,* and *WarningVariable*. For more information, see the topic, "about_CommonParameters," which you can access by typing the following command, and then pressing ENTER:

PowerShell

```
Get-Help about_CommonParameters
```

Outputs

This cmdlet outputs a **PSObject** type object that contains the path to the operating system catalog.

Example 1

PowerShell

```
Get-MDTOperatingSystemCatalog -ImageFile "DS001:\Operating Systems\Windows 8\sources\install.wim" -Index 2
```

Description

This example returns the operating system catalog for the operating system image at the index of 2 in the operating system image file DS001:\Operating Systems\Windows 8\sources\install.wim.

Get-MDTPersistentDrive

This section describes the **Get-MDTPersistentDrive** Windows PowerShell cmdlet. Run this cmdlet from a Windows PowerShell console that has the MDT PowerShell snap-in loaded. For more information on how to start a Windows PowerShell console that has the MDT PowerShell snap-in loaded, see "Loading the MDT Windows PowerShell Snap-In".

Syntax

PowerShell

```
Get-MDTPersistentDrive [<CommonParameters>]
```

Description

This cmdlet displays the list of persisted MDT Windows PowerShell drives. The list of persisted MDT Windows PowerShell drives is managed using the Add-MDTPersistentDrive and Remove-MDTPersistentDrive cmdlets or the Deployment Workbench.

The output from this cmdlet contains the following information:

- Windows PowerShell drive name, such as DS001
- Directory path, such as \\WDG-MDT-01\DeploymentShare$

  Persisted MDT Windows PowerShell drives are similar to persisted network drive mappings.

> ⓘ **Note**
>
> This list of persisted MDT Windows PowerShell drives is maintained on a per user basis and are stored in the user profile.

Parameters

This subsection provides information about the various parameters that can be used with the **Get- MDTPersistentDrive** cmdlet.

<CommonParameters>

This cmdlet supports the following common parameters: *Verbose, Debug, ErrorAction, ErrorVariable, OutBuffer, OutVariable, WarningAction,* and *WarningVariable*. For more information, see the topic, "about_CommonParameters," which you can access by typing the following command, and then pressing ENTER:

```
PowerShell

Get-Help about_CommonParameters
```

Outputs

This cmdlet outputs a **PSObject** type object for each MDT persisted drive that is identical to the **PSObject** type object that the New-PSDrive cmdlet returns.

Example 1

```
PowerShell

Get-MDTPersistentDrive
```

Description

This example displays a list of the MDT persisted drives.

Import-MDTApplication

This section describes the **Import-MDTApplication** Windows PowerShell cmdlet. Run this cmdlet from a Windows PowerShell console that has the MDT PowerShell snap-in loaded. For more information on how to start a Windows PowerShell console that has the MDT PowerShell snap-in loaded, see "Loading the MDT Windows PowerShell Snap-In".

Syntax

```
```

PowerShell

```
Import-MDTApplication [-Path <String>] -Name <String> ApplicationSourcePath
<String> -DestinationFolder <String> [-Move] [<CommonParameters>]
```

-or-

PowerShell

```
Import-MDTApplication [-Path <String>] -Name <String> NoSource
[<CommonParameters>]
```

-or-

PowerShell

```
Import-MDTApplication [-Path <String>] -Name <String> Bundle
[<CommonParameters>]
```

Description

This cmdlet imports an application into a deployment share. The following application types can be imported using this cmdlet:

- Applications that have source files, using the *ApplicationSourcePath*, *DestinationFolder*, and *Move* parameters. The first syntax example illustrates the use of this cmdlet for this type of application.
- Applications without source files or with source files located on other network shared folders using the *NoSource* parameter. The second syntax example illustrates the use of this cmdlet for this type of application.
- Application bundles, which are used to group a set of related applications, using the *Bundle* parameter. The last syntax example illustrates the use of this cmdlet for this type of application.

Parameters

This subsection provides information about the various parameters that can be used with the **Import-MDTApplication** cmdlet.

-Path <String>

This parameter specifies the fully qualified path to an existing folder where the application being imported will be placed within the deployment share. If the *DestinationFolder* parameter is used, then the folder specified in the *DestinationFolder* parameter is created beneath the folder specified in this parameter. This parameter is used in all syntax usages for this cmdlet.

> **Note**
>
> If this parameter is not provided, the Windows PowerShell working directory must default to the desired location within the deployment share.

Expand table

Parameter	Value
Required?	False
Position?	Named
Default value	-
Accept pipeline input?	False
Accept wildcard characters?	False

-Name <String>

This parameter specifies the name of the application to be added to the deployments share and must be unique within the deployment share. This parameter is used in all syntax usages for this cmdlet.

Expand table

Parameter	Value
Required?	True
Position?	Named
Default value	-
Accept pipeline input?	False
Accept wildcard characters?	False

-ApplicationSourcePath <String>

This parameter specifies the fully qualified path to the application source files for the application that will be imported into the deployment share. This parameter is only valid for use in the first syntax example.

Expand table

Parameter	Value
Required?	True
Position?	Named
Default value	-
Accept pipeline input?	False
Accept wildcard characters?	False

-DestinationFolder <String>

This parameter specifies the folder in the deployment share where the application source files are to be imported. This folder is created beneath the folder specified in the *Path* parameter. This parameter is only valid for use in the first syntax example.

Expand table

Parameter	Value
Required?	True
Position?	Named
Default value	-
Accept pipeline input?	False
Accept wildcard characters?	False

-Move [<SwitchParameter>]

This parameter specifies whether the application's source files should be moved (instead of copied) from the folder where the application's source files are located, which is specified in the *ApplicationSourcePath* parameter.

If this parameter is:

- Specified, then the files are moved and the files in the folder specified in the *ApplicationSourcePath* parameter are deleted
- Not specified, then the files are copied and the files in the folder specified in the *ApplicationSourcePath* parameter are retained

  This parameter is only valid for use in the first syntax example.

Expand table

Parameter	Value
Required?	False
Position?	Named
Default value	-
Accept pipeline input?	False
Accept wildcard characters?	False

-NoSource [<SwitchParameter>]

This parameter specifies that the application being imported is an application that has no source files to be copied. When using this parameter, the application source files are:

- On a network shared folder, which is specified in the application installation command line or working directory configuration settings
- Already present in the operating system image

  This parameter is only valid for use in the second syntax example.

Expand table

Parameter	Value
Required?	False
Position?	Named
Default value	-
Accept pipeline input?	True (ByValue)

Parameter	Value
Accept wildcard characters?	False

-Bundle [<SwitchParameter>]

This parameter specifies that the application being imported is an application that is a bundle of two or more applications. This parameter is only valid for use in the last syntax example.

Expand table

Parameter	Value
Required?	False
Position?	Named
Default value	-
Accept pipeline input?	True (ByValue)
Accept wildcard characters?	False

<CommonParameters>

This cmdlet supports the following common parameters: *Verbose, Debug, ErrorAction, ErrorVariable, OutBuffer, OutVariable, WarningAction,* and *WarningVariable*. For more information, see the topic, "about_CommonParameters," which you can access by typing the following command, and then pressing ENTER:

```
PowerShell

Get-Help about_CommonParameters
```

Outputs

This cmdlet outputs a **PSObject** type object that references the application just imported.

Example 1

```
PowerShell
```

```
Import-MDTApplication -Path "DS001:\Applications" -Name "Office 2010
Professional Plus 32-bit" ApplicationSourcePath "\\WDG-MDT-
01\Source$\Office2010ProPlus\x86" DestinationFolder "Office2010ProPlusx86"
```

Description

This example imports an application with source files from the network shared folder at \\WDG-MDT-01\Source$\Office2010ProPlus\x86 and copies the source files to DS001:\Applications\Office2010ProPlusx86 within the deployment share. The source files are retained.

Example 2

PowerShell

```
Import-MDTApplication -Path "DS001:\Applications" -Name "Office 2010
Professional Plus 32-bit" ApplicationSourcePath "\\WDG-MDT-
01\Source$\Office2010ProPlus\x86" DestinationFolder "Office2010ProPlusx86" -
Move
```

Description

This example imports an application with source files from the network shared folder at \\WDG-MDT-01\Source$\Office2010ProPlus\x86 and moves the source files to DS001:\Applications\Office2010ProPlusx86 within the deployment share. The source files are removed from the network shared folder at \\WDG-MDT-01\Source$\Office2010ProPlus\x86. The application is named *Office 2012 Professional Plus 32-bit*.

Example 3

PowerShell

```
Import-MDTApplication -Path "DS001:\Applications" -Name "Office 2010
Professional Plus 32-bit" NoSource
```

Description

This example imports an application named *Office 2012 Professional Plus 32-bit* with no source files.

Example 4

```
PowerShell

Import-MDTApplication -Path "DS001:\Applications" -Name "Woodgrove Bank Core
Applications" Bundle
```

Description

This example imports an application bundle named *Woodgrove Bank Core Applications*.

Import-MDTDriver

This section describes the **Import-MDTDriver** Windows PowerShell cmdlet. Run this cmdlet from a Windows PowerShell console that has the MDT PowerShell snap-in loaded. For more information on how to start a Windows PowerShell console that has the MDT PowerShell snap-in loaded, see "Loading the MDT Windows PowerShell Snap-In".

Syntax

```
PowerShell

Import-MDTDriver [-Path <String>] -SourcePath <String[]> [ImportDuplicates]
[<CommonParameters>]
```

Description

This cmdlet imports one or more device drivers into a deployment share. This cmdlet searches for device drivers starting at the folder specified in the *SourcePath* parameter. This cmdlet will locate multiple device drivers found in that folder structure.

Parameters

This subsection provides information about the various parameters that can be used with the **Import-MDTDriver** cmdlet.

-Path <String>

This parameter specifies the fully qualified path to an existing folder where the device driver being imported will be placed within the deployment share.

> **Note**
>
> If this parameter is not provided, then the Windows PowerShell working directory must default to the desired location within the deployment share. This parameter must be provided if the *SourcePath* parameter is not provided.

Expand table

Parameter	Value
Required?	False
Position?	Named
Default value	-
Accept pipeline input?	False
Accept wildcard characters?	False

-SourcePath <String[ ]>

This parameter specifies one or more fully qualified paths in a string array for the source folders where the device driver files are located. Each folder structure, starting with the folder specified in this parameter, is searched for device drivers, including all subfolders and the contents of .cab files in the folder structure.

> **Note**
>
> If this parameter is not provided, then the Windows PowerShell working directory must default to the folder where the device driver files are located. This parameter must be provided if the *Path* parameter is not provided.

Expand table

Parameter	Value
Required?	True
Position?	1 and **Named**

Parameter	Value
Default value	-
Accept pipeline input?	False
Accept wildcard characters?	False

-ImportDuplicates [<SwitchParameter>]

This parameter specifies whether this cmdlet should import duplicate device drivers. By default, duplicate device drivers are not imported. Duplicate device drivers are detected by calculating a hash values for all the files in a device driver folder. If the calculated hash value matches another device driver, the device driver to be imported is considered a duplicate.

If a duplicate driver is detected and this parameter is not provided, the device driver will be added and linked to the original, existing device driver.

If this parameter is:

- Specified, then the duplicate device drivers are imported
- Not specified, then the device drivers will be added and linked to the original, existing device drivers

Expand table

Parameter	Value
Required?	False
Position?	Named
Default value	-
Accept pipeline input?	True (ByValue)
Accept wildcard characters?	False

<CommonParameters>

This cmdlet supports the following common parameters: *Verbose, Debug, ErrorAction, ErrorVariable, OutBuffer, OutVariable, WarningAction,* and *WarningVariable*. For more

information, see the topic, "about_CommonParameters," which you can access by typing the following command, and then pressing ENTER:

```
PowerShell

Get-Help about_CommonParameters
```

Outputs

This cmdlet outputs one or more **PSObject** type objects (one for each device driver imported).

Example 1

```
PowerShell

Import-MDTDriver -Path "DS001:\Out-of-Box Drivers" SourcePath "\\WDG-MDT-01\Source$\Drivers"
```

Description

This example imports all device drivers in the folders structure with the root of the folder structure at \\WDG-MDT-01\Source$\Drivers. The device drivers are stored in the Out-of-Box Drivers folder in the deployment share that is mapped to the DS001: MDTProvder Windows PowerShell drive. If any duplicate device drivers are detected, the device drivers will be added and linked to the original, existing device drivers in the deployment share.

Example 2

```
PowerShell

$DriverSourcePath="\\WDG-MDT-01\Source$\VendorADrivers", "\\WDG-MDT-01\Source$\VendorBDrivers"
Import-MDTDriver -Path "DS001:\Out-of-Box Drivers" SourcePath $DriverSourcePath ImportDuplicates
```

Description

This example imports all device drivers in the folders structure specified in the string array $DriverSourcePath. The device drivers are stored in the Out-of-Box Drivers folder in the deployment share that is mapped to the DS001: MDTProvder Windows PowerShell drive. If any duplicate device drivers are detected, the duplicate device drivers are imported.

Import-MDTOperatingSystem

This section describes the **Import-MDTOperatingSystem** Windows PowerShell cmdlet. Run this cmdlet from a Windows PowerShell console that has the MDT PowerShell snap-in loaded. For more information on how to start a Windows PowerShell console that has the MDT PowerShell snap-in loaded, see "Loading the MDT Windows PowerShell Snap-In".

Syntax

PowerShell

```
Import-MDTOperatingSystem [-Path <String>] -SourcePath <String> [-
DestinationFolder <String>] [-Move] [<CommonParameters>]
```

-or-

PowerShell

```
Import-MDTOperatingSystem [-Path <String>] [DestinationFolder <String>] -
SourceFile <String> [SetupPath <String>] [-Move] [<CommonParameters>]
```

-or-

PowerShell

```
Import-MDTOperatingSystem [-Path <String>] -WDSServer <String>
[<CommonParameters>]
```

Description

This cmdlet imports an operating system into a deployment share. The following operating system types can be imported using this cmdlet:

- Operating systems from the original source files, using the *SourcePath* parameters. The first syntax example illustrates the use of this cmdlet for this type of operating

system import.

- Custom operating systems image files, such as capture images from reference computers, using the *SourceFile* parameter. The second syntax example illustrates the use of this cmdlet for this type of operating system import.
- Operating system images that are present in Windows Deployment Services using the *WDSServer* parameter. The last syntax example illustrates the use of this cmdlet for this type of operating system import.

Parameters

This subsection provides information about the various parameters that can be used with the **Import-MDTOperatingSystem** cmdlet.

-Path <String>

This parameter specifies the fully qualified path to an existing folder within the deployment share where the operating system being imported will be placed. If the *DestinationFolder* parameter is used, then the folder specified in the *DestinationFolder* parameter is created beneath the folder specified in this parameter. This parameter is used in all syntax usages for this cmdlet.

> ⓘ **Note**
>
> If this parameter is not provided, then the Windows PowerShell working directory must default to the desired location within the deployment share.

Expand table

Parameter	Value
Required?	**False**
Position?	**Named**
Default value	-
Accept pipeline input?	**False**
Accept wildcard characters?	**False**

-SourcePath <String>

This parameter specifies the fully qualified path to the operating system source files for the operating system that will be imported into the deployment share. This parameter is only valid for use in the first syntax example.

Expand table

Parameter	Value
Required?	True
Position?	Named
Default value	-
Accept pipeline input?	False
Accept wildcard characters?	False

-DestinationFolder <String>

This parameter specifies the folder in the deployment share where the operating system source files are to be imported. This folder is created beneath the folder specified in the *Path* parameter. This parameter is only valid for use in the first and second syntax examples.

Expand table

Parameter	Value
Required?	True
Position?	Named
Default value	-
Accept pipeline input?	False
Accept wildcard characters?	False

-Move [<SwitchParameter>]

This parameter specifies if the operating system source files should be moved (instead of copied) from the folder where the operating system source files are located, which is specified in the *DestinationFolder* parameter.

If this parameter is:

- Specified, then the files are moved and the files in the folder specified in the *DestinationFolder* parameter are deleted
- Not specified, then the files are copied and the files in the folder specified in the *DestinationFolder* parameter are retained

  This parameter is only valid for use in the first and second syntax examples.

Expand table

Parameter	Value
Required?	False
Position?	Named
Default value	-
Accept pipeline input?	False
Accept wildcard characters?	False

-SourceFile <String>

This parameter specifies the fully qualified path to the operating system source .wim file for the operating system that will be imported into the deployment share. This parameter is only valid for use in the second syntax example.

Expand table

Parameter	Value
Required?	True
Position?	Named
Default value	-
Accept pipeline input?	False
Accept wildcard characters?	False

-SetupPath <String>

This parameter specifies the fully qualified path to the operating system setup files that need to be imported along with the .wim file specified in the *SourceFile* parameter. This

parameter is only valid for use in the second syntax example.

Expand table

Parameter	Value
Required?	True
Position?	Named
Default value	-
Accept pipeline input?	False
Accept wildcard characters?	False

-WDSServer <String>

This parameter specifies the name of the Windows Deployment Services server on which the operating system image files to be imported are located. All operating image files on the Windows Deployment Services server will be imported into the deployment share. The actual operating system image files are not copied to the deployment share. Instead, the deployment share contains a link to each operating system file on the Windows Deployment Services server.

This parameter is only valid for use in the last syntax example.

Expand table

Parameter	Value
Required?	False
Position?	Named
Default value	-
Accept pipeline input?	False
Accept wildcard characters?	False

<CommonParameters>

This cmdlet supports the following common parameters: *Verbose*, *Debug*, *ErrorAction*, *ErrorVariable*, *OutBuffer*, *OutVariable*, *WarningAction*, and *WarningVariable*. For more

information, see the topic, "about_CommonParameters," which you can access by typing the following command, and then pressing ENTER:

```
PowerShell

Get-Help about_CommonParameters
```

Outputs

This cmdlet outputs one or more **PSObject** type objects (one for each operating system that was imported).

Example 1

```
PowerShell

Import-MDTOperatingSystem -Path "DS001:\Operating Systems" SourcePath
"\\WDGMDT01\Source$\Windows8" DestinationFolder "Windows8x64"
```

Description

This example imports an operating system from the network shared folder at \\WDG-MDT-01\Source$\Windows8 and copies the source files to DS001:\Operating Systems\Windows8x64 within the deployment share. The source files are retained.

Example 2

```
PowerShell

Import-MDTOperatingSystem -Path "DS001:\Operating Systems" SourcePath
"\\WDGMDT01\Source$\Windows8" DestinationFolder "Windows8x64" -Move
```

Description

This example imports an operating system from the network shared folder at \\WDG-MDT-01\Source$\Windows8 and copies the source files to DS001:\Operating Systems\Windows8x64 within the deployment share. The source files are removed from the network shared folder at \\WDG-MDT-01\Source$\Windows8.

Example 3

PowerShell

```
Import-MDTOperatingSystem -Path "DS001:\Operating Systems" DestinationFolder
"Windows8x64-Reference" -SourceFile "\\WDGMDT01\Capture$\WDG-REF-
01_Capture.wim"
```

Description

This example imports an operating system captured, custom image file (.wim file) from \\WDG-MDT-01\ Capture$\WDG-REF-01_Capture.wim and copies the image file to DS001:\Operating Systems\Windows8x64-Reference within the deployment share. The source .wim file is retained on the network shared folder.

Example 4

PowerShell

```
Import-MDTOperatingSystem -Path "DS001:\Operating Systems" WDSServer "WDG-
WDS-01"
```

Description

This example imports all the operating system images from the Windows Deployment Services server named WDG-WDS-01 and creates a link to each operating system image in DS001:\Operating Systems within the deployment share. The source operating system image files on the Windows Deployment Services server are retained on the Windows Deployment Services server.

Import-MDTPackage

This section describes the **Import-MDTPackage** Windows PowerShell cmdlet. Run this cmdlet from a Windows PowerShell console that has the MDT PowerShell snap-in loaded. For more information on how to start a Windows PowerShell console that has the MDT PowerShell snap-in loaded, see "Loading the MDT Windows PowerShell Snap-In".

Syntax

PowerShell

```
Import-MDTPackage [-Path <String>] [[-SourcePath] <String[]>]
[<CommonParameters>]
```

Description

This cmdlet imports one or more operating system packages into a deployment share. The types of operating system packages that can be imported include security updates, language packs, or new components. Service packs should not be imported as operating system packages as they cannot be installed offline.

Parameters

This subsection provides information about the various parameters that can be used with the **Import-MDTPackage** cmdlet.

-Path <String>

This parameter specifies the fully qualified path to an existing folder within the deployment share where the operating system packages being imported will be placed.

> **Note**
>
> If this parameter is not provided, then the Windows PowerShell working directory must default to the desired location within the deployment share.

Expand table

Parameter	Value
Required?	False
Position?	Named
Default value	-
Accept pipeline input?	False
Accept wildcard characters?	False

-SourcePath <String>

This parameter specifies the fully qualified path to a folder structure to be scanned for operating system packages to import. The specified folder structure will be scanned for .cab and .msu files. For .msu files, the .cab files inside the .msu files are automatically extracted.

Expand table

Parameter	Value
Required?	**True**
Position?	**1** and **Named**
Default value	-
Accept pipeline input?	**False**
Accept wildcard characters?	**False**

<CommonParameters>

This cmdlet supports the following common parameters: *Verbose, Debug, ErrorAction, ErrorVariable, OutBuffer, OutVariable, WarningAction,* and *WarningVariable*. For more information, see the topic, "about_CommonParameters," which you can access by typing the following command, and then pressing ENTER:

PowerShell

```
Get-Help about_CommonParameters
```

Outputs

This cmdlet outputs a **PSObject** type object that references the package just imported.

Example 1

PowerShell

```
Import-MDTOperatingSystem -Path "DS001:\Packages" SourcePath
"\\WDGMDT01\Source$\OSPackages"
```

Description

This example scans network shared folder at \\WDG-MDT-01\Source$\OSPackages for operating system packages and copies the source files to DS001:\Packages folder within the deployment share. The source files are removed from the network shared folder at \\WDG-MDT-01\Source$\OSPackages.

Import-MDTTaskSequence

This section describes the **Import-MDTTaskSequence** Windows PowerShell cmdlet. Run this cmdlet from a Windows PowerShell console that has the MDT PowerShell snap-in loaded. For more information on how to start a Windows PowerShell console that has the MDT PowerShell snap-in loaded, see "Loading the MDT Windows PowerShell Snap-In".

Syntax

PowerShell

```
Import-MDTTaskSequence [-Path <String>] -Template <String> -Name <String> -
ID <String> [[-Comments] <String>] [[-Version] <String>] [-
OperatingSystemPath <String>] [-OperatingSystem <PSObject>] [-FullName
<String>] [-OrgName <String>] [-HomePage <String>] [-ProductKey <String>] [-
OverrideProductKey <String>] [-AdminPassword <String>] [<CommonParameters>]
```

Description

This cmdlet imports a task sequence into a deployment share. The newly imported task sequence will be based on an existing task sequence template specified in the *Template* property.

Parameters

This subsection provides information about the various parameters that can be used with the **Import-MDTPackage** cmdlet.

-Path <String>

This parameter specifies the fully qualified path to an existing folder within the deployment share where the task sequence being imported will be placed. By default, the path should point to the Control folder and or a subfolder of the Control folder in the deployment share. The value of the *ID* parameter will be used to create a subfolder within the path specified in this parameter.

> **Note**
>
> If this parameter is not provided, then the Windows PowerShell working directory must default to the desired location within the deployment share.

Expand table

Parameter	Value
Required?	False
Position?	Named
Default value	-
Accept pipeline input?	False
Accept wildcard characters?	False

-Template <String>

This parameter specifies the task sequence template to be used for importing the new task sequence. Task sequence templates are .xml files that contain the task sequence steps for a particular type of task sequence. If the task sequence template is located in:

- The *installation_folder*\Templates folder (where *installation_folder* is the folder in which MDT is installed), then only the .xml file name is required.
- Another folder, then the fully qualified path, including the name of the task sequence template .xml, is required.

  For more information on the task sequence templates that are included with MDT for LTI deployments, see the section "Create a New Task Sequence in the Deployment Workbench" in the MDT document, *Using the Microsoft Deployment Toolkit*.

Expand table

Parameter	Value
Required?	True
Position?	1 and Named
Default value	-

Parameter	Value
Accept pipeline input?	False
Accept wildcard characters?	False

-Name <String>

This parameter specifies the name of the task sequence to be imported. The value of this parameter must be unique within the deployment share.

Expand table

Parameter	Value
Required?	True
Position?	2 and **Named**
Default value	-
Accept pipeline input?	False
Accept wildcard characters?	False

-ID <String>

This parameter specifies the identifier of the task sequence to be imported. The value of this parameter must be unique within the deployment share. The value assigned to this parameter should be in uppercase and not have any spaces or special characters. This value is used to create a subfolder in the folder specified in the *Path* parameter, which should be under the Control folder in the deployment share.

Expand table

Parameter	Value
Required?	True
Position?	3 and **Named**
Default value	-
Accept pipeline input?	False
Accept wildcard characters?	False

-Comments <String>

This parameter specifies the text that provides additional, descriptive information about the task sequence to be imported. This descriptive information is visible in the Deployment Workbench.

Expand table

Parameter	Value
Required?	False
Position?	4 and Named
Default value	-
Accept pipeline input?	False
Accept wildcard characters?	False

-Version <String>

This parameter specifies the version number of the task sequence to be imported. The value of this parameter is informational only and is not used by MDT for version-related processing.

Expand table

Parameter	Value
Required?	False
Position?	4 and Named
Default value	-
Accept pipeline input?	False
Accept wildcard characters?	False

-OperatingSystemPath <String>

This parameter specifies the fully qualified Windows PowerShell path to the folder in the deployment share that contains the operating system to be used with this task sequence, such as DS001:\Operating Systems\Windows 8. The operating system must already exist in the deployment share where the task sequence is being imported.

> **Note**
>
> If you do not provide this parameter and the task sequence needs to reference an operating system, then you must provide the *OperatingSystem* parameter.

Expand table

Parameter	Value
Required?	False
Position?	Named
Default value	-
Accept pipeline input?	False
Accept wildcard characters?	False

-OperatingSystem <PSObject>

This parameter specifies the operating system object to be used with this task sequence. The operating system must already exist in the deployment share where the task sequence is being imported.

You can retrieve the Windows PowerShell object for an operating system using the **Get-Item** cmdlet, such as the following example:

```PowerShell
$OS=Get-Item "DS001:\Operating Systems\Windows 8"
```

For more information on the **Get-Item** cmdlet, see Using the Get-Item Cmdlet.

> **Note**
>
> If you do not provide this parameter and the task sequence needs to reference an operating system, then you must provide the *OperatingSystemPath* parameter.

Expand table

Parameter	Value
Required?	False
Position?	Named
Default value	-
Accept pipeline input?	False
Accept wildcard characters?	False

-FullName <String>

This parameter specifies the name of the registered owner of the operating system to be used with this task sequence. This name is saved in the **RegisteredOwner** registry key at **HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion**. The value of this parameter is injected into the Unattend.xml file to be associated with this task sequences.

Expand table

Parameter	Value
Required?	False
Position?	Named
Default value	-
Accept pipeline input?	False
Accept wildcard characters?	False

-OrgName <String>

This parameter specifies the name of the organization for the registered owner of the operating system to be used with this task sequence. This name is saved in the **RegisteredOrganization** registry key at **HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion**. The value of this parameter is injected into the Unattend.xml file to be associated with this task sequences.

Expand table

Parameter	Value
Required?	False
Position?	Named
Default value	-
Accept pipeline input?	False
Accept wildcard characters?	False

-HomePage <String>

This parameter specifies the URL to be used as the home page in Internet Explorer. The value of this parameter is injected into the Unattend.xml file to be associated with this task sequences.

Expand table

Parameter	Value
Required?	False
Position?	Named
Default value	-
Accept pipeline input?	False
Accept wildcard characters?	False

-ProductKey <String>

This parameter specifies the product key to be used for the operating system to be used with this task sequence. This product key is valid only for retail versions of Windows operating systems. The value of this parameter is injected into the Unattend.xml file to be associated with this task sequences.

> **Note**
>
> If this parameter is not provided, then the product key must be provided when deploying this task sequence in the Deployment Wizard, in the CustomSettings.ini file, or in the MDT DB.

Expand table

Parameter	Value
Required?	False
Position?	Named
Default value	-
Accept pipeline input?	False
Accept wildcard characters?	False

-OverrideProductKey <String>

This parameter specifies the MAK key to be used for the operating system to be used with this task sequence. This product key is valid only for volume license versions of Windows. The value of this parameter is injected into the Unattend.xml file to be associated with this task sequences.

> **Note**
>
> If this parameter is not provided, then the MAK key must be provided when deploying this task sequence in the Deployment Wizard, in the CustomSettings.ini file, or in the MDT DB.

Expand table

Parameter	Value
Required?	False
Position?	Named
Default value	-
Accept pipeline input?	False
Accept wildcard characters?	False

-AdminPassword <String>

This parameter specifies the password to be assigned to the built-in, local Administrator account on the target computer. The value of this parameter is injected into the

Unattend.xml file to be associated with this task sequences.

> **Note**
>
> If this parameter is not provided, then the password to be assigned to the built-in, local Administrator account on the target computer must be provided when deploying this task sequence in the Deployment Wizard, in the CustomSettings.ini file, or in the MDT DB.

Expand table

Parameter	Value
Required?	False
Position?	Named
Default value	-
Accept pipeline input?	False
Accept wildcard characters?	False

<CommonParameters>

This cmdlet supports the following common parameters: *Verbose, Debug, ErrorAction, ErrorVariable, OutBuffer, OutVariable, WarningAction,* and *WarningVariable*. For more information, see the topic, "about_CommonParameters," which you can access by typing the following command, and then pressing ENTER:

```
PowerShell

Get-Help about_CommonParameters
```

Outputs

This cmdlet outputs a **PSObject** type object that references the task sequence just imported.

Example 1

```
PowerShell
```

```
Import-MDTTaskSequence -Path "DS001:\Control" -Template "Client.xml" -Name
"Deploy Windows 8 to Reference Computer" -ID "WIN8REFERENCE" -Comments "Task
sequence for deploying Windows 8 to the reference computer (WDG-REF-01)" -
Version "1.00" -OperatingSystemPath "DS001:\Operating Systems\Windows 8_x64"
-FullName "Woodgrove Bank Employee" -OrgName "Woodgrove Bank" HomePage
"https://www.woodgrovebank.com"  OverrideProductKey
"123451234512345123451234512345" AdministratorPassword "P@ssw0rd"
```

Description

This example imports a task sequence named *Deploy Windows 8 to Reference Computer* and creates the task sequence in the DS001:\Control\WIN8REFERENCE folder in the deployment share. The comment, "Task sequence for deploying Windows 8 to the reference computer (WDG-REF-01)," is assigned to the task sequence. The version number of the task sequence is set to **1.00**.

The operating system associated with the task sequence is located at DS001:\Operating Systems\Windows 8_x64 in the deployment share. The registered owner of the operating system will be set to **Woodgrove Bank Employee**. The registered organization of the operating system will be set to **Woodgrove Bank**. The Internet Explorer home page will default to `https://www.woodgrovebank.com`. The password for the local, built-in Administrator account will be set to a value of `P@ssw0rd`. The product key for the operating system will be set to **123451234512345123451234512345**.

Example 2

PowerShell

```
$OSObject=Get-Item "DS001:\Operating Systems\Windows 8_x64"
Import-MDTTaskSequence -Path "DS001:\Control" -Template "Client.xml" -Name
"Deploy Windows 8 to Reference Computer" -ID "WIN8REFERENCE" -Comments "Task
sequence for deploying Windows 8 to the reference computer (WDG-REF-01)" -
Version "1.00"-OperatingSystem $OSObject -FullName "Woodgrove Bank Employee"
-OrgName "Woodgrove Bank" HomePage "https://www.woodgrovebank.com"
AdministratorPassword "P@ssw0rd"
```

Description

This example imports a task sequence named *Deploy Windows 8 to Reference Computer* and creates the task sequence in the DS001:\Control\WIN8REFERENCE folder in the deployment share. The comment, "Task sequence for deploying Windows 8 to the

reference computer (WDG-REF-01)," is assigned to the task sequence. The version number of the task sequence is set to **1.00**.

The operating system associated with the task sequence is located at DS001:\Operating Systems\Windows 8_x64 in the deployment share, which is passed to the cmdlet using the *$OSObject* variable. The *$OSObject* variable is set to an existing operating system object using the **Get-Item** cmdlet.

The registered owner of the operating system will be set to **Woodgrove Bank Employee**. The registered organization of the operating system will be set to **Woodgrove Bank**. The Internet Explorer home page will default to `https://www.woodgrovebank.com`. The password for the local, built-in Administrator account will be set to a value of `P@ssw0rd`. The product key for the operating system will need to be provided when deploying this task sequence in the Deployment Wizard, in the CustomSettings.ini file, or in the MDT DB.

New-MDTDatabase

This section describes the **New-MDTDatabase** Windows PowerShell cmdlet. Run this cmdlet from a Windows PowerShell console that has the MDT PowerShell snap-in loaded. For more information on how to start a Windows PowerShell console that has the MDT PowerShell snap-in loaded, see "Loading the MDT Windows PowerShell Snap-In".

Syntax

PowerShell

```
New-MDTDatabase [-Path <String>] [-Force] -SQLServer <String> [-Instance
<String>] [-Port <String>] [-Netlib <String>] -Database <String> [-SQLShare
<String>] [<CommonParameters>]
```

Description

This cmdlet creates a new MDT DB database that is associated with a deployment share. Each deployment share can be associated with only one MDT DB database.

Parameters

This subsection provides information about the various parameters that can be used with the **New-MDTDatabase** cmdlet.

-Path <String>

This parameter specifies the fully qualified Windows PowerShell path to the deployment share to which the new MDT DB database will be associated placed.

> **Note**
>
> If this parameter is not provided, then the Windows PowerShell working directory must default to the desired location within the deployment share.

Expand table

Parameter	Value
Required?	False
Position?	Named
Default value	-
Accept pipeline input?	False
Accept wildcard characters?	False

-Force [<SwitchParameter>]

This parameter specifies that tables within the MDT DB should be recreated if the database specified in the *Database* parameter already exist. If this parameter is:

- Provided, then the tables within an existing MDT DB will be re-created
- Omitted, then the tables within an existing MDT DB will not be re-created

Expand table

Parameter	Value
Required?	False
Position?	Named
Default value	-
Accept pipeline input?	True (ByValue)
Accept wildcard characters?	False

-SQLServer <String>

This parameter specifies the name of the computer running SQL Server where the new MDT DB database will be created.

Expand table

Parameter	Value
Required?	True
Position?	Named
Default value	-
Accept pipeline input?	False
Accept wildcard characters?	False

-Instance <String>

This parameter specifies the SQL Server instance in which the new MDT DB database will be created. If this parameter is omitted, the MDT DB database is created in the default SQL Server instance.

> **Note**
>
> The SQL Server Browser service must be running on the computer running SQL Server for the cmdlet to locate the instance specified in this parameter.

Expand table

Parameter	Value
Required?	False
Position?	Named
Default value	-
Accept pipeline input?	False
Accept wildcard characters?	False

-Port <String>

This parameter specifies the TCP port to be used in communication with the SQL Server instance specified in the *SQLServer* parameter. The default port that SQL Server uses is 1433. Specify this parameter when SQL Server is configured to use a port other than the default value. The value of this parameter must match the port configured for SQL Server.

Expand table

Parameter	Value
Required?	False
Position?	Named
Default value	-
Accept pipeline input?	False
Accept wildcard characters?	False

-Netlib <String>

This parameter specifies the SQL Server network library used in communication with the SQL Server instance specified in the *SQLServer* parameter. The parameter can be set to one of the following values:

- **DBNMPNTW**, which is used to specify named pipes communication
- **DBSMSOCN**, which is used to specify TCP/IP sockets communication

  If this parameter is not provided, the named pipes SQL Server network library (DBNMPNTW) is used.

Expand table

Parameter	Value
Required?	False
Position?	Named
Default value	-
Accept pipeline input?	False
Accept wildcard characters?	False

-Database <String>

This parameter specifies the name of the database to be created in the SQL Server instance specified in the *Instance* parameter on the SQL Server specified in the *SQLServer* parameter. The default location and naming convention will be used for the database and log files when creating the database.

If the database specified in this parameter already exists, the database will not be recreated. The tables within the database can be recreated based on the *Force* parameter.

Expand table

Parameter	Value
Required?	True
Position?	Named
Default value	-
Accept pipeline input?	False
Accept wildcard characters?	False

-SQLShare <String>

This parameter specifies the name of a network shared folder on the computer where SQL Server is running. This connection is used to establish Windows Integrated Security connections using the Named Pipes protocol.

> **Note**
>
> If this parameter is not included, then a secured IPC$ connection is not established. As a result, named pipes communication with SQL Server may fail.

Expand table

Parameter	Value
Required?	False
Position?	Named
Default value	-

Parameter	Value
Accept pipeline input?	False
Accept wildcard characters?	False

<CommonParameters>

This cmdlet supports the following common parameters: *Verbose, Debug, ErrorAction, ErrorVariable, OutBuffer, OutVariable, WarningAction,* and *WarningVariable*. For more information, see the topic, "about_CommonParameters," which you can access by typing the following command, and then pressing ENTER:

```
PowerShell

Get-Help about_CommonParameters
```

Outputs

This cmdlet outputs a **PSObject** type object for the new MDT DB that was created.

Example 1

```
PowerShell

New-MDTDatabase -Path "DS001:" -SQLServer "WDGSQL01" Database "MDTDB" -
SQLShare "\\WDGSQL01\MDTShare$"
```

Description

This example creates an MDT DB named *MDTDB* in the default SQL Server instance on a computer named *WDG-SQL-01*. If the database already exists, the tables in the existing database will not be recreated. The connection will be made using the default SQL Server TCP port and the Named Pipes protocol.

Example 2

```
PowerShell

New-MDTDatabase -Path "DS001:" -Force -SQLServer "WDGSQL01" -Instance
```

```
"MDTInstance" Database "MDTDB" -SQLShare "\\WDGSQL01\MDTShare$"
```

Description

This example creates an MDT DB named *MDTDB* in the SQL Server instance named *MDTInstance* on a computer named *WDG-SQL-01*. If the database already exists, the tables in the existing database will be recreated. The connection will be made using the default SQL Server TCP port and the Named Pipes protocol.

Remove-MDTMonitorData

This section describes the **Get-MDTPersistentDrive** Windows PowerShell cmdlet. Run this cmdlet from a Windows PowerShell console that has the MDT PowerShell snap-in loaded. For more information on how to start a Windows PowerShell console that has the MDT PowerShell snap-in loaded, see "Loading the MDT Windows PowerShell Snap-In".

Syntax

PowerShell

```
Remove-MDTMonitorData [-Path <String>] [-ID <Int32>] [<CommonParameters>]
```

-or-

PowerShell

```
Remove-MDTMonitorData [-Path <String>] [-ComputerObject <PSObject>]
[<CommonParameters>]
```

Description

This cmdlet removes collected monitoring data from the existing collected monitoring data in a deployment share. You can identify the monitoring data to remove by specifying the:

- Identifier (ID) of the monitoring item for a specific deployment share. The monitoring item IDs are automatically generated and assigned to the item when the item is created for the deployment share. The first syntax example illustrates this usage.

- Computer object for the monitoring item in the deployment share. The computer object can be obtained using the **Get-MDTMonitorData** cmdlet. The last syntax example illustrates this usage.

> ⓘ Note
>
> Once the monitoring data has been removed, there is no method for recovering the information.

Parameters

This subsection provides information about the various parameters that can be used with the **Get- MDTMonitorData** cmdlet.

-Path <String>

This parameter specifies the **MDTProvider** Windows PowerShell drive for the desired deployment share.

> ⓘ Note
>
> If this parameter is not provided, then the Windows PowerShell working directory must default to a location within the desired MDTProvider Windows PowerShell drive.

Expand table

Parameter	Value
Required?	False
Position?	Named
Default value	-
Accept pipeline input?	False
Accept wildcard characters?	False

-ID <Nullable>

This parameter specifies the monitoring data item to be removed using the identifier of the monitoring data item. If this parameter is not specified, then the *ComputerObject* parameter must be specified to identify a particular monitoring data item.

Expand table

Parameter	Value
Required?	False
Position?	Named
Default value	-
Accept pipeline input?	True (ByValue)
Accept wildcard characters?	False

-ComputerObject <PSObject>

This parameter specifies the monitoring data item to be removed using a computer object. If this parameter is not specified, then the *ID* parameter must be specified to identify a particular monitoring data item.

Expand table

Parameter	Value
Required?	False
Position?	Named
Default value	-
Accept pipeline input?	True (ByValue)
Accept wildcard characters?	False

<CommonParameters>

This cmdlet supports the following common parameters: *Verbose, Debug, ErrorAction, ErrorVariable, OutBuffer, OutVariable, WarningAction,* and *WarningVariable*. For more information, see the topic, "about_CommonParameters," which you can access by typing the following command, and then pressing ENTER:

```
PowerShell
```

```
Get-Help about_CommonParameters
```

Outputs

This cmdlet may output a **String** type object if the *Verbose* common parameter is included; otherwise, no output is generated.

Example 1

PowerShell

```
Remove-MDTMonitorData -Path "DS001:" -ID 3
```

Description

This example removes the monitoring data item with an ID that has a value of **3** from the deployment share at the Windows PowerShell path DS001:.

Example 2

PowerShell

```
Remove-MDTMonitorData -ID 3
```

Description

This example removes the monitoring data item with an ID that has a value of **3** from the deployment share at the default Windows PowerShell path.

Example 3

PowerShell

```
$MonitorObject=Get-MDTMonitorData | Where-Object {$_.Name eq 'WDG-REF-01'}
Remove-MDTMonitorData -ComputerObject $MonitorObject
```

Description

This example removes any monitoring data item where the name of the computer is WDG-REF-01. The object is found using the **Get-MDTMonitorData** cmdlet and the **Where-Object** cmdlet. For more information on the **Where-Object** cmdlet, see Using the Where-Object Cmdlet.

Remove-MDTPersistentDrive

This section describes the **Remove-MDTPersistentDriveWindows** Windows PowerShell cmdlet. Run this cmdlet from a Windows PowerShell console that has the MDT PowerShell snap-in loaded. For more information on how to start a Windows PowerShell console that has the MDT PowerShell snap-in loaded, see "Loading the MDT Windows PowerShell Snap-In".

Syntax

PowerShell

```
Remove-MDTPersistentDrive [-Name] <String> [[-InputObject] <PSObject>]
[<CommonParameters>]
```

Description

This cmdlet removes an existing Windows PowerShell drive created using the **MDTProvider** from the list of drives that are persisted in the Deployment Workbench or in a Windows PowerShell session using the Restore-MDTPersistentDrive cmdlet. This cmdlet is called when a deployment share is closed in (removed from) the Deployment Workbench.

> ⓘ **Note**
>
> The list of persisted **MDTProvider** drives is maintained on a per-user based in the user profile.

The list of persisted **MDTProvider** drives can be displayed using the **Get-MDTPersistentDrive** cmdlet. An MDTProvider drive can be added to the list of persisted drives using the **Add-MDTPersistentDrive** cmdlet.

Parameters

This subsection provides information about the various parameters that can be used with the **Add-MDTPersistentDriveWindows** cmdlet.

-Name <String>

Specifies the name of a Windows PowerShell drive created using the MDT provider and corresponds to an existing deployment share. The name was created using the New-PSDrive cmdlet and specifying the **MDTProvider** in the *PSProvider* parameter.

For more information on how to create a new Windows PowerShell drive using the **MDTProvider** and how to create a deployment share using Windows PowerShell, see the section "Creating a Deployment Share Using Windows PowerShell" in the MDT document, *Microsoft Deployment Toolkit Samples Guide*.

Expand table

Parameter	Value
Required?	True
Position?	1 and Named
Default value	None
Accept pipeline input?	True (ByValue)
Accept wildcard characters?	False

-InputObject <PSObject>

This parameter specifies a Windows PowerShell drive object that was created earlier in the process. Enter a **PSObject** object, such as one generated by the New-PSDrive cmdlet.

Expand table

Parameter	Value
Required?	False
Position?	2 and Named
Default value	-
Accept pipeline input?	True (ByValue)

Parameter	Value
Accept wildcard characters?	False

<CommonParameters>

This cmdlet supports the following common parameters: *Verbose, Debug, ErrorAction, ErrorVariable, OutBuffer, OutVariable, WarningAction,* and *WarningVariable*. For more information, see the topic, "about_CommonParameters," which you can access by typing the following command, and then pressing ENTER:

```PowerShell
Get-Help about_CommonParameters
```

Outputs

This cmdlet provides no outputs.

Example 1

```PowerShell
Remove-MDTPersistentDrive -Name "DS001:"
```

Description

This example removes the deployment share with the Windows PowerShell drive name of *DS001* from the list of persisted drives.

Example 2

```PowerShell
$MDTPSDrive = Get-PSDrive | Where-Object {$_.Root -eq "C:\DeploymentShare" -
and $_.Provider -like "*MDTProvider"}
Remove-MDTPersistentDrive -InputObject $MDTPSDrive
```

Description

This example removes the deployment share at C:\DeploymentShare$ from the list of persisted drives. The **GetPSDrive** and **Where-Object** cmdlets are used to return the MDT persisted Windows PowerShell drive to the **Remove-MDTPersistentDrive** cmdlet using the *$MDTPSDrive* variable. For more information on the **Where-Object** cmdlet, see Using the Where-Object Cmdlet. For more information on the **Get-PSDrive** cmdlet, see Using the Get-PSDrive Cmdlet.

Restore-MDTPersistentDrive

This section describes the **Restore-MDTPersistentDrive** Windows PowerShell cmdlet. Run this cmdlet from a Windows PowerShell console that has the MDT PowerShell snap-in loaded. For more information on how to start a Windows PowerShell console that has the MDT PowerShell snap-in loaded, see "Loading the MDT Windows PowerShell Snap-In".

Syntax

PowerShell

```
Restore-MDTPersistentDrive [-Force] [<CommonParameters>]
```

Description

This cmdlet restores a persisted MDT Windows PowerShell drive to the list of active Windows PowerShell drive for each deployment share that was added to the list of persisted MDT Windows PowerShell drives. The list of persisted MDT Windows PowerShell drives is managed using the Add-MDTPersistentDrive and Remove-MDTPersistentDrive cmdlets or the Deployment Workbench.

This cmdlet calls the **New-PSDrive** cmdlet to create a Windows PowerShell drive for each drive in the MDT persisted list. Persisted MDT Windows PowerShell drives are similar to persisted network drive mappings.

> ⓘ **Note**
>
> This list of persisted MDT Windows PowerShell drives is maintained on a per-user basis and are stored in the user profile.

Parameters

This subsection provides information about the various parameters that can be used with the **Restore-MDTPersistentDrive** cmdlet.

-Force [<SwitchParameter>]

This parameter specifies that the deployment share should be upgraded when restored (if required). If this parameter is:

- Provided, then the deployment share will be upgraded when restored (if required)
- Omitted, then deployment share will not be upgraded when restored

Expand table

Parameter	Value
Required?	**False**
Position?	**Named**
Default value	-
Accept pipeline input?	**True** (**ByValue**)
Accept wildcard characters?	**False**

<CommonParameters>

This cmdlet supports the following common parameters: *Verbose, Debug, ErrorAction, ErrorVariable, OutBuffer, OutVariable, WarningAction,* and *WarningVariable*. For more information, see the topic, "about_CommonParameters," which you can access by typing the following command, and then pressing ENTER:

```
PowerShell

Get-Help about_CommonParameters
```

Outputs

This cmdlet outputs a **PSObject** type object for each MDT Provider Windows PowerShell drive that is restored.

Example 1

PowerShell

```
Get-MDTPersistentDrive
```

Description

This example restores the list of MDT persisted drives, by creating a Windows PowerShell drive using the **MDTProvider** type. The deployment share will not be upgraded when restored.

Example 2

PowerShell

```
Get-MDTPersistentDrive -Force
```

Description

This example restores the list of MDT persisted drives, by creating a Windows PowerShell drive using the **MDTProvider** type. The deployment share will be upgraded when restored (if required).

Set-MDTMonitorData

This section describes the **Get-MDTPersistentDrive** Windows PowerShell cmdlet. Run this cmdlet from a Windows PowerShell console that has the MDT PowerShell snap-in loaded. For more information on how to start a Windows PowerShell console that has the MDT PowerShell snap-in loaded, see "Loading the MDT Windows PowerShell Snap-In".

Syntax

PowerShell

```
Set-MDTMonitorData [-Path <String>] [-ComputerObject <PSObject>] [-Settings
<Hashtable>] [<CommonParameters>]
```

-or-

PowerShell

```
Set-MDTMonitorData [-Path <String>] [-MacAddress <String>] [Settings
<Hashtable>] [<CommonParameters>]
```

Description

This cmdlet creates a new monitoring data item, or updates an existing monitoring data item, in a deployment share. You can identify the monitoring data to remove by specifying the:

- Computer object for the monitoring item in the deployment share. The computer object can be obtained using the **Get-MDTMonitorData** cmdlet. The first syntax example illustrates this usage.
- MAC address of the primary network adapter of the monitoring item for a specific deployment share. The MAC address is automatically assigned to the monitoring data item when the item is created for the deployment share. The last syntax example illustrates this usage.

> ⓘ **Note**
>
> Once the monitoring data has been removed, there is no method for recovering the information.

Parameters

This subsection provides information about the various parameters that can be used with the **Get- MDTMonitorData** cmdlet.

-Path <String>

This parameter specifies the MDTProvider Windows PowerShell drive for the desired deployment share.

> ⓘ **Note**
>
> If this parameter is not provided, then the Windows PowerShell working directory must default to a location within the desired MDTProvider Windows PowerShell drive.

Expand table

Parameter	Value
Required?	False
Position?	Named
Default value	-
Accept pipeline input?	False
Accept wildcard characters?	False

-ComputerObject <PSObject>

This parameter specifies the monitoring data item to be created or updated using a computer object. If this parameter is not specified, then the *MACAddress* parameter must be specified to identify a particular monitoring data item.

Expand table

Parameter	Value
Required?	False
Position?	Named
Default value	-
Accept pipeline input?	True (ByValue)
Accept wildcard characters?	False

-MACAddress <String>

This parameter specifies the monitoring data item to be created or updated using the MAC address of the primary network adapter of the computer being monitored. The format of the MACAddress is *xx:xx:xx:xx:xx:xx,* where *x* is a hexadecimal character specified in uppercase (as required). If this parameter is not specified, then the *ComputerObject* parameter must be specified to identify a particular monitoring data item.

Expand table

Parameter	Value
Required?	False
Position?	Named
Default value	-
Accept pipeline input?	True (ByValue)
Accept wildcard characters?	False

-Settings <Hashtable>

This parameter specifies the monitoring data settings for the monitoring data item to be created or updated. The format of the hashtable provided with this parameter is `@{"Setting"="Value"; "Setting1"="Value1"; "Setting2"="Value2}`. If this parameter is not specified, then the monitoring data item is created, but no monitoring information is stored.

`"Setting"` can be any property listed in the ZTIGather.xml file. `Value` can be any valid value for the property specified in `"Setting"`.

Expand table

Parameter	Value
Required?	False
Position?	Named
Default value	-
Accept pipeline input?	True (ByValue)
Accept wildcard characters?	False

<CommonParameters>

This cmdlet supports the following common parameters: *Verbose, Debug, ErrorAction, ErrorVariable, OutBuffer, OutVariable, WarningAction,* and *WarningVariable*. For more information, see the topic, "about_CommonParameters," which you can access by typing the following command, and then pressing ENTER:

```
PowerShell
```

```
Get-Help about_CommonParameters
```

Outputs

This cmdlet does not generate any output.

Example 1

PowerShell

```
$MonitorObject=Get-MDTMonitorData | Where-Object {$_.Name eq 'WDG-REF-01'}
Set-MDTMonitorData -ComputerObject $MonitorObject Setting
@{"OSDComputerName"="WDG-MDT-01";"SkipWizard"="YES"}
```

Description

This example removes any monitoring data item where the name of the computer is *WDG-REF-01*. The object is found using the **Get-MDTMonitorData** cmdlet and the **Where-Object** cmdlet. For more information on the **Where-Object** cmdlet, see Using the Where-Object Cmdlet. The OSDComputerName property is recorded as having a value of **WDG-MDT-01**, and the SkipWizard property is recorded as having a value of **YES**.

Example 2

PowerShell

```
Set-MDTMonitorData -MACAddress "00:11:22:33:44:55" MonitorObject Setting
@{"OSDComputerName"="WDG-MDT-01";"SkipWizard"="YES"}
```

Description

This example creates or updates a monitoring data item with a **MACAddress** that has a value of **00:11:22:33:44:55**. The OSDComputerName property is recorded as having a value of **WDG-MDT-01**, and the SkipWizard property is recorded as having a value of **YES**.

Test-MDTDeploymentShare

Although this cmdlet is returned using the **Get-Command** cmdlet as being in the Microsoft.BDD.PSSnapIn snap-in, it is not implemented.

Test-MDTMonitorData

This section describes the **Test-MDTMonitorData** Windows PowerShell cmdlet. Run this cmdlet from a Windows PowerShell console that has the MDT PowerShell snap-in loaded. For more information on how to start a Windows PowerShell console that has the MDT PowerShell snap-in loaded, see "Loading the MDT Windows PowerShell Snap-In".

Syntax

```
PowerShell

Test-MDTMonitorData -ServerName <String> -EventPort <Int32> -DataPort
<Int32> [<CommonParameters>]
```

Description

This cmdlet validates if the MDT monitoring service, which runs on the computer on which MDT is installed, is enabled and running properly. The MDT monitoring service collects monitoring information that can be displayed:

- In the Monitoring node in a deployment share in the Deployment Workbench
- Using the Get-MDTMonitorData cmdlet

  The MDT monitoring service can be disabled using the Disable-MDTMonitorService. Monitoring information can be written to the MDT monitoring service using the Set-MDTMonitorData cmdlet.

> ⓘ **Note**
>
> For this cmdlet to function properly there must be at least one MDT monitoring data item in the deployment share. If no MDT monitoring information has been recorded, the deployment share will fail the test.

For more information on the MDT monitoring service, see the section "Monitoring MDT Deployments" in the MDT document, *Using the Microsoft Deployment Toolkit*.

Parameters

This subsection provides information about the various parameters that can be used with the **Test-MDTMonitorData** cmdlet.

-Server <String>

Specifies the name of the computer on which MDT is installed and the MDT monitoring service is running.

Expand table

Parameter	Value
Required?	True
Position?	Named
Default value	None
Accept pipeline input?	False
Accept wildcard characters?	False

-DataPort <Int32>

This parameter specifies the TCP port used as the data port for the MDT monitoring service.

Expand table

Parameter	Value
Required?	True
Position?	Named
Default value	-
Accept pipeline input?	False
Accept wildcard characters?	False

-EventPort <Int32>

This parameter specifies the TCP port used as the event port for the MDT monitoring service.

Expand table

Parameter	Value
Required?	True
Position?	Named
Default value	-
Accept pipeline input?	False
Accept wildcard characters?	False

<CommonParameters>

This cmdlet supports the following common parameters: *Verbose, Debug, ErrorAction, ErrorVariable, OutBuffer, OutVariable, WarningAction,* and *WarningVariable*. For more information, see the topic, "about_CommonParameters," which you can access by typing the following command, and then pressing ENTER:

```PowerShell
Get-Help about_CommonParameters
```

Outputs

This cmdlet outputs a Boolean value that represents the success (true) or failure (false) of the text.

Example 1

```PowerShell
Test-MDTMonitorData -Server "WDG-MDT-01" -DataPort "9801" EventPort "9800"
```

Description

This example verifies if the MDT monitoring service on WDG-MDT-01 is installed and running. The cmdlet will verify using a data port of 9801 and an event port of 9800.

Update-MDTDatabaseSchema

This section describes the **Update-MDTDatabaseSchema** Windows PowerShell cmdlet. Run this cmdlet from a Windows PowerShell console that has the MDT PowerShell snap-in loaded. For more information on how to start a Windows PowerShell console that has the MDT PowerShell snap-in loaded, see "Loading the MDT Windows PowerShell Snap-In".

Syntax

PowerShell

```
Update-MDTDatabaseSchema -SQLServer <String> [-Instance <String>] [-Port
<String>] [-Netlib <String>] -Database <String> [-SQLShare <String>]
[<CommonParameters>]
```

Description

This cmdlet updates an existing MDT DB database to the latest version of the MDT DB database schema. Each deployment share can be associated with only one MDT DB database.

This cmdlet is automatically called when a deployment share is being upgraded, such as when running the Restore-MDTPersistentDrive cmdlet with the *Force* parameter and the Update-MDTDeploymentShare cmdlet.

Parameters

This subsection provides information about the various parameters that can be used with the **Upgrade-MDTDatabaseSchema** cmdlet.

-SQLServer <String>

This parameter specifies the name of the computer running SQL Server where the MDT DB database will be upgraded.

Expand table

Parameter	Value
Required?	True
Position?	Named
Default value	-
Accept pipeline input?	False
Accept wildcard characters?	False

-Instance <String>

This parameter specifies the SQL Server instance on which the MDT DB database to be upgraded exists. If this parameter is omitted, then the MDT DB database is assumed to be in the default SQL Server instance.

> **Note**
>
> The SQL Server Browser service must be running on the computer running SQL Server for the cmdlet to locate the instance specified in this parameter.

Expand table

Parameter	Value
Required?	False
Position?	Named
Default value	-
Accept pipeline input?	False
Accept wildcard characters?	False

-Port <String>

This parameter specifies the TCP port to be used in communication with the SQL Server instance specified in the *SQLServer* parameter. The default port that SQL Server uses is 1433. Specify this parameter when SQL Server is configured to use a port other than the default value. The value of this parameter must match the port configured for SQL Server.

Expand table

Parameter	Value
Required?	False
Position?	Named
Default value	-
Accept pipeline input?	False
Accept wildcard characters?	False

-Netlib <String>

This parameter specifies the SQL Server network library that is used in communication with the SQL Server instance specified in the *SQLServer* parameter. The parameter can be set to one of the following values:

- **DBNMPNTW**, which is used to specify named pipes communication
- **DBSMSOCN**, which is used to specify TCP/IP sockets communication

  If this parameter is not provided, the named pipes SQL Server network library (**DBNMPNTW**) is used.

> **Note**
>
> The Deployment Workbench does not provide the option for configuring the SQL Server network library. The Deployment Workbench always uses named pipes communication. However, the SQL Server network library can be configured in the CustomSettings.ini file.

Expand table

Parameter	Value
Required?	False
Position?	Named
Default value	-
Accept pipeline input?	False

Parameter	Value
Accept wildcard characters?	False

-Database <String>

This parameter specifies the name of the database to be upgraded in the SQL Server instance specified in the *Instance* parameter on the SQL Server instance specified in the *SQLServer* parameter.

Expand table

Parameter	Value
Required?	True
Position?	Named
Default value	-
Accept pipeline input?	False
Accept wildcard characters?	False

<CommonParameters>

This cmdlet supports the following common parameters: *Verbose, Debug, ErrorAction, ErrorVariable, OutBuffer, OutVariable, WarningAction,* and *WarningVariable*. For more information, see the topic, "about_CommonParameters," which you can access by typing the following command, and then pressing ENTER:

```
PowerShell

Get-Help about_CommonParameters
```

Outputs

This cmdlet outputs a **PSObject** type object for the MDT database that was upgraded. This cmdlet also outputs a **String** type data if the *Verbose* common parameter is included.

Example 1

PowerShell

```
Update-MDTDatabaseSchema -SQLServer "WDGSQL01" Database "MDTDB"
```

Description

This example updates the schema for an MDT database named *MDTDB* in the default SQL Server instance on a computer named *WDG-SQL-01*. The connection will be made to the SQL Server instance using the default TCP port and the Named Pipes protocol.

Example 2

PowerShell

```
Update-MDTDatabaseSchema -SQLServer "WDGSQL01" -Instance "MDTInstance" -Port
"6333" Database "MDTDB"
```

Description

This example updates the schema for an MDT database named *MDTDB* in the SQL Server instance named *MDTInstance* on a computer named *WDG-SQL-01*. The connection will be made to the SQL Server using TCP port 6333 and the Named Pipes protocol.

Update-MDTDeploymentShare

This section describes the **Update-MDTDeploymentShare** Windows PowerShell cmdlet. Run this cmdlet from a Windows PowerShell console that has the MDT PowerShell snap-in loaded. For more information on how to start a Windows PowerShell console that has the MDT PowerShell snap-in loaded, see "Loading the MDT Windows PowerShell Snap-In".

Syntax

PowerShell

```
Update-MDTDeploymentShare [-Path <String>] [-Force] [Compress]
[<CommonParameters>]
```

Description

This cmdlet updates an existing deployment share with the latest files from the Windows ADK. This cmdlet also updates or regenerates the required Windows PE boot images in both WIM and ISO file formats.

Parameters

This subsection provides information about the various parameters that can be used with the **Update-MDTDeploymentShare** cmdlet.

-Path <String>

This parameter specifies the fully qualified path to an existing folder in the deployment share that is being updated.

> **Note**
>
> If this parameter is not provided, then the Windows PowerShell working directory must default to the desired location within the deployment share.

Expand table

Parameter	Value
Required?	False
Position?	Named
Default value	-
Accept pipeline input?	False
Accept wildcard characters?	False

-Force [<SwitchParameter>]

This parameter specifies whether Windows PE boot images (.iso and .wim files) for the deployment share should be completely regenerated. If this parameter is:

- Provided, then the cmdlet creates new versions of the Windows PE boot images. This process takes more time than optimizing the existing Windows PE boot images.

- Omitted, then the cmdlet optimizes the existing Windows PE boot images. This process takes less time than generating new versions of the Windows PE boot images. If this parameter is omitted, the *Compress* parameter can be used to reduce the size of the boot images as a part of the Windows PE boot image optimization process.

Expand table

Parameter	Value
Required?	False
Position?	Named
Default value	-
Accept pipeline input?	True (ByValue)
Accept wildcard characters?	False

-Compress [<SwitchParameter>]

This parameter specifies whether Windows PE boot images (.iso and .wim files) for the deployment share should be compressed when they are optimized (without the *Force* parameter). If this parameter is:

- Provided, then the cmdlet compresses the Windows PE boot images as they are being optimized
- Omitted, then the cmdlet does not compress the Windows PE boot images as they are being optimized

> **Note**
>
> This parameter should only be provided if the *Force* parameter is not provided. If the *Force* parameter is included, new Windows PE boot images are generated and are compressed to the minimal size.

Expand table

Parameter	Value
Required?	False

Parameter	Value
Position?	Named
Default value	-
Accept pipeline input?	True (ByValue)
Accept wildcard characters?	False

<CommonParameters>

This cmdlet supports the following common parameters: *Verbose, Debug, ErrorAction, ErrorVariable, OutBuffer, OutVariable, WarningAction,* and *WarningVariable*. For more information, see the topic, "about_CommonParameters," which you can access by typing the following command, and then pressing ENTER:

```
PowerShell

Get-Help about_CommonParameters
```

Outputs

This cmdlet outputs a **String** type data and produces additional **String** type data if the *Verbose* common parameter is included.

Example 1

```
PowerShell

Update-MDTDepoymentShare
```

Description

This example updates the deployment share at the Windows PowerShell working directory. The Windows PE boot images will be optimized. The Windows PE boot images will not be compressed.

Example 2

```
PowerShell
```

```
Update-MDTDepoymentShare -Path "DS001:"
```

Description

This example updates the deployment share at the MDT Windows PowerShell drive named *DS001:*. The Windows PE boot images will be optimized. The Windows PE boot images will not be compressed.

Example 3

PowerShell

```
Update-MDTDepoymentShare -Path "DS001:" -Compress
```

Description

This example updates the deployment share at the MDT Windows PowerShell drive named *DS001:*. The Windows PE boot images will be optimized. The Windows PE boot images will be compressed.

Example 4

PowerShell

```
Update-MDTDepoymentShare -Path "DS001:" -Force
```

Description

This example updates the deployment share at the MDT Windows PowerShell drive named *DS001:*. New versions of the Windows PE boot images will be generated.

Update-MDTLinkedDS

This section describes the **Update-MDTLinkedDS** Windows PowerShell cmdlet. Run this cmdlet from a Windows PowerShell console that has the MDT PowerShell snap-in loaded. For more information on how to start a Windows PowerShell console that has the MDT PowerShell snap-in loaded, see "Loading the MDT Windows PowerShell Snap-In".

Syntax

```
PowerShell

Update-MDTLinkedDS -Path <String> [<CommonParameters>]
```

Description

This cmdlet replicates content from a deployment share to a linked deployment share using the selection profile used to define the linked deployment share. The replication behavior is determined based on the configuration settings for the linked deployment share.

Parameters

This subsection provides information about the various parameters that can be used with the **Update-MDTLinkedDS** cmdlet.

-Path <String>

This parameter specifies the fully qualified path to the linked deployment share that is being updated.

> **Note**
>
> If this parameter is not provided, then the Windows PowerShell working directory must default to the desired location within the deployment share.

Expand table

Parameter	Value
Required?	True
Position?	Named
Default value	-
Accept pipeline input?	False
Accept wildcard characters?	False

<CommonParameters>

This cmdlet supports the following common parameters: *Verbose, Debug, ErrorAction, ErrorVariable, OutBuffer, OutVariable, WarningAction,* and *WarningVariable*. For more information, see the topic, "about_CommonParameters," which you can access by typing the following command, and then pressing ENTER:

```
PowerShell

Get-Help about_CommonParameters
```

Outputs

This cmdlet outputs a **String** type data and produces additional **String** type data if the *Verbose* common parameter is included.

Example 1

```
PowerShell

Update-MDTLinkedDS -Path "DS001:\Linked Deployment Shares\LINKED001"
```

Description

This example replicates content from the deployment share to the linked deployment share at the Windows PowerShell path DS001:\Linked Deployment Shares\LINKED001 folder.

Update-MDTMedia

This section describes the **Update-MDTMedia** Windows PowerShell cmdlet. Run this cmdlet from a Windows PowerShell console that has the MDT PowerShell snap-in loaded. For more information on how to start a Windows PowerShell console that has the MDT PowerShell snap-in loaded, see "Loading the MDT Windows PowerShell Snap-In".

Syntax

```
PowerShell
```

```
Update-MDTMedia -Path <String> [<CommonParameters>]
```

Description

This cmdlet replicates content from a deployment share to a folder that contains deployment media using the selection profile used to define the deployment media. The replication behavior is determined based on the configuration settings for the deployment media.

Media in LTI allows you to perform LTI deployments solely from local media without connecting to a deployment share. You can store the media on a DVD, USB hard disk, or other portable device. After you create the media, generate bootable WIM images that allow the deployment to be performed from portable media devices locally available on the target computer.

Parameters

This subsection provides information about the various parameters that can be used with the **Update-MDTMedia** cmdlet.

-Path <String>

This parameter specifies the fully qualified path to the folder that contains the deployment media that is being updated.

> **Note**
>
> If this parameter is not provided, then the Windows PowerShell working directory must default to the desired location within the deployment share.

Expand table

Parameter	Value
Required?	True
Position?	Named
Default value	-
Accept pipeline input?	False

Parameter	Value
Accept wildcard characters?	False

<CommonParameters>

This cmdlet supports the following common parameters: *Verbose, Debug, ErrorAction, ErrorVariable, OutBuffer, OutVariable, WarningAction,* and *WarningVariable*. For more information, see the topic, "about_CommonParameters," which you can access by typing the following command, and then pressing ENTER:

```
PowerShell

Get-Help about_CommonParameters
```

Outputs

This cmdlet outputs a **String** type data and produces additional **String** type data if the *Verbose* common parameter is included.

Example 1

```
PowerShell

Update-MDTMedia -Path "DS001:\Media\MEDIA001"
```

Description

This example replicates content from the deployment share to the folder containing the deployment media at the Windows PowerShell path DS001:\Media \MEDIA001 folder.

Related articles

- Task Sequence Steps.
- Properties.
- Scripts.
- Support Files.
- Utilities.
- Tables and Views in the MDT DB.
- Windows 7 Feature Dependency Reference.

- UDI Reference.

Feedback

Was this page helpful? Yes No

Provide product feedback

Tables and Views in the MDT DB

Article • 02/12/2024

In MDT, many property settings can be stored (typically configured in the CustomSettings.ini file) in a database. Configuring the properties in a database helps create a generic CustomSettings.ini file that requires fewer modifications and allows one CustomSettings.ini file to be used in more images (because the file is more generic).

Customize the database in the Database node in the Deployment Workbench. Using the Deployment Workbench, the deployment settings can be configured and saved in tables.

However, queries about the information in the tables are done using views. Views help simplify the queries by joining results from multiple tables. ZTIGather.wsf queries the views to return the result set that the **Parameters** and **ParameterCondition** properties specify.

Tables in the MDT DB

The following table lists the database tables that Deployment Workbench creates and manages.

Expand table

Table	Description
ComputerIdentity	Used to identify a specific computer using any combination of the **AssetTag, UUID, SerialNumber**, and **MACAddress** properties. The table includes a **Description** column to provide a user-friendly method of describing the computer (usually the computer name).
Descriptions	Contains descriptions of all properties configurable via the database.
LocationIdentity	Used to identify geographic locations using the **Location** property. The values for this property are stored in a corresponding column in the table.
LocationIdentity_DefaultGateway	Relates the default gateway values with a location identified in the LocationIdentity table. There is a one-to-many relationship between this table and the LocationIdentity table.
MakeModelIdentity	Used to identify a specific make and model of a computer using the **Make** and **Model** properties. The values for these

Table	Description
	properties are stored in corresponding columns in the table.
PackageMapping	Used to associate the name presented in the Add or Remove Programs Control Panel item with a Configuration Manager package and program to be deployed in place of the application in Add or Remove Programs. For more information on this table, see the section, "Deploying Applications Based on Earlier Application Versions", in the MDT document *Microsoft Deployment Toolkit Samples Guide*.
RoleIdentity	Used to identify the purpose of a computer or the users of a computer using the **Role** property. The values for this property are stored in a corresponding column in the table.
Settings	Identifies the settings that are applied to an individual computer or a group of computers based on the settings in the Computers, Roles, Locations, and Make and Model nodes in the Database node in the Deployment Workbench.
Settings_Administrators	Identifies the user accounts to be added to the local Administrator group on the target computer based on the settings in the Computers, Roles, Locations, and Make and Model nodes in the Database node in the Deployment Workbench.
Settings_Applications	Identifies the applications to be deployed to the target computer based on the settings in the Computers, Roles, Locations, and Make and Model nodes in the Database node in the Deployment Workbench.
Settings_Packages	Identifies the packages to be deployed to the target computer based on the settings in the Computers, Roles, Locations, and Make and Model nodes in the Database node in the Deployment Workbench.
Settings_Roles	Identifies the roles to be associated with the target computer based on the settings in the Computers, Locations, and Make and Model nodes in the Database node in the Deployment Workbench.

Views in the MDT DB

The following table lists and describes the database views that are used when querying configuration information in the MDT DB.

Expand table

View	Description
ComputerAdministrators	Used to find all accounts to be made members of the local Administrators group on the target computer. The view is a join of the ComputerIdentity and Settings_Administrators tables.
ComputerApplications	Used to find all applications to be deployed to the target computer. The view is a join of the ComputerIdentity and Settings_Applications tables.
ComputerPackages	Used to find all packages to be deployed to the target computer. The view is a join of the ComputerIdentity and Settings_Packages tables.
ComputerRoles	Used to find all roles to be associated with the target computer. The view is a join of the ComputerIdentity and Settings_Roles tables.
ComputerSettings	Used to find all property settings to be configured for the target computer. The view is a join of the ComputerIdentity and Settings tables.
LocationAdministrators	Used to find all the accounts to be made a member of the local Administrators group on the target computers within a location. The view is a join of the LocationIdentity, LocationIdentity_DefaultGateway, and Settings_Administrators tables.
LocationApplications	Used to find all the applications to be deployed to the target computers within a location. The view is a join of the LocationIdentity, LocationIdentity_DefaultGateway, and Settings_Applications tables.
LocationPackages	Used to find all the packages to be deployed to the target computers within a location. The view is a join of the LocationIdentity, LocationIdentity_DefaultGateway, and Settings_Packages tables.
LocationRoles	Used to find all the roles to be associated with the target computers within a location. The view is a join of the LocationIdentity, LocationIdentity_DefaultGateway, and Settings_Roles tables.
Locations	Used to find the IP addresses for the default gateways within a location or for all the locations that contain a specified IP address for a default gateway. The view is a join of the LocationIdentity and LocationIdentity_DefaultGateway tables.
LocationSettings	Used to find all the property settings to be configured for the target computers within a location. The view is a join of the LocationIdentity, LocationIdentity_DefaultGateway, and Settings tables.

View	Description
MakeModelAdministrators	Used to find all accounts to be made members of the local Administrators group on the target computers with a given make and model. The view is a join of the MakeModelIdentity and Settings_Administrators tables.
MakeModelApplications	Used to find all applications to be deployed to the target computers with a given make and model. The view is a join of the MakeModelIdentity and Settings_Applications tables.
MakeModelPackages	Used to find all packages to be deployed to the target computers with a given make and model. The view is a join of the MakeModelIdentity and Settings_Applications tables.
MakeModelRoles	Used to find all roles associated with the target computers with a given make and model. The view is a join of the MakeModelIdentity and Settings_Roles tables.
MakeModelSettings	Used to find all property settings to be configured for the target computers with a given make and model. The view is a join of the MakeModelIdentity and Settings tables.
RoleAdministrators	Used to find all accounts to be made members of the local Administrators group on the target computers with a given role. The view is a join of the RoleIdentity and Settings_Administrators tables.
RoleApplications	Used to find all applications to be deployed to the target computers with a given role. The view is a join of the RoleIdentity and Settings_Applications tables.
RolePackages	Used to find all packages to be deployed to the target computers with a given role. The view is a join of the RoleIdentity and Settings_Packages tables.
RoleSettings	Used to find all property settings to be configured for the target computers with a given role. The view is a join of the RoleIdentity and Settings tables.

Related articles

- Task Sequence Steps.
- Properties.
- Scripts.
- Support Files.
- Utilities.
- MDT Windows PowerShell Cmdlets.
- Windows 7 Feature Dependency Reference.

- UDI Reference.

Feedback

Was this page helpful? Yes No

Provide product feedback

Windows 7 Feature Dependency Reference

Article • 02/12/2024

Table 8 lists the Windows 7 features, the parent feature, and any dependent features. You can use this information to determine which features and roles need to be installed to support a specific feature using the Install Roles and Features and Uninstall Roles and Features task sequence steps.

Table 8. Windows 7 Feature Dependency Reference

Expand table

Feature	Parent Feature	Dependent Features
Windows Media® Center	Media Features	Might affect other Windows features
Windows DVD Maker	Media Features	Might affect other Windows features
Windows Media Player	Media Features	Might affect other Windows features
Windows Search	N/A	Might affect other Windows features
Internet Explorer (amd64)	N/A	Might affect other Windows features
World Wide Web services	Microsoft Internet Information Services (IIS)	- Microsoft Message Queuing (MSMQ) HTTP support - Windows Communication Foundation (WCF) HTTP activation
IIS 6 WMI compatibility	IIS, Web management tools, IIS 6 management compatibility	IIS 6 scripting tooling
Microsoft .NET extensibility	IIS, World Wide Web services, application development features	- Microsoft ASP.NET - MSMQ HTTP support - WCF HTTP activation

Feature	Parent Feature	Dependent Features
Default document	IIS, World Wide Web services, common HTTP features	MSMQ HTTP support
Directory browsing	IIS, World Wide Web services, common HTTP features	MSMQ HTTP support
HTTP redirection	IIS, World Wide Web services, common HTTP features	MSMQ HTTP support
Static content	IIS, World Wide Web services, common HTTP features	- Web-based Distributed Authoring and Versioning (WebDAV) publishing - MSMQ HTTP support
Custom logging	IIS, World Wide Web services, health and diagnostics	MSMQ HTTP support
HTTP logging	IIS, World Wide Web services, health and diagnostics	MSMQ HTTP support
ODBC logging	IIS, World Wide Web services, health and diagnostics	MSMQ HTTP support
Request Monitor	IIS, World Wide Web services, health and diagnostics	MSMQ HTTP support
Tracing	IIS, World Wide Web services, health and diagnostics	MSMQ HTTP support
Static content compression	IIS, World Wide Web services, performance features	MSMQ HTTP support
Security	IIS, World Wide Web services	- Microsoft .NET extensibility - MSMQ HTTP support - WCF HTTP activation
Request Filtering	IIS, World Wide Web services, security	- Microsoft .NET extensibility - MSMQ HTTP support - WCF HTTP activation
XPS Viewer	N/A	Might affect other Windows features

Related articles

- Task Sequence Steps.
- Properties.
- Scripts.
- Support Files.
- Utilities.
- MDT Windows PowerShell Cmdlets.
- Tables and Views in the MDT DB.
- UDI Reference.

Feedback

Was this page helpful? Yes No

Provide product feedback

UDI Reference

Article • 02/12/2024

This reference provides further information about UDI and includes topics on:

- UDI concepts as described in UDI Concepts
- OSDResults as described in OSDResults Reference
- User Centric App Installer as described in User-Centric App Installer Reference
- UDI stages as described in UDI Stage Reference
- UDI tasks as described in UDI Task Reference
- UDI validators as described in UDI Validator Reference
- UDI Wizard Pages as described in UDI Wizard Page Reference

Each of these reference topics are discussed in subsequent sections.

UDI Concepts

This section contains concepts that help describe UDI, the UDI Wizard, and the UDI Wizard Designer.

Display Name

The display name is used to provide a user-friendly, descriptive name for a wizard page within the Page Library in the UDI Wizard Designer. The display name is displayed in blue text for each wizard page in the Page Library and on the **Flow** tab in the UDI Wizard Designer.

When you add a page to the Page Library, you must provide the display name. After the wizard page is added to the Page Library, you cannot change the display name.

Flow

The **Flow** tab displays the list of wizard pages within a UDI stage in the UDI Wizard Designer. You can use the **Flow** tab to perform the following tasks:

- Add a wizard page from the Page Library to a UDI stage by dragging the page from the Page Library to the UDI stage.

- Remove a wizard page from a UDI stage.
- Change the sequence of wizard pages within a UDI stage.

Page Library

The Page Library contains all the pages currently loaded in the UDI Wizard Designer. When loading a UDI Wizard configuration file, all of the wizard pages defined in the configuration file are displayed to the Page Library. The Page Library shows the wizard pages in alphabetical order by page types. Each instance of a specific page type is listed under the page type.

For example, you may need two different **WelcomePage** wizard pages for different stages. The two **WelcomePage** wizard pages will be listed under the **WelcomePage** wizard page type in the Page Library in the UDI Wizard Designer.

In addition, each wizard page instance in the Page Library indicates how many times the wizard page is used in the stage flows. When you hover over a wizard page in the Page Library, a thumbnail of the wizard page is displayed along with the stages that include that page.

Page Name

The page name is used to uniquely identify a wizard page within the Page Library in the UDI Wizard Designer. The *page name* is the name a UDI stage references so that the UDI Wizard knows which wizard page to display within a specific UDI stage. When you add a page to the Page Library, you must provide the page name. After the wizard page is added to the Page Library, you cannot change the page name. In the UDI Wizard Designer, the page name is shown at the bottom of each wizard page in the Page Library in smaller, non-bold text.

Prestaged Media Deployments

Prestaged media support is an operating system deployment feature in Configuration Manager that allows an administrator to copy and apply prestage bootable media and an operating system image to a hard disk prior to the provisioning process. This work can reduce network traffic and the time needed for the provisioning process. Prestaged media can be deployed as part of the manufacturing process or at an enterprise staging center that is not connected to the Configuration Manager environment.

For more information about prestaged media deployments, see the following resources:

- Planning for Media Operating System Deployments in Configuration Manager
- About Prestaged Media for Operating System Deployment

Stage Group

Use a stage group to group one or more stages in the UDI Wizard Designer. UDI stage groups are loosely related to MDT deployment scenarios, but there is no one-to-one correlation between the two.

Stage

A *stage* is a subset of all the pages in the UDI Wizard configuration file that an MDT deployment scenario uses. When you start the UDI Wizard using the **UDI Wizard** task sequence step, the **/stage** parameter specifies the stage to run, which in turn specifies the set of pages to use. You can preview how wizard pages will appear in a stage by selecting **Preview** in the **Preview Wizard** group on the Ribbon. You can use a UDI stage in more than one MDT deployment scenario, even though the UDI stage is defined only once in the UDI Wizard Designer. For example, the NewComputer stage can be used in the MDT New Computer and Replace Computer deployment scenarios.

Task

UDI tasks are software that is run on a wizard page to perform specific functions. In some instances, these tasks are used to verify that the target computer is ready for deployment. Other tasks can be used to perform deployment steps, such as copying configuration or result files.

> ⓘ **Note**
>
> The **Next** button on the wizard page where the tasks are run will be disabled if any of the tasks finish with warning or error completion status.

UDI includes several built-in tasks that allow you to perform most of the tasks necessary for deployment. For more information about the UDI built-in tasks, see Built-in UDI Tasks.

The **Shell Execute** built-in UDI task allows you to run any software (scripts) that can be initiated from a command line, such as Visual Basic or Windows PowerShell scripts. This functionality allows you create tasks using familiar scripting languages. For more information, see Shell Execute Task.

If your requirements go beyond scripting, you can write custom UDI tasks. *UDI tasks* are DLLs written in C++ and implement the **ITask** interface. You register the DLL with the UDI Wizard Designer task library by creating a UDI Wizard Designer configuration (.config) file and placing it in the *installation_folder*\Bin\Config folder (where *installation_folder* is the folder in which you installed MDT). For more information on developing custom UDI tasks, see the section, "Creating Custom UDI Tasks", in the *User-Driven Installation Developers Guide*.

UDI Task Sequence

You create a UDI task sequence using one of the following UDI-specific MDT task sequence templates, which run the UDI Wizard at the appropriate step in the task sequence:

- **User-Driven Installation Task Sequence.** This task sequence template is used for the New Computer, Refresh Computer, and Replace Computer MDT deployment scenarios.
- **User-Driven Installation Replace Task Sequence.** This task sequence template is the first step in a two-step process in the Replace Computer deployment scenario and is used to capture user state migration data. The second step in the two-step process is the User-Driven Installation Task Sequence task sequence template, which you use to deploy the target applications and operating system and restore the user state migration data saved during the first step of the process.

  For more information about UDI task sequence templates, see the section, "Identify the UDI Task Sequence Templates in MDT", in the MDT document *Using the Microsoft Deployment Toolkit*. For more information about these components, see the section, "Identify UDI Deployment Process Components", in the MDT document *Using the Microsoft Deployment Toolkit*, which is included with MDT.

UDI Wizard

The UDI Wizard provides the UI for collecting deployment settings that the UDI task sequences consume. The UDI Wizard is initiated as a part of a UDI task sequence and collects the necessary configuration information for customizing the deployment of the Windows client operating systems and applications. The wizard pages read their configuration settings from the UDI Wizard configuration file, which is customized using the UDI Wizard Designer.

The UDI Wizard is initiated by the **UDI Wizard** task sequence step in task sequences created using the UDI task sequence templates. The **UDI Wizard** task sequence step

runs the UDIWizard.wsf script, which in turn initiates the UDI Wizard (OSDSetupWizard.exe). Table 9 lists the UDI Wizard command-line parameters and provides a brief description of each.

Table 9. UDI Wizard Command-Line Parameters

Expand table

Parameter	Description
/preview	Allows you to preview the current configuration of the wizard by enabling the **Next** button, which allows you to move from page to page without requiring valid input.
/xml	Specifies the name of the UDI Wizard configuration file. The UDIWizard.wsf script automatically sets this parameter to the OSDSetupWizard.xml file, which is stored in the folder in which the task sequence stores log files. This parameter defaults to the config.xml file. The syntax for this parameter is as follows (where `<full_path>` is the fully qualified path to the .xml file, including the file name and extension): `/xml:<full_path>`
/stage	Specifies the name of the UDI stage to run. The UDIWizard.wsf script automatically sets this parameter to the appropriate stage, as described in UDI Stage Reference. This parameter defaults to the first stage in the UDI Wizard configuration file. The syntax for this parameter is as follows (where `<stage_name>` is the name of the stage to be run): `/stage:<stage_name>` Note: The value for <stage_name> is case sensitive.
/locale	Specifies the language to use in the UDI Wizard in the form of a locale identifier (LCID), which is represented by a numeric value. For a list of the available LCIDs, see Locale IDs Assigned by Microsoft. You would use this list to identify the language you want to use, and then provide the corresponding LCID. The syntax for this parameter is as follows (where `<locale_id>` is the numeric value of the LCID to be used): `/locale:<locale_id>`

UDI Wizard Application Configuration File

The **ApplicationPage** wizard page configures the UDI Wizard application configuration file, which maintains the list of software to be installed. This file contains an entry for each Configuration Manager application or program and package that was added using the UDI Wizard Designer.

This file has the same name as the UDI Wizard configuration file but with a .app extension. For example, if the UDI Wizard configuration file is named *Config.xml,* then the corresponding UDI Wizard application configuration file would be *Config.xml.app*. This file is the companion to the UDI Wizard configuration file.

UDI Wizard Configuration File

The UDI Wizard reads the UDI Wizard configuration file to determine the wizard pages to be displayed, the sequence of the wizard pages, any default for controls on the wizard pages, and whether the controls are enabled or disabled. This file contains all the configuration settings that are displayed in the UDI Wizard and are configured using the UDI Wizard Designer.

A separate configuration file—the UDI Wizard application configuration file—is used to configure applications to be installed on the target computer.

UDI Wizard Designer

The UDI Wizard Designer is the primary tool for customizing wizard pages for the different deployment scenarios that UDI supports. Changes made in the UDI Wizard Designer are saved in the UDI Wizard configuration file and ultimately reflected in the user experience in the UDI Wizard. The user performing the deployment will see only the wizard pages in the UDI Wizard that you have selected and configured using the UDI Wizard Designer.

Although the UDI Wizard would run with the default UDI Wizard configuration file, the wizard pages would not be configured correctly. It is recommended that you use the UDI Wizard Designer to configure the UDI Wizard user experience.

> ⓘ **Note**
>
> To run the UDI Wizard Designer, you must have the appropriate rights in Configuration Manager to access objects such as packages, applications, or images.

Validator

You use UDI validators to help ensure that the correct information is entered into text fields on wizard pages in the UDI Wizard. UDI includes several built-in validators that help you perform typical validations of fields used for entering text, such as preventing users from entering invalid characters and ensuring that the field is not empty. When a validator detects an invalid entry in a text box, a message is displayed on the wizard page, and the **Next** button is disabled until all invalid entries are resolved.

UDI includes built-in validators that allow you to perform most of the validation necessary for deployment. For more information about the UDI built-in validators, see Built-in UDI Validators.

If your requirements go beyond the built-in UDI validators, you can write custom UDI validators. *UDI validators* are DLLs written in C++ that implement the **IValidator** interface. Register the DLL with the UDI Wizard Designer validator library by creating a UDI Wizard Designer configuration (.config) file and placing it in the *installation_folder*\Bin\Config folder (where *installation_folder* is the folder in which you installed MDT). For more information on developing custom UDI tasks, see the section, "Creating Custom UDI Validators", in the MDT document *User-Driven Installation Developers Guide*.

Wizard Page

You use a wizard page to collect configuration information in the UDI Wizard. Configure UDI wizard pages using the UDI Wizard Designer. The configuration settings are stored in the UDI Wizard configuration file and are read by the wizard page when the page is initialized in the UDI Wizard.

Wizard pages are stored in the wizard Page Library, and they can be used in one or more UDI stages. This design allows you to configure a wizard page that is shared between stages once for all stages, dramatically reducing the amount of effort required and the complexity of updating wizard page configuration.

UDI includes built-in wizard pages and wizard page editors that are typically sufficient for most deployments. For more information about the built-in wizard pages, see Built-in UDI Wizard Pages.

If your requirements go beyond the built-in UDI wizard pages and corresponding wizard page editors, you can write custom UDI wizard pages and wizard page editors. UDI wizard pages are implemented as DLLs that the UDI Wizard reads. Wizard page editors are created using C++ in Visual Studio.

For more information on developing custom UDI wizard pages, see the section, "Creating Custom UDI Wizard Pages", in the MDT document *User-Driven Installation Developers Guide*.

Wizard Page Editor

You use a wizard page editor to configure a wizard page in the UDI Wizard Designer. A wizard page editor updates the wizard page configuration settings in the UDI Wizard configuration file; UDI includes a built-in wizard page editor for each built-in wizard page. For more information about the built-in wizard pages and wizard page editors, see Built-in UDI Wizard Pages.

If your requirements go beyond the built-in UDI wizard pages and corresponding wizard page editors, you can write custom UDI wizard pages and wizard page editors. UDI wizard page editors are implemented as DLLs that the UDI Wizard Designer reads. Create wizard page editors using:

- Windows Presentation Foundation version 4.0
- Microsoft Prism version 4.0
- Microsoft Unity Application Block (Unity) version 2.1

  For more information on developing custom UDI wizard page editors, see the section, "Creating Custom Wizard Page Editors", in the MDT document *User-Driven Installation Developers Guide*.

OSDResults Reference

OSDResults is a part of UDI that displays the results of a deployment performed using UDI. **OSDResults** displays the **Deployment Complete** dialog box. **OSDResults** is displayed prior to Windows logon the first time the target computer is started. The user can use **OSDResults** and the information in the **Deployment Complete** dialog box to determine the completion status of the deployment process and the configuration of the computer prior to logging on for the first time. In addition, the information in **OSDResults** can be used for troubleshooting any problems encountered during the deployment process.

You can configure some of the user interface elements for **OSDResults** using the OSDResults.exe.config file, which resides in Tools\OSDResults in the MDT files Configuration Manager package. Table 10 lists the configuration settings in the OSDResults.exe.config file.

Table 10. Configuration Settings in the OSDResults.exe.config File

Expand table

Setting	Description
headerImagePath	This setting allows you to specify the fully qualified or relative path to a .bmp file that is displayed in the header of the **OSDResults** dialog box. The default value for this setting is as follows: `images\UDI_Wizard_Banner.bmp`
backgroundWallpaper	This setting allows you to specify the fully qualified or relative path to a .jpg file that is displayed as the wallpaper in the **OSDResults** dialog box. The default value for this setting is as follows: `images\Wallpaper.jpg`
welcomeText	This setting allows you to specify the text that welcomes the user and provides information about the deployment process. It is displayed in the **OSDResults** dialog box.
completedText	This setting allows you to specify the text that indicates whether the deployment process is complete. It is displayed in the **OSDResults** dialog box.
timeoutMinutes	This setting allows you to specify the length of time the **OSDResults** dialog box is displayed prior to automatically displaying the Windows logon screen. The value for this setting is specified in minutes. The default value for this setting is zero (0), which indicates that the **OSDResults** dialog box will be displayed indefinitely until manually closed.

The following is the high-level process for how the **OSDResults** feature works in UDI:

1. A task sequence runs on the target computer.

   The task sequence is based on one of the followUDI task sequence templates:

   - **User Driven Installation Task Sequence**. This task sequence template is used for the MDT New Computer, Refresh Computer, and Replace Computer MDT deployment scenarios.

- **User-Driven Installation Replace Task Sequence**. This task sequence template is the first step in a two-step process in the MDT Replace Computer deployment scenario and is used to capture user state migration data. The second step in the two step process is the MDT New Computer deployment scenario using the **User Driven Installation Task Sequence** task sequence template, which is used to deploy the target applications and operating system and restore the user state migration data saved during the first step of the process

  For more information about the:

- UDI task sequence templates, see the section, "Identify the UDI Task Sequence Templates in MDT", in the MDT document *Using the Microsoft Deployment Toolkit*

- Relationship between MDT deployment scenarios and UDI stages, see UDI Stage Reference

2. During the task sequence, the configuration settings provided by task sequence variables and from user input in the UDI Wizard are saved in the *%DEPLOYROOT%*\Tools\OSDResults folder on the target computer (where *%DEPLOYROOT%* is the root of the folder in which the MDT files are locally cached on the target computer).

3. In the **OSD Results and Branding** group in the task sequence, the following task sequence steps are run that affect **OSDResults**:

   - **Cache OSD Results.** This task sequence step copies the contents of the *%DEPLOYROOT%*\Tools\OSDResults folder to the %WINDIR%\UDI folder on the target computer. This ensures that the contents of the OSDResults folder will be persisted after the task sequence finishes.

   - **Run OSD Results.** This tasks sequence step configures the target computer to run **OSDResults** the first time the computer starts.

4. The target computer starts for the first time, and OSDResults.exe is run prior to the Windows logon screen.

   The **Welcome** tab in the **Deployment Complete** dialog box is displayed. The **Welcome** tab provides helpful information about the deployment and contact information in the event that issues with the deployment are discovered.

   Review the information on the **Deployment Summary** and **Applications Installed** tabs to verify that the operating system and applications were installed correctly.

When you have completed reviewing these tables, select **Start Windows** to log on to Windows 7 for the first time.

> ⓘ **Note**
>
> Configuration Manager applications are not displayed on the **Applications Installed** tab. The Configuration Manager applications are detected after the user logs on to the target computer the first time.

5. The Windows logon screen is displayed, and the logon process continues normally.

   AppInstall.exe is run the first time a user logs on to the target computer. For more information on this process, see User-Centric App Installer Reference.

User-Centric App Installer Reference

The User-Centric App Installer feature in UDI is used to report any applications installed during the UDI deployment process to the Application Catalog feature in Configuration Manager. The User-Centric App Installer feature provides the link between the applications selected on the **ApplicatonPage** wizard page in the UDI Wizard and any optional Configuration Manager applications advertised to the users.

For more information on the Application Catalog feature in Configuration Manager, see Application Management in Configuration Manager.

The following is the high-level process for how the App Install feature works in UDI:

1. Configuration Manager applications are created in Configuration Manager.

   For more information about creating and managing Configuration Manager applications, see the following resources:

   - How to Create Applications in Configuration Manager
   - Operations and Maintenance for Application Management in Configuration Manager

2. The Configuration Manager user collections are created, and users are added to the collection.

   For more information about creating and managing user collections and adding users to collections, see the following resources:

   - Collections in Configuration Manager

- How to Create Collections in Configuration Manager

3. The Configuration Manager applications are deployed to the user collections.

   For more information about how to deploy the applications to user collections, see How to Deploy Applications in Configuration Manager.

4. The Configuration Manager applications are made available on the **ApplicatonPage** wizard page using the UDI Wizard Designer.

   For more information about how to make Configuration Manager applications available on the **ApplicatonPage** wizard page, see the section, "Step 5-11: Customize the UDI Wizard Configuration File for the Target Computer", in the MDT document *Quick Start Guide for User-Driven Installation*.

5. UDA is configured using one of the following methods:

   - In the Configuration Manger console (For more information about configuring UDA in the Configuration Manager console, see How to Manage User Device Affinity in Configuration Manager.)
   - On the **UDAPage** wizard page in the UDI Wizard. For more information about the **UDAPage** wizard page, see UDAPage.

     After UDA is configured, the specified user account will be the primary user for the target computer.

   > **Note**
   >
   > UDA can only be configured by UDI in the New Computer deployment scenario. It cannot be configured in the Refresh Computer or Replace Computer deployment scenarios.

6. The task sequence is run, and the user selects the Configuration Manager applications on the **ApplicatonPage** wizard page in the UDI Wizard.

   The UDI Wizard is run in the **UDI Wizard** task sequence step in the **Preinstall** group of the task sequence. When the user selects Configuration Manager applications on the **ApplicatonPage** wizard page, the wizard page creates a separate task sequence variable for each application selected.

   For more information on selecting the Configuration Manager applications on the **ApplicatonPage** wizard page in the UDI Wizard, see the section, "Step 6-4: Start

the Target Computer with the Task Sequence Bootable Media", in the MDT document *Quick Start Guide for User-Driven Installation*.

7. The task sequence installs the Configuration Manager applications that were selected in the previous step.

   The Configuration Manager applications are installed using the following task sequence steps in the **Install Applications** group in the task sequence:

   - **Convert list to two digits**
   - **Install Application**

8. The task sequence performs the following tasks in the **OSD Results and Branding** group prior to starting the target operating system for the first time:

   - Copies the information used for OSDResults.exe to the %WINDIR%\UDI folder on the target computer in the **Cache OSD Results** task sequence step
   - Records the task sequence variables created in step 6 for the Configuration Manager applications in the registry on the target computer in the **Branding to Reg** and **Branding to Reg x64** task sequence steps

     The tasks sequence variables are saved in the following location in the registry:

     HKEY_LOCAL_MACHINE\Software\Microsoft\MPSD\OSD

   - Configures the target operating system to automatically run OSDResults.exe when the computer starts prior to the Windows logon screen in the **Run OSD Results** task sequence step
   - Configures the target operating system to automatically run AppInstall.exe when a user logs on to the computer for the first time in the **Run OSD Results** task sequence step
   - Configures a task on the target operating system to remove the %WINDIR%\UDI folder one month from the date of the deployment

9. The target computer is started, and OSDResults.exe is run.

   For more information about OSDResults.exe, see OSDResults Reference.

10. A user logs on to the target computer, and AppInstall.exe starts automatically.

11. AppInstall checks whether the currently logged-on user is a primary user who was configured in UDA.

A *primary user* is a user who uses the device on a regular basis and is considered the owner, or one of the owners, of the device.

If the currently logged-on user is:

- Not a primary user, then AppInstall.exe stops
- A primary user, then AppInstall.exe reads the registry entries saved in step 8 to determine which applications were installed

12. AppIntaller connects to Configuration Manager and reads the Application Catalog using the following steps:

    a. AppInstall will wait 5 minutes after it starts to allow the Configuration Manager policies to be available.

    b. After 5 minutes, AppInstall attempts to connect to the Application Catalog.

    c. If AppInstall is unable to connect, then it will wait for a period of time before attempting to connect again.

    d. AppInstall attempts to connect up to five times before exiting.

    You can configure the connection time-out delay and the number of retries for AppInstall using the AppInstall.exe.config file, which resides in the Tools\OSDResults folder in the MDT files Configuration Manager package. Table 11 lists the configuration settings in the AppInstall.exe.config file.

Table 11. Configuration Settings in the AppInstall.exe.config File

Expand table

Setting	Description
timeoutMinutes	This setting allows you to specify the length of time for AppInstall to wait for a response from the Configuration Manager Application Catalog before timing out. The value is specified in minutes. The default value for this setting is **5**.
delayTimer	This setting allows you to specify the length of time for AppInstall to wait prior to attempting the connection to the Configuration Manager Application Catalog. The value is specified in minutes. The default value for this setting is **5**.

1. AppInstall compares the list of applications discovered in the registry with the list of applications available from the Configuration Manger Application Catalog for

the user currently logged on.

If the application discovered in the registry:

- Is available in the Application Catalog, then AppInstall.exe maps the applications and identifies the applications as existing both in the registry and in the Application Catalog. These applications will be used in the following step.

- Is not available in the Application Catalog, then AppInstall.exe does not create a mapping. These applications will not be used in the following step.

2. AppInstall uses Configuration Manager APIs to initiate the installation of the mapped applications.

   The applications used in this step were mapped in the previous step. That is to say, they were both listed in the registry and found in the Application Catalog.

3. As a part of the installation process, Configuration Manager detects whether the application is already installed.

   Because the application has already been installed, Configuration Manager records that the application has been successfully deployed to that user, and the application will be listed in Software Center for that user. Configuration Manager begins management and monitoring of the application for that user.

4. After 1 month, the task created on the target computer in step 8 runs and removes the %WINDIR%\UDI folder.

   The folder is retained for 1 month so that the primary users have an opportunity to be log on and run AppInstall.exe.

UDI Stage Reference

The MDT deployment scenarios use one or more UDI stage. Each UDI stage used in the MDT deployment scenarios is discussed in a subsequent section in the context of the MDT deployment scenario. In some MDT deployment scenarios, only one stage is used. In other MDT deployment scenarios, multiple stages are used within the scenario. For more information on the MDT deployment scenarios, see the section, "Identifying Deployment Scenarios", in the MDT document *Using the Microsoft Deployment Toolkit*.

Table 12 lists the MDT deployment scenarios and provides a brief description of each, how each scenario is selected, and which UDI stages are used in each deployment scenario. MDT automatically determines which MDT deployment scenario to use based

on the MDT task sequence template you use to create your task sequence and on how the task sequence is initiated.

Each UDI stage used in the MDT deployment scenarios is discussed in a subsequent section in the context of the MDT deployment scenario. In some MDT deployment scenarios, only one stage is used. In other MDT deployment scenarios, multiple stages are used within the scenario. For more information on the MDT deployment scenarios, see the section, "Identifying Deployment Scenarios", in the MDT document *Using the Microsoft Deployment Toolkit*.

Table 12. MDT Deployment Scenarios and UDI Stages

Expand table

Scenario	Description
New Computer	MDT for UDI automatically selects this scenario when you: - Create the advertised task sequence using the User-Driven Installation Task Sequence task sequence template - Start the task sequence in Windows PE using PXE boot, task sequence boot media, or prestaged media for the NEWCOMPUTER.Prestaged stage This scenario can be used with traditional deployments or with prestaged media deployments as supported in Configuration Manager. Run the UDI Wizard with the following UDI stages to support each type of deployment: - **NEWCOMPUTER stage.** The UDI Wizard is run with this stage in the **User-Driven Installation Task Sequence** task sequence when the operating system image is stored on distribution points. For more information, see NEWCOMPUTER Stage. - **NEWCOMPUTER.Prestage stage.** The UDI Wizard is run with this stage in the **User-Driven Installation Task Sequence** task sequence when the operating system image is stored on a local disk on the target computer (prestaged). For more information, see NEWCOMPUTER.Prestaged Stage.
Refresh Computer	MDT for UDI automatically selects this scenario when you: - Create the advertised task sequence using the User-Driven Installation Task Sequence task sequence template - Start the task sequence in the existing Windows operating system on the target computer (not in Windows PE)

Scenario	Description
	- The UDI Wizard is run with the REFRESH stage to support this deployment scenario. For more information, see REFRESH Stage.
Replace Computer	This scenario includes an existing computer and a replacement computer. A separate task sequence is created and run on each computer as described in the following process: - **On the existing computer.** MDT for UDI automatically selects this portion of the scenario when you: - Create the advertised task sequence using the User-Driven Installation Replace Task Sequence task sequence template Start the task sequence in the existing Windows operating system on the target computer (not in Windows PE) The UDI Wizard is run with the following UDI stages to support this deployment scenario: - **REPLACE stage.** This stage is run in the existing Windows operating system and captures configuration information from within Windows. - **REPLACE.WinPE stage.** This stage is run in Windows PE and completes the capturing of configuration information from the existing computer—for example, running USMT and capturing the user state migration data. The user state is captured to a network shared folder or to a local USB drive. For more information on the REPLACE and REPLACE.WinPE stages, see REPLACE and REPLACE.WinPE Stages. - **On the replacement computer.** This portion of the scenario is identical to the New Computer scenario, except that the user state captured in the previous step is restored. MDT for UDI automatically selects this portion of the scenario when you: - Create the advertised task sequence using the User-Driven Installation Task Sequence task sequence template - Start the task sequence in Windows PE using PXE boot, task sequence boot media, or prestaged media for the NEWCOMPUTER.Prestaged stage. This portion of the scenario can be used with traditional deployments or with prestaged media deployments as supported in Configuration Manager. As a part of this portion of the scenario, the user state migration data is restored. The UDI Wizard is run with the following UDI stages to support each type of deployment: - **NEWCOMPUTER stage.** The UDI Wizard is run with this stage in the **User-Driven Installation Task Sequence** task sequence when the operating system image is

Scenario	Description
	stored on distribution points. For more information, see NEWCOMPUTER Stage. - **NEWCOMPUTER.Prestage stage.** The UDI Wizard is run with this stage in the **User-Driven Installation Task Sequence** task sequence when the operating system image is stored on a local disk on the target computer (prestaged). For more information, see NEWCOMPUTER.Prestaged Stage.

NEWCOMPUTER Stage

Figure 1 illustrates the use of the NEWCOMPUTER stage in a task sequence created using the User-Driven Installation Task Sequence task sequence template. The primary difference between the task sequences calling the NEWCOMPUTER stage and the NEWCOMPUTER.Prestaged stage is that the task sequence calling the NEWCOMPUTER.Prestaged stage does not run the **Apply Operating System Image** task sequence step, because the operating system image is already located on the target computer.

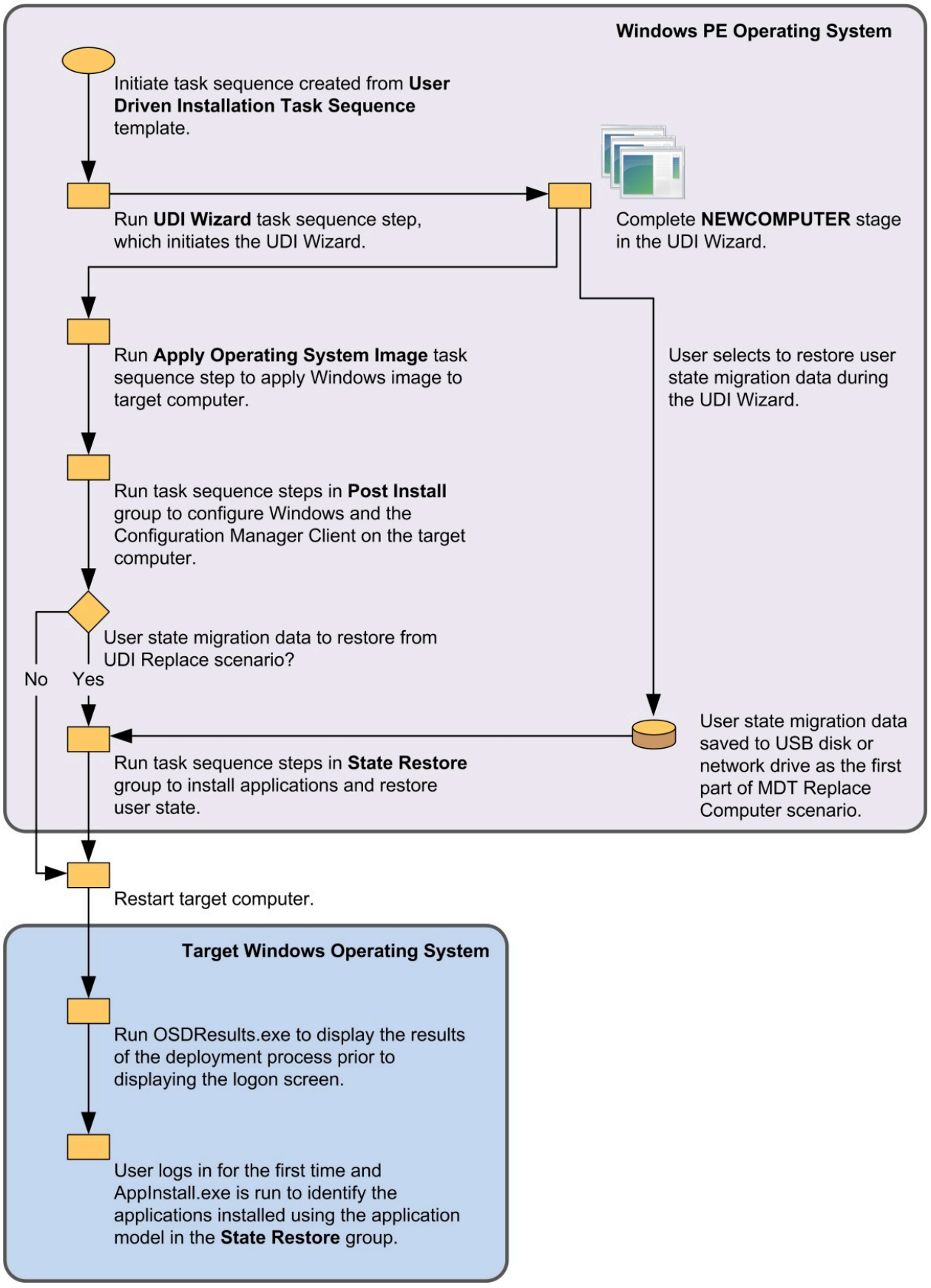


Figure SEQ Figure * ARABIC 1. Process flow for the NEWCOMPUTER stage

NEWCOMPUTER.Prestaged Stage

Figure 2 illustrates the high-level process flow for the NEWCOMPUTER.Prestaged stage in a task sequence created using the User-Driven Installation Task Sequence task

sequence template. The primary difference between the task sequences calling the NEWCOMPUTER stage and the NEWCOMPUTER.Prestaged stage is that the task sequence calling the NEWCOMPUTER.Prestaged stage does not run the **Apply Operating System Image** task sequence step, because the operating system image is already located on the target computer.

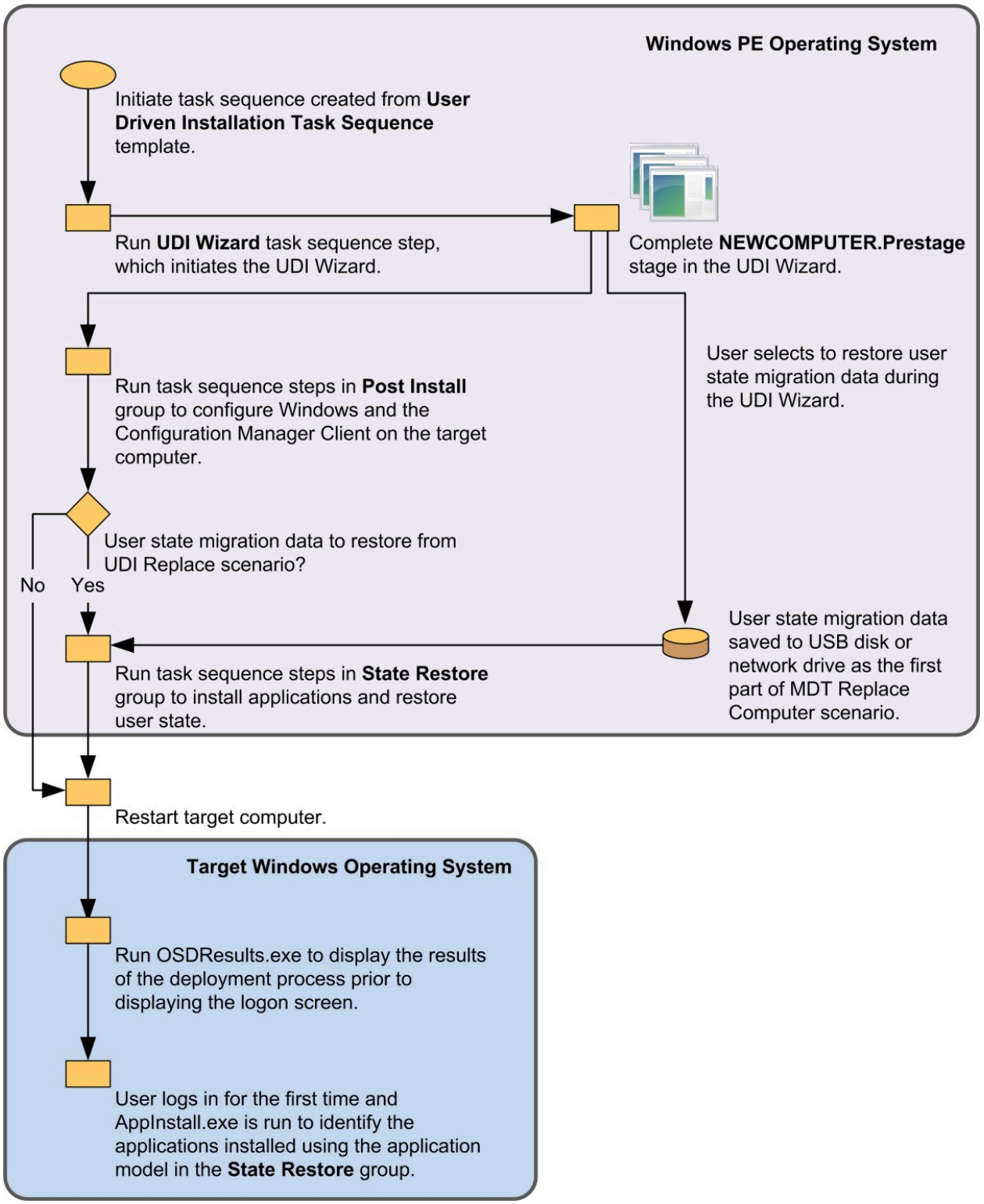


Figure 2. Process flow for the NEWCOMPUTER.Prestaged stage

REFRESH Stage

Figure 3 illustrates the high-level process flow for the REFRESH stage in a task sequence created using the User-Driven Installation Task Sequence task sequence template.

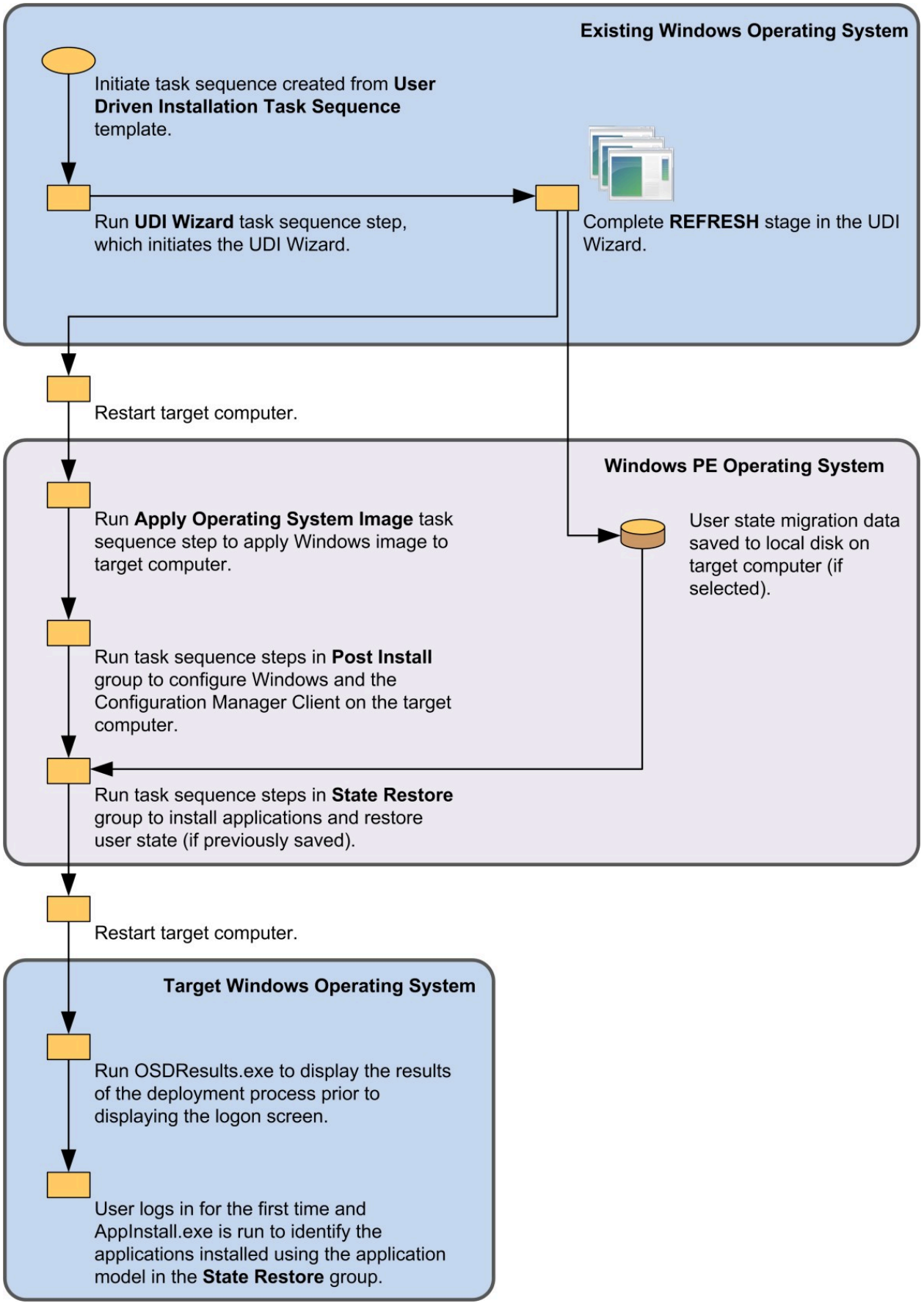


Figure SEQ Figure * ARABIC 3. Process flow for the REFRESH stage

REPLACE and REPLACE.WinPE Stages

Figure 4 illustrates the high-level process flow for the REPLACE and REPLACE.WinPE stages in a task sequence created using the User-Driven Installation Replace Task Sequence task sequence template.

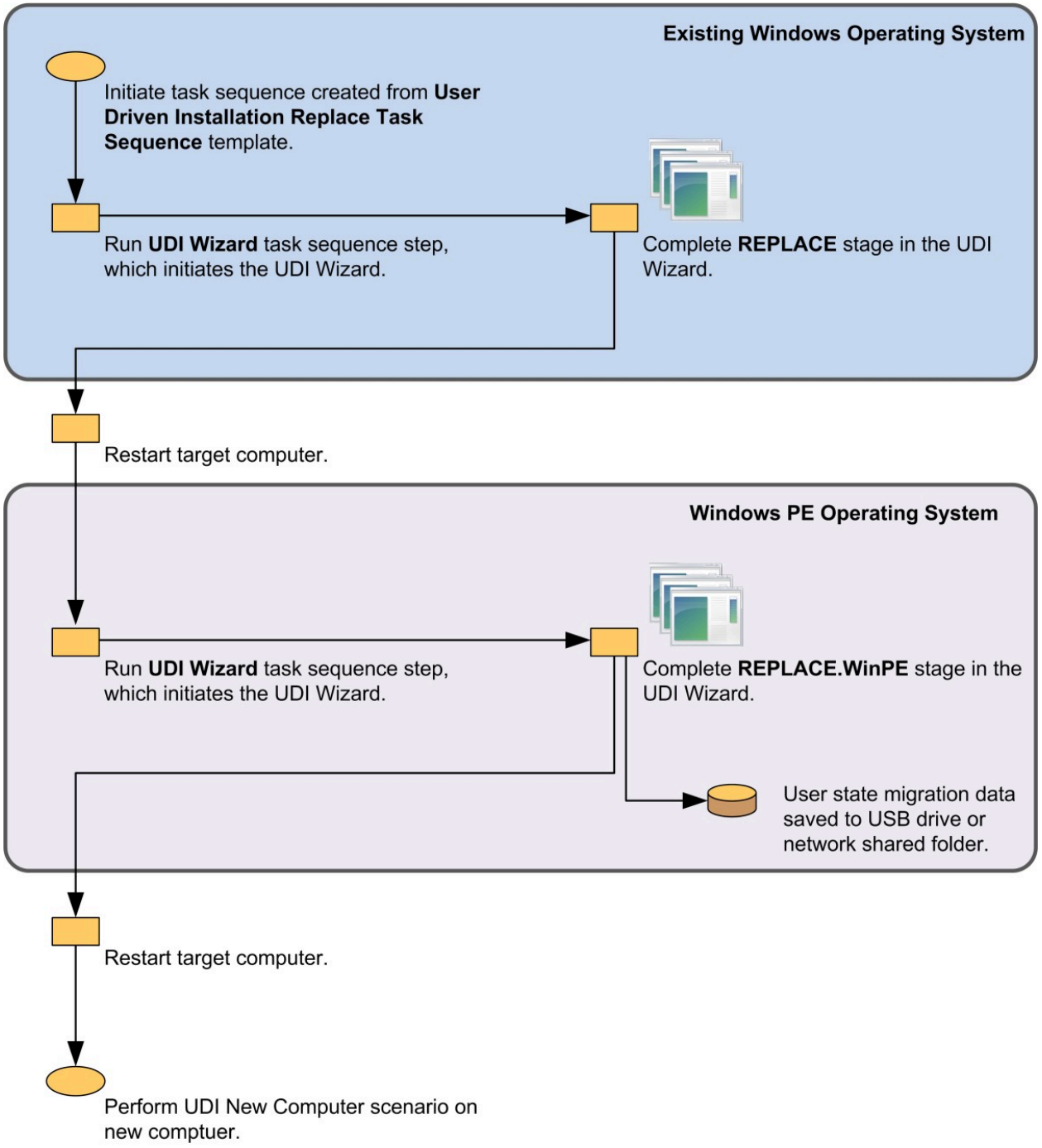


Figure 4. Process flow for the REPLACE and REPLACE.WinPE stages

UDI Task Reference

UDI tasks are software that is run on a wizard page that perform specific functions. In some instances, these tasks are used to verify that the target computer is ready for

deployment. Other tasks can be used to perform deployment steps, such as copying configuration or result files.

> ⓘ **Note**
>
> The **Next** button on the wizard page where the tasks are run will be disabled if any of the tasks finish with warning or error completion status.

This reference includes:

- An overview of UDI tasks, as described in UDI Task Overview
- A description of the configuration settings for UDI tasks, as described in UDI Task Configuration Settings
- A description of the built-in UDI validators that are provided with MDT, as described in Built-in UDI Tasks

UDI Task Overview

UDI tasks allow you to run software on the target computer that helps with the deployment process. UDI includes several built-in tasks that help you perform common tasks, such as ensuring that the target computer is not running on a battery and is connected to a wired network connection.

In addition to the built-in UDI tasks, you can create custom UDI tasks using the UDI software development kit (SDK). For more information about creating custom UDI tasks using the UDI SDK, see *User-Driven Installation Developers Guide*.

UDI Task Configuration Settings

You manage tasks using the UDI Wizard Designer. You can add tasks, remove tasks, and edit the configuration of a task in the UDI Wizard Designer. The configuration settings for a task are stored in the UDI Wizard configuration file and are read by the UDI Wizard when the wizard page that contains the task is displayed.

UTI tasks have some configuration settings that are common to all UDI tasks, as listed in Table 13. For the configuration settings that are specific to each UDI task, see the corresponding section in Built-in UDI Tasks.

Table 13. Configuration Settings Common to All UDI Tasks

Expand table

Task	Description
Bitmap Filename	This parameter specifies the graphic used to indicate the task type.
Display Name	This specifies the name of the task, which is displayed on the wizard page when the task is run.
Exit Code Values	This specifies a list of possible return codes for the task. An item exists in the list for each possible return code.
Error Code Values	This specifies a list of possible unexpected exceptions that may be encountered (thrown) by the task. An item exists in the list for each possible exception.

Built-in UDI Tasks

Table 14 lists the built-in UDI tasks. Each built-in UDI task is discussed in a subsequent section.

Table 14. Built-in UDI Tasks

Expand table

Task	Description
AC Power Check	This UDI task is used to identify whether the target computer is connected to AC power, not solely on battery.
Application Discovery	This UDI task is used to discover applications that are installed on the target computer.
CheckSMSFolderOnUSB	This UDI task is used to determine whether the _SMSTaskSequence folder is located on a USB drive on the target computer.
Copy Files Task	This UDI task is used to copy files while the UDI Wizard is running on the target computer.
Shell Execute Task	This UDI task is used to run software that can be initiated from a command line.

Task	Description
Wired Network Check	This UDI task is used to identify whether the target computer is connected to a wired network, not connected using a wireless network connection.

AC Power Check

Use this UDI task to identify whether the target computer is connected to AC power. This task uses only those parameters common to all UDI tasks. For more information about these parameters, see UDI Task Configuration Settings.

Table 15 lists the error and exit codes that the **AC Power Check** task generates.

Table 15. Error and Exit Codes for the AC Power Check Task

Expand table

Exit or error code	Value	Status
Exit	**0**	**Success**, which indicates that the target computer is plugged into AC power
Exit	*	**Error**, which indicates that the target computer is not plugged into AC power

Application Discovery

Use this UDI task to discover applications that are installed on the target computer.

Table 16 lists the parameters that the **Application Discovery** task uses.

Table 16. Parameters Used by the Application Discovery Task

Expand table

Task	Description
Readcfg	This parameter specifies the fully qualified or relative path to the location of the .app file that has a list of applications for the task to discover. The .app file contains the list of available software items from which the user can select. The **Application Discovery** task reads the .app file and determines whether any of these software items is installed. If a software item is installed, the item is added to the file specified in the **Writecfg** parameter. Ensure that this parameter uses the same location and file name as the ApplicationPage wizard page.
Writecfg	This parameter specifies the fully qualified or relative path to the location of the .xml file that contains a list of the applications discovered by the task.
Log	This parameter specifies the fully qualified or relative path to the location of the log file generated by this task. The file name of the log file is AppDiscovery.log.

In addition to the parameters in Table 16, this task uses the parameters common to all UDI tasks. For more information about these common parameters, see UDI Task Configuration Settings.

Table 17 lists the error and exit codes that the **Application Discovery** task generates.

Table 17. Error and Exit Codes for the Application Discovery Task

Expand table

Exit or error code	Value	Status and description
Exit	**0**	**Success**, which indicates that the task successfully scanned for applications
Exit	*****	**Warning**, which indicates that the application discovery engine could not be run for some unknown reason
Exit	**1**	**Warning**, which indicates that the application discovery engine encountered one or more warnings
Exit	**16777216**	**Warning**, which indicates that critical problems were encountered while initializing the application discovery engine
Exit	**33554432**	**Warning**, which indicates that critical problems were encountered while processing the application master list

CheckSMSFolderOnUSB

Use this UDI task to identify whether the _SMSTaskSequence folder is located on a USB drive on the target computer. By default, the Configuration Manager task sequencer places the _SMSTaskSequence folder on the drive with the most available free disk space. This can cause problems later in the deployment process if the USB drive is removed.

This task checks to see whether the folder is located on a USB drive and prevents the deployment from proceeding if it is. This task uses only those parameters common to all UDI tasks. For more information about these parameters, see UDI Task Configuration Settings.

If the _SMSTaskSequence folder is located on a USB drive, this task fails and prevents the deployment from continuing. To resolve this issue and perform the deployment, complete the following steps:

1. Disconnect the USB drive from the target computer before starting the task sequence.
2. Start the task sequence.
3. Wait until the UDI Wizard starts.
4. Connect the USB drive.
5. Complete the UDI Wizard.

Table 18 lists the error and exit codes that the **CheckSMSFolderOnUSB** task generates.

Table 18. Error and Exit Codes for the CheckSMSFolderOnUSB Task

Expand table

Exit or error code	Value	Status
Exit	**0**	**Success**, which indicates that the _SMSTaskSequence folder is not located on a USB drive and the deployment can continue.
Exit	*	**Error**, which indicates that the _SMSTaskSequence folder is located on a USB drive and the deployment cannot continue.

Copy Files Task

Use this UDI task to copy files while the UDI Wizard is running on the target computer.

Table 19 lists the parameters that the **Copy Files** task uses.

Table 19. Parameters Used by the Copy Files Task

Expand table

Task	Description
Source	This parameter specifies the fully qualified or relative path to the source file, which can contain wildcards to copy multiple files using a single task.
Destination	This parameter specifies the fully qualified or relative path to the destination file without a file name.

In addition to the parameters in Table 19, this task uses parameters common to all UDI tasks. For more information about these parameters, see UDI Task Configuration Settings.

Table 20 lists the error and exit codes that the **Copy Files** task generates.

Table 20. Error and Exit Codes for the Copy Files Task

Expand table

Exit or error code	Value	Status and description
Exit	**0**	**Success**, which indicates that the copy process succeed
Exit	*	**Error**, which indicates that the copy process failed
Error	**-1**	**Error**, which indicates that the copy process failed

Shell Execute Task

Use this UDI task to run software that can be initiated from a command line.

Table 21 lists the parameters that the **Shell Execute** task uses.

Table 21. Parameters Used by the Shell Execute Task

Expand table

Task	Description
Filename	This parameter specifies the fully qualified or relative path to the command for the task to run.
Parameters	This parameter specifies the command-line parameters that are to be provided when running the command.

In addition to the parameters in Table 21, this task uses parameters common to all UDI tasks. For more information about these parameters, see UDI Task Configuration Settings.

You can also run custom Visual Basic scripts designed to run in cscript.exe using the **Shell Execute** task. To run Visual Basic scripts, perform the following steps:

1. Type the following text in the **Filename** parameter:

   Windows Command Prompt

   ```
   %windir%\system32\cscript.exe
   ```

2. Type name of the Visual Basic script file (.vbs file) in the **Parameters** parameter, including any command-line parameters for the script.

   For example, to run a Visual Basic script named *SelfTest.vbs* with a parameter value of **Debug**, type the following (where *script_path* is the fully qualified path to the SelfTest.vbs file):

   Windows Command Prompt

   ```
   <script_path>\SelfTest.vbs Debug
   ```

   Table 22 lists the common error and exit codes that the **Shell Execute** task generates.

> **Note**

Each specific task based on the **Shell Execute** task has a unique set of error and exit codes. Please check the return codes for the software you are running using this task.

Table 22. Common Error and Exit Codes for the Shell Execute Task

Expand table

Exit or error code	Value	Status and description
Exit	0	**Success**, which indicates that the task finished successfully
Exit	*	**Error**, which indicates that the task failed

Wired Network Check

Use this UDI task to determine whether the target computer is connected to a wired network, not using a wireless network connection. This task only uses parameters common to all UDI tasks. For more information about these parameters, see UDI Task Configuration Settings.

Table 23 lists the common error and exit codes that the **Wired Network Check** task generates.

Table 23. Error and Exit Codes for the Wired Network Check Task

Expand table

Exit or error code	Value	Status and description
Exit	0	**Success**, which indicates that the target computer is connected to a wired network
Exit	*	**Error**, which indicates that the target computer is not connected to a wired network

UDI Validator Reference

UDI validators are used to validate values entered in text fields on wizard pages. When a UDI validator detects an invalid entry, a message is displayed for the first error encountered at the bottom of the wizard page. The next validation error message, if any, is displayed after you resolve the first validation error. This process continues until all validation errors are resolved. The **Next** button is disabled until all validation errors on the wizard page are resolved.

This reference includes:

- An overview of UDI validators, as described in UDI Validator Overview
- A description of the built-in UDI validators provided with MDT, as described in Built-in UDI Validators

UDI Validator Overview

UDI validators are used to help ensure that users provide the correct information in the text fields on wizard pages in the UDI Wizard. UDI includes several built-in validators that help you perform typical validations of fields used for entering text, such as preventing users from entering invalid characters or ensuring that the field is not empty.

In addition to the built-in UDI validators, you can create custom UDI validators using the UDI SDK. For more information about creating custom UDI validators using the UDI SDK, see the MDT document *User-Driven Installation Developers Guide*.

Built-in UDI Validators

Table 24 lists the built-in UDI validators. Each built-in validator is discussed in a subsequent section. When a validator detects an invalid entry in a text box, a message is displayed on the wizard page, and the **Next** button is disabled until all invalid entries are resolved.

Table 24. Built-in UDI Validators

Expand table

Validator	Description
InvalidChars	This validator identifies any invalid characters that have been entered from a list that you configure.
NamedPattern	This validator helps ensure that the text follows a predefined pattern.

Validator	Description
NonEmpty	This validator is used to require text in a field.
RegEx	This validator allows you ensure that the text matches a regular expression that you specify as a part of the validator.

InvalidChars

This validator prevents users from entering specific characters. The **Message** box allows you to enter a message that is displayed if the text field contains any of the invalid characters. The **Invalid Characters** box allows you to enter the characters that are considered invalid. The characters are entered without spaces between them.

NamedPattern

This validator helps ensure that the text follows a predefined pattern. The **Message** box allows you to enter a message that is displayed if the text field does not match the named pattern. The **Named Pattern** box allows you to enter the name of the predefined pattern and must be **Username**, **ComputerName**, or **Workgroup**. The names are case insensitive.

NonEmpty

Use this validator to require text in a field. The **Message** box allows you to enter a message that is displayed if the text field is empty.

RegEx

This validator allows you ensure that the text matches a regular expression that you specify as a part of the validator. The **Message** box allows you to enter a message that is displayed if the text field does not match the regular expression. The **Regular Expression** box allows you to enter the regular expression used for the validation. For more information about how to build regular expressions for this validator, see TR1 Regular Expressions.

UDI Wizard Page Reference

You add a UDI wizard page to stages from the Page Library in the UDI Wizard Designer. UDI wizard pages are displayed in the UDI Wizard.

This reference includes:

- An overview of UDI wizard pages, as described in UDI Wizard Page Overview
- A description of the built-in UDI wizard pages that are provided with MDT, as described in Built-in UDI Wizard Pages

UDI Wizard Page Overview

Wizard pages are displayed in the UDI Wizard and collect the information required to complete the deployment process. You create wizard pages using C++ in Visual Studio. The custom wizard pages are implemented as DLLs that the UDI Wizard reads.

Each built-in UDI wizard page has a corresponding UDI wizard page editor, which you use to configure the wizard page in the UDI Wizard Designer.

In addition to the built-in UDI wizard pages, you can create custom UDI wizard pages using the UDI SDK. For more information about creating custom UDI wizard pages using the UDI SDK, see the MDT document *User-Driven Installation Developers Guide*.

Each wizard page can reference the following types of variables:

- Task sequence variables
- Memory variables
- Environment variables

  You can reference task sequence and environment variables by bracketing the variable using percent signs (%), such as *%OSDImageIndex%*. You can reference memory variables by bracketing the variable using dollar signs ($), such as *$VolumeArchitecture$*.

> **Note**
>
> If a task sequence variable and an environment variable both have the same name, then the task sequence variable takes precedence over the environment variable.

Table 25 lists the memory variables that are set when the UDI Wizard starts, the description of the variables, and whether the UDI Wizard reads or writes the variables during startup.

Table 25. Memory Variables Set by the UDI Wizard at Startup and Their Descriptions

Expand table

Variable	Read	Write
LogPath Specifies the fully qualified path to the log files for the UDI Wizard. You can set this variable to one of the following values: - The value in the **_SMSTSLogPath** task sequence variable - The value of the %TEMP% environment variable if the **_SMSTSLogPath** task sequence variable is not set	No	Yes
WizardConfigFilename Specifies the name of the UDI Wizard configuration file currently in use. The **ApplicationPage** wizard page reads the value of this variable to find the corresponding .app file, which contains the list of applications. For example, if the UDI Wizard configuration file is named *config.xml,* then the wizard page will look for the corresponding .app file (config.xml.app).	No	Yes

Built-in UDI Wizard Pages

Table 26 lists the built-in UDI wizard pages. Each built-in UDI wizard page is discussed in a subsequent section.

Table 26. Built-in Wizard Pages and Their Descriptions

Expand table

Wizard page	Description
AdminAccounts	Use this wizard page to set the password for the local administrator account and add other users to the local Administrators group on the target computer.
ApplicationPage	Use this wizard page to configure the list of applications that can be installed during the setup process. These applications can include applications or packages and programs from Configuration Manager.

Wizard page	Description
BitLockerPage	Use this wizard page to configure BitLocker settings for the target computer.
ComputerPage	Use this wizard page to configure the computer name of the target computer, the domain or workgroup to join, and the credential to be used when joining a domain.
ConfigScanPage	Use this wizard page to run UDI tasks that scan the configuration of the target computer to determine whether the target computer is ready for the deployment of the operating system image. This readiness includes having sufficient system resources and ensuring that any prerequisite software is installed and configured properly.
LanguagePage	Use this wizard page to determine which language pack should be installed, the default language for the target operating system, the keyboard locale, and the time zone in which the computer will be physically located.
ProgressPage	Use this wizard page to run UDI tasks that capture the user state migration data from the target computer.
RebootPage	Use this wizard page to notify the user that the target computer is going to be restarted. You can configure the notification message using the UDI Wizard Designer.
SummaryPage	Use this wizard page to notify the user about the configuration options that were selected while running the UDI Wizard. The configuration information displayed on this wizard page is automatically collected from other wizard pages. Some fields on other wizard pages allow you to configure the caption (label) displayed on this wizard page using the UDI Wizard Designer.
UDAPage	Use this wizard page to configure the UDA between the target computer and a specified user. Defining affinity between a computer and a user allows automatic installation of software that is deployed to a user. The UDA feature is only available in Configuration Manager and in the UDI New Computer scenario.
UserStatePage	Use this wizard page to configure the settings for capturing or restoring user state migration data. This wizard page allows the user to select the location to capture user state migration to or restore user state migration data from.
VolumePage	Use this wizard page to configure the settings for the disk volume on target computer where the operating system will be deployed. These settings include selecting the target operating system, selecting the target drive, selecting any Windows installation, and determining whether the target drive should be formatted as a part of the deployment process.
WelcomePage	Use this wizard page to provide information to the user about UDI Wizard and the deployment process. You can configure the notification message using the UDI Wizard Designer.

AdminAccounts

Use this wizard page to set the password for the local administrator account and to add other user to the local Administrators group on the target computer.

Task Sequence Variables

Table 27 lists the **AdminAccounts** task sequence variables with the description and determines whether the variable is read by the wizard page, written by the wizard page, or can be configured in the UDI Wizard configuration file.

Table 27. AdminAccounts Task Sequence Variables

Expand table

Variable	Read	Write	Config
OSDAddAdmin Specifies a list of additional user names to be added to the local Administrators group on the target computer.	Yes	Yes	Yes
OSDLocalAdminPassword Specifies the passwords for the local built-in Administrator account on the target computer.	Yes	Yes	Yes

ApplicationPage

Use this wizard page to configure the list of application software that can be installed during the setup process. These applications can include applications or packages and programs from Configuration Manager.

> **Note**
>
> If applications appear to be disabled, the application may require administrator approval but has not yet been approved. If the **Require administrator approval if users request this application** check box is selected for the application, verify that the application has been approved. For more information, see How to Deploy Applications in Configuration Manager.

Task Sequence Variables

Table 28 lists the **ApplicationPage** task sequence variables with the description and whether the variable is read by the wizard page, written by the wizard page, or can be configured in the UDI Wizard configuration file.

Table 28. ApplicationPage Task Sequence Variables

Expand table

Variable	Read	Write	Config
ApplicationBaseVariable Specifies the name used as the base for the task sequence variable names created for each Configuration Manager application selected on the **ApplicationPage** wizard page. This variable is configured using the **Edit Software Settings** button in the **Edit Settings** group on the Ribbon in the UDI Wizard Designer. A separate task sequence variable is created for each application selected on this page. The default value for this variable is **APPLICATIONS**. So, for example, the default names of the task sequence variables created for each application selected on this page will be *APPLICATIONS001, APPLICATIONS002, APPLICATIONS003,* and so forth.	No	Yes	Yes
OSDApplicationList Specifies the list of application identifiers that should be initially selected. The variable contains a list of numeric values separated by semicolons (;). The application identifiers are found in the **Id** attribute of the **Application** element in the UDI Wizard application configuration file (UDIWizard_Config.xml.app). There is a separate **Application** element for each application displayed in this wizard page.	Yes	No	No
OSDArchitecture Specifies the processor architecture of the target computer. The **ApplicationPage** wizard page uses this variable to filter the available applications when the **VolumeArchitecture** memory variable has not been set. However, if the **VolumeArchitecture** memory variable has been set, it always takes precedence over this task sequence variable for filtering the available applications.	Yes	No	No

Variable	Read	Write	Config
The value for this variable can be: - **x86**, which indicates a 32-bit processor architecture - **amd64**, which indicates a 64-bit processor architecture			
OSDBaseVariableName Specifies the name used as the base for the task sequence variable names created for each Configuration Manager package and program selected on the **ApplicationPage** wizard page. This variable is configured using the **Edit Software Settings** button in the **Page Behavior** group on the Ribbon in the UDI Wizard Designer. A separate task sequence variable is created for each application selected on this page. The default value for this variable is **PACKAGES**. So, for example, the default names of the task sequence variables created for each application selected on this page will be *PACKAGES001, PACKAGES002, PACKAGES003,* and so forth.	No	Yes	Yes

Memory Variables

Table 29 lists the **ApplicationPage** memory variables with the description and whether the variable is read or written by the wizard page.

Table 29. ApplicationPage Memory Variables

Expand table

Variable	Read	Write
VolumeArchitecture Specifies the processor architecture of the target operating system image to be deployed (whether the image contains a 32-bit or 64-bit operating system). When this page is displayed, it checks to see if this variable has changed. If the variable has changed since the last time the wizard page was displayed, the wizard page filters the programs available for selection based on architecture of the target operating system. For example, if a 32-bit operating system is to be deployed, then the wizard page removes (filters) any 64-bit applications from the list of available applications on the wizard page.	Yes	No
WizardConfigFilename Specifies the name of the UDI Wizard configuration file currently in use. If the	Yes	No

Variable	Read	Write
value of the **Link.Uri** setter property is empty, the **ApplicationPage** wizard page reads the value of this variable to find the corresponding .app file, which contains the list of applications. For example, if the UDI Wizard configuration file is named *config.xml,* then the wizard page will look for the corresponding .app file (config.xml.app). This variable is set when the UDI Wizard starts. The **Link.Uri** setter property is set on the **Software Settings** dialog box, which can be opened using the **Edit Software Settings** button in the **Page Behavior** group on the Ribbon in the UDI Wizard Designer.		

BitLockerPage

This wizard page is used to configure BitLocker settings for the target computer.

Task Sequence Variables

Table 30 lists the BitLockerPage task sequence variables with a description and whether the variable is read by the wizard page, written by the wizard page, or can be configured in the UDI Wizard configuration file.

Table 30. BitLockerPage Task Sequence Variables

Expand table

Variable	Read	Write	Config
BDEInstallSuppress Specifies whether BitLocker installation should be suppressed. If the variable is set to: - **YES**, then the **Enable BitLocker** check box is selected and the installation is performed - **NO**, then the **Enable BitLocker** check box is cleared and the installation is not performed	Yes	Yes	Yes
BDEKeyLocation Specifies the fully qualified path to the location where the BitLocker encryption keys are stored, which can be a local or UNC path. This variable is set to the value of the **KeyLocation** setter value in the UDI	No	Yes	No

Variable	Read	Write	Config
Wizard configuration file for the **BitLockerPage**. This variable is only considered valid when the **OSDBitLockerMode** is set to **TPMKEY** or **KEY**.			
BDEPin Specifies the BitLocker PIN value if the **Enable BitLocker using TPM and Pin** option is selected.	Yes	Yes	Yes
OSDBitLockerCreateRecoveryPassword Specifies whether a BitLocker recovery password should be stored in AD DS. If the variable is set to: - **AD**, then the **In Active Directory** option is selected and the recovery keys will be stored in AD DS (recommended) - **NONE**, then the **Do not create a recovery key** option is selected and the recovery keys will not be stored in AD DS (not recommended)	No	Yes	No
OSDBitLockerMode Specifies the mode to be used when enabling BitLocker on the target computer. Valid values include: - **TPM.** This value indicates that the **Enable BitLocker using TPM only** option is selected and that only TPM will be used when enabling BitLocker on the target computer. - **TPMPIN.** This value indicates that the **Enable BitLocker using TPM and Pin** option is selected and that TPM and a user-specified PIN will be used when enabling BitLocker on the target computer. - **TPMKEY.** This value indicates that the **Enable BitLocker using TPM and Startup Key** option is selected and that TPM and a startup key will be used when enabling BitLocker on the target computer. - **KEY.** This value indicates that the **Enable BitLocker using only an External Startup Key** option is selected and that only an external startup key will be used when enabling BitLocker on the target computer.	No	Yes	No
OSDBitLockerStartupKeyDrive Specifies the drive letter where the BitLocker external startup key will be stored on the target computer. This variable is only considered valid when **OSDBitLockerMode** is set to **TPMKEY** or **KEY**.	No	Yes	No
OSDBitLockerWaitForEncryption Specifies whether the task sequence should wait until BitLocker	Yes	Yes	Yes

Variable	Read	Write	Config
encryption finishes. If the variable is set to: - **YES**, then the **Wait for BitLocker Encryption to complete on all drives before continuing** check box is selected and the task sequence will wait until the installation is complete - **NO**, then the **Wait for BitLocker Encryption to complete on all drives before continuing** check box is cleared and the task sequence will not wait until the installation is complete			

Configuration Variables

Table 31 lists the **BitLockerPage** configuration variables with a description and whether the variable is read by the wizard page, written by the wizard page, or can be configured in the UDI Wizard configuration file.

Table 31. BitLockerPage Configuration Variables

Expand table

Variable	Read	Write	Config
KeyLocation Specifies the fully qualified path to the location where the BitLocker encryption keys are stored, which can be a local or UNC path. This configuration value is used to set the value of the **BDEKeyLocation** task sequence variable for the **BitLockerPage**. This variable is only considered valid when **OSDBitLockerMode** is set to **TPMKEY** or **KEY**.	Yes	No	Yes

ComputerPage

Use this wizard page to configure the computer name of the target computer, the domain or workgroup to join, and the credentials to be used when joining a domain. When you configure this page to join the target computer to a domain, this wizard page will validate the credentials you provide for joining the domain in AD DS by default. Then, this wizard page attempts to modify a computer object in AD DS to verify that the user credentials provided on this page have permissions to create or modify the computer object. You can disable either of these behaviors. If you disable the validation of the credentials, then the verification of permissions for creating or modifying computer objects is also disabled. Both of these validations occur when the **Next** button

is selected. If either of the validations encounters an error, an error message will be displayed and this page will continue to be displayed.

The following is the order of precedence for determining the default computer name:

1. If the **UserExistingComputerName** value in the UDI Wizard configuration file is set to **TRUE**, then the existing computer name is used (if present).
2. If the **OSDComputerName** task sequence variable is set, then the computer name in that variable is used.
3. If a default value is specified for the computer name in the UDI Wizard configuration file, then that value is used.

Task Sequence Variables

Table 32 lists the **ComputerPage** task sequence variables with a description and whether the variable is read by the wizard page, written by the wizard page, or can be configured in the UDI Wizard configuration file.

Table 32. ComputerPage Task Sequence Variables

Expand table

Variable	Read	Write	Config
OSDComputerName Specifies the name of the target computer. The value of this variable is set in the **Computer name** box.	Yes	Yes	Yes
OSDDomainName Specifies the name of the domain to which the target computer is to be joined. The value of this variable is set in the **Domain** box.	Yes	Yes	Yes
OSDDomainOUName Specifies the name of the OU within the domain to which the target computer object is to be placed. The value of this variable is set in the **Organizational Unit** box.	Yes	Yes	Yes
OSDJoinAccount	Yes	Yes	Yes

Variable	Read	Write	Config
Specifies the user account used to join the target computer to the domain. The value for this variable is set in the **User name** box.			
OSDJoinPassword Specifies the password for the user account used to join the target computer to the domain. The value for this variable is set in the **Password** and **Confirm password** boxes.	Yes	Yes	Yes
OSDNetworkJoinType Specifies if the target computer is to be joined to a workgroup or a domain. If the value is set to: - **0**, then the **Domain** option is selected and the target computer will be joined to a domain - **1**, then the **Workgroup** option is selected and the target computer will be joined to a workgroup	No	Yes	No
SMSTSAssignUsersMode Specifies the mode for configuring user affinity in Configuration Manager. Use this variable to configure the behavior of creating affinity between the target computer and user accounts in the **SMSTSUdaUsers** task sequence variable. If this variable is not specified prior to displaying this page, the value of this variable is set to **Pending**. Possible values for this variable include: - **Auto.** The affinity processing is automatically approved by Configuration Manager. - **Pending.** The affinity processing rules will require approval by a Configuration Manager administrator. - **Disabled.** No affinity processing will occur.	No	Yes	No

Configuration Variables

Table 33 lists the **ComputerPage** configuration variables with a description and whether the variable is read by the wizard page, written by the wizard page, or can be configured in the UDI Wizard configuration file.

Table 33. ComputerPage Configuration Variables

Expand table

Variable	Read	Write	Config
ADComputerObjectCheck Specifies whether the **ComputerPage** wizard page will validate that the credentials provided have the appropriate permissions to modify a computer object in AD DS prior to continuing to the next wizard page. Note: This configuration setting is ignored if **ADCredentialCheck** is set to **FALSE**. If the value is set to: - **TRUE**, then the **Active Directory Computer Object Check** check box is selected in the wizard page editor in the **Domain Join Credentials** section in the UDI Wizard Designer, and permissions to modify a computer object for the credentials are validated - **FALSE**, then the **Active Directory Computer Object Check** check box is cleared in the wizard page editor in the **Domain Join Credentials** section in the UDI Wizard Designer, and permissions to modify a computer object for the credentials are not validated	Yes	No	Yes
ADCredentialCheck Specifies whether the **ComputerPage** wizard page will validate the credentials provided for joining a domain prior to continuing to the next wizard page. If the value is set to: - **TRUE**, then the **Active Directory Credential Check** check box is selected in the wizard page editor in the **Domain Join Credentials** section in the UDI Wizard Designer, and credentials are validated If this configuration setting is set to **TRUE**, then the credentials are validated even if the credential fields are disabled (locked). - **FALSE**, then the **Active Directory Credential Check** check box is cleared in the wizard page editor in the **Domain Join Credentials** section in the UDI Wizard Designer, and credentials are not validated If this configuration setting is set to **FALSE**, then the	Yes	No	Yes

Variable	Read	Write	Config
ADComputerObjectCheck configuration setting is ignored and the validation that the provided credentials can modify a computer object in AD DS is not performed.			
UseExistingComputerName Specifies whether the **ComputerPage** wizard page will use the existing computer name on the target computer as the default for the computer name. Note: This check box is only relevant for the Refresh Computer deployment scenario. If the value is set to: - **TRUE**, then the **Use Existing Computer Name** check box is selected in the wizard page editor in the **Computer Name** section in the UDI Wizard Designer, and the existing computer name will be used as the default computer name for the target computer after the new operating system is deployed - **FALSE**, then the **Use Existing Computer Name** check box is cleared in the wizard page editor in the **Computer Name** section in the UDI Wizard Designer, and the existing computer name will not be used as the default computer name for the target computer after the new operating system is deployed	Yes	No	Yes

ConfigScanPage

Use this wizard page to run UDI tasks that scan the configuration of the target computer to determine whether the target computer is ready for the deployment of the operating system image. This readiness includes having sufficient system resources and any prerequisite software being installed and configured properly. In addition, other UDI tasks are run that collect configuration information about the target computer, such as identifying:

- Whether the computer is connected to power (as opposed to running on a battery)
- Whether the computer is connected to a wired network connection (as opposed to using a wireless network connection)
- Any installed applications

- Any installed printers

LanguagePage

Use this wizard page to determine which language packs should be installed, the default language for the target operating system, the keyboard locale, and the time zone in which the computer will be located.

Task Sequence Variables

Table 34 lists the **LanguagePage** task sequence variables with a description and whether the variable is read by the wizard page, written by the wizard page, or can be configured in the UDI Wizard configuration file.

Table 34. LanguagePage Task Sequence Variables

Expand table

Variable	Read	Write	Config
InputLocale Specifies the input locale of the target operating system. You set the value of this variable in the **Time and currency format** box. If not specified, the input locale configured in the image is used.	Yes	Yes	Yes
KeyboardLocale Specifies the keyboard locale of the target operating system. Set the value of this variable in the **Keyboard layout** box. If not specified, the keyboard locale configured in the image is used.	Yes	Yes	Yes
OSDTimeZone Specifies the time zone where the target computer will be physically located. Set the value of this variable in the **Time zone** box. If not specified, the time zone configured in the image is used.	Yes	Yes	Yes
UILanguage Specifies the default language to be used for the target operating system. Set the value of this variable in the **Language to install** box. If not specified, the language configured in the image is used.	Yes	Yes	Yes

ProgressPage

Use this wizard page to run UDI tasks that capture the user state migration data from the target computer. These tasks include:

- Copying the application discovery file to the location selected on the UserStatePage wizard page
- Copying the printer configuration file to the location selected on the UserStatePage wizard page
- Copying the list of installed products to the location selected on the UserStatePage wizard page
- Running the USMT and saving the user state migration data to the location selected on the UserStatePage wizard page

RebootPage

Use this wizard page to notify the user that the target computer is going to be restarted. You can configure the notification message using the UDI Wizard Designer.

SummaryPage

Use this wizard page to notify the user about the configuration options that were selected while running the UDI Wizard. The configuration information displayed on this wizard page is automatically collected from other wizard pages. Some fields on other wizard pages allow you to configure the caption (label) displayed on this wizard page using the UDI Wizard Designer.

UDAPage

Use this wizard page to configure the UDA between the target computer and a specified user. Assigning a user as the primary user of a computer allows automatic installation of software that is deployed to that user. The UDA feature is only available in Configuration Manager and only in the New Computer deployment scenario.

Task Sequence Variables

Table 35 lists the **UDAPage** task sequence variables with a description and whether the variable is read by the wizard page, written by the wizard page, or can be configured in the UDI Wizard configuration file.

Table 35. UDAPage Task Sequence Variables

Expand table

Variable	Read	Write	Config
SMSTSAssignUsersMode Specifies the mode for configuring user affinity in Configuration Manager. Use this variable to configure the behavior of creating affinity between the target computer and user accounts in the **SMSTSUdaUsers** task sequence variable. To set this variable, select the **Use User Device Affinity** check box. If the variable is set to: - **Auto**, then the affinity processing is automatically approved by Configuration Manager - **Pending**, then the affinity processing rules will require approval by a Configuration Manager administrator (This is the value used when the **Use User Device Affinity** check box is selected.) - **Disabled**, then no affinity processing will occur	No	Yes	No
SMSTSUdaUsers Specifies the users to be associated with the target computer. The **User Device Affinity Account** sets this variable. This variable can have one or many users specified and is in the format `Domain\User1, Domain\User2`.	Yes	Yes	Yes

UserStatePage

Use this wizard page to configure the settings for capturing or restoring user state migration data. This wizard page is used for both user state migration data capture and restore.

The **UserStatePage** can capture or restore user state migration data from a disk locally attached to the target computer, a USB drive attached to the target computer, or a network shared folder. In addition, you can select to not restore any user data. The code logic behind the wizard page enables, disables, or automatically selects each of the following options based on the deployment scenario and whether the disk is being formatted:

- **No Data to Restore.** This option indicates that there is no user state migration data to restore and sets the **OSDUserStateMode** task sequence variable and

UserStateMode variable to **NoData**.

- **Local.** This option indicates that the user state migration data should be stored on a disk locally attached to the target computer and sets the **OSDUserStateMode** task sequence variable and **UserStateMode** variable to **Local**.
- **USB.** This option indicates that the user state migration data should be stored on a USB disk locally attached to the target computer and sets the **OSDUserStateMode** task sequence variable and **UserStateMode** variable to **USB**.
- **Network.** This option indicates that the user state migration data should be stored on a network shared folder and sets the **OSDUserStateMode** task sequence variable and **UserStateMode** variable to **Network**.

NEWCOMPUTER Stage Behavior

The NEWCOMPUTER stage is used for computers on which no user state migration data exists. The New Computer deployment scenario can be used as the second part of the Replace Computer deployment scenario. If the user selects to:

- Format the disk on the target computer, then the **UserStatePage** assumes that no user state migration data is located on the local hard disk, so the **Local** option is disabled and all other options are enabled
- Not format the disk on the target computer, then the **UserStatePage** assumes that there is user state migration data to be restored, and all options are disabled other than the **Local** option (Using the **Local** option provides a faster method for restoring the user state migration data than the USB or network shared folder methods.)

  Table 36 lists the behavior of the options on the wizard page for the NEWCOMPUTER stage. The **Format** column indicates whether the target hard disk is to be formatted as a part of the deployment. The other columns indicate the configuration of the options when the **UserStatePage** is loaded.

Table 36. Behavior of Options for the NEWCOMPUTER Stage

Expand table

Format	NoData	Local	USB	Network
Yes	Enabled	Disabled	Enabled	Enabled
No	Disabled	Selected	Disabled	Disabled

NewComputer.Prestaged Stage Behavior

The NEWCOMPUTER.Prestaged stage is based on the prestaged media feature in Configuration Manager. Because the local hard disk is new, there is no user state migration data to be restored from the local hard disk, so the **Local** option is disabled. All other options are valid for this deployment scenario and are enabled. No default option is selected.

Table 37 lists the behavior of the options on the wizard page for the NewComputer.Prestaged stage. The **Format** column indicates whether the target hard disk is to be formatted as a part of the deployment. The other columns indicate the configuration of the options when the **UserStatePage** is loaded.

Table 37. Behavior of Options for the NewComputer.Prestaged Stage

Expand table

Format	NoData	Local	USB	Network
N/A	Enabled	Disabled	Enabled	Enabled

REFRESH Stage Behavior

The REFRESH stage is initiated in a full Windows operating system, instead of Windows PE. If the user selects to:

- Format the disk on the target computer, then the **UserStatePage** assumes that no user state migration data is to be restored, and all options are disabled other than the **NoData** option
- Not format the disk on the target computer, then the **UserStatePage** assumes that there is user state migration data to be restored, and all options are disabled other than the **Local** option (Using the **Local** option provides a faster method for restoring the user state migration data than the USB or network shared folder methods.)

Table 38 lists the behavior of the options on the wizard page for the REFRESH stage. The **Format** column indicates whether the target hard disk is to be formatted as a part of the deployment. The other columns indicate the configuration of the options when the **UserStatePage** is loaded.

Table 38. Behavior of Options for the REFRESH Stage

Expand table

Format	NoData	Local	USB	Network
Yes	Selected	Disabled	Disabled	Disabled
No	Disabled	Selected	Disabled	Disabled

REPLACE.WinPE Stage Behavior

The REPLACE.WinPE stage captures the user state migration data from the existing (old) computer, and then restores the user state migration data later using one of the New Computer deployment scenarios. Because two different computers are involved in the deployment, the user state migration data must be saved to a USB drive or to a network shared folder. Saving user state migration data to a local disk is unavailable.

Table 39 lists the behavior of the options on the wizard page for the REPLACE.WinPE stage. The **Format** column indicates whether the target hard disk is to be formatted as a part of the deployment. The other columns indicate the configuration of the options when the **UserStatePage** is loaded.

Table 39. Behavior of Options for the REPLACE.WinPE Stage

Expand table

Format	NoData	Local	USB	Network
N/A	Disabled	Disabled	Enabled	Enabled

Task Sequence Variables

Table 40 lists the **UserStatePage** task sequence variables with a description and whether the variable is read by the wizard page, written by the wizard page, or can be configured in the UDI Wizard configuration file.

Table 40. UserStatePage Task Sequence Variables

Expand table

Variable	Read	Write	Config
_SMSTsInWinPE Specifies whether the UDI Wizard is running in Windows PE. If the variable is set to: - **TRUE**, then the UDI Wizard is running in Windows PE - **FALSE**, then the UDI Wizard is not running in Windows PE, but rather in a full Windows operating system	Yes	No	No
OSDDataSourceDirectory Specifies the directory in which the user state migration data is stored.	No	Yes	No
OSDDataSourceDrive Specifies the USB drive used for capturing and restoring user state migration data, which you select from the **USB Target Drive** box. If the variable is set prior to showing the wizard page, the value of the variable is used as the default value.	Yes	Yes	No
OSDDiskPart Specifies whether the drive selected for the installation of the target operating system should be formatted and partitioned. You set this variable on the VolumePage wizard page, and the code on this wizard page uses it to determine which options are selected and enabled by default. For more information, see UserStatePage.	Yes	No	Yes
OSDHardLinks Specifies whether the user state migration data is to be captured to or restored from a local drive. If the variable is set to: - **TRUE**, then the **Local** option was selected, and user state migration data will be captured or restored from a local drive that is attached to	No	Yes	No

Variable	Read	Write	Config
the target computer - **FALSE**, then the **Local** option was not selected, and no user state migration data will be captured or restored from a local drive that is attached to the target computer			
OSDRestoreData Specifies whether there is data to be restored. If the variable is set to: - **TRUE**, then the **Local**, **USB Target Drive**, or **Network** option was selected, and user state migration data will be captured or restored from the target computer - **FALSE**, then the **No Data to Restore** option was selected, and no user state migration data will be captured or restored from the target computer	No	Yes	No
OSDUserStateKey Specifies the user name used to secure the user state migration data. The user name is provided when the user state migration data is captured. The same user name and password must be provided when the user state migration data is restored. You set the value of this variable in the **User name** box.	Yes	Yes	Yes
OSDUserStateKeyPassword Specifies the password for the user name used to secure the user state migration data. Set the value of this variable in the **Password** and **Confirm password** boxes.	Yes	Yes	Yes
OSDUserStateMode Specifies the mode (method) for capturing or restoring the user state migration data. The value of this variable is set by the options selected. If the variable is set to: - **NoData**, then the **No Data to Restore** option was selected, and no user state migration data will be captured or restored - **Local**, then the **Local** option was selected, and the user state migration data will be captured or restored from a local hard disk on the target computer - **Network**, then the **Network** option was selected, and the user state **migration** data will be captured to or restored from a network shared folder	No	Yes	No

Variable	Read	Write	Config
- When used in capture mode, this option creates a folder based on a hash of the user name and password so that the identity of the user state migration data is protected. The exact same user name and password must be used when restoring the user state migration data so that the wizard page can accurately locate the folder. - **USB**, then the **USB Target Drive** option was selected, and the user state migration data will be captured to or restored from a USB drive that is physically attached to the target computer - The wizard page behavior for USB drives is also affected by the **Format**, **FormatPrompt**, and **MinimumDriveSize** variables.			
SMSConnectNetworkFolderPath Specifies the network shared folder used for capturing and restoring user state migration data, which is selected from the **Network** box. The **Network** box displays a user-friendly name for the network shared folder that is configured in the **Network Shares** box in the **Network Combo Box** section on the wizard page editor in the UDI Wizard Designer. If the variable is set prior to showing the wizard page, the value of the variable is used as the default value.	Yes	Yes	Yes

Memory Variables

Table 41 lists the **UserStatePage** memory variables with a description and whether the variable is read or written by the wizard page.

Table 41. UserStatePage Memory Variables

Expand table

Variable	Read	Write
DriveLetter Specifies the drive letter for the USB drive selected in the **USB Target Drive** box on the wizard page. The value of this variable will be the drive letter, including the colon (:) suffix, such as *M:*.	No	Yes
TargetDrive Specifies the caption displayed in the **USB Target Drive** box on the wizard page for the USB drive selected on the target computer. The value of this variable will	No	Yes

Variable	Read	Write
be similar to the following example: `M: VendorA Ultra TD v1.0 USB Device (74.5 GB)`		
UserStateMode Specifies the option selected with the options on the wizard page and is set to the same value as the **OSDUserStateMode** variable. Valid values for this variable include: - **NoData**, which indicates that the **No Data to Restore** option was selected - **Local**, which indicates that the **Local** option was selected - **USB**, which indicates that the **USB Target Drive** option was selected - **Network**, which indicates that the **Network** option was selected	No	Yes

Configuration Variables

Table 42 lists the **UserStatePage** configuration variables with a description and whether the variable is read by the wizard page, written by the wizard page, or can be configured in the UDI Wizard configuration file.

Table 42. UserStatePage Configuration Variables

Expand table

Variable	Read	Write	Config
DataSourceText Specifies an informational message that instructs the user performing the user state capture or restore about how to use the wizard page. You set the value of this variable in the **Instruction Text** box in the **Message** section on the wizard paged editor in the UDI Wizard Designer.	Yes	No	Yes
Format Specifies whether the USB drive selected for capturing user state on the target computer should be partitioned and formatted prior to capturing user state migration data. Set the value of this variable by selecting the **Format the USB drive before capture** check box in the **USB Combo Box**	Yes	No	Yes

Variable	Read	Write	Config
section on the wizard paged editor in the UDI Wizard Designer. If the variable is set to: - **TRUE**, then the drive is formatted prior to capturing user state migration data - **FALSE**, then the drive is not formatted prior to capturing user state migration data			
FormatPrompt Specifies whether the user must confirm that the USB drive used for capturing user state migration data is to be formatted prior to performing the capture. Set the value of this variable by selecting the **Prompt the user before formatting the target drive** check box in the **USB Combo Box** section on the wizard paged editor in the UDI Wizard Designer. Note: This variable is only valid if the **OSDUserStateMode** task sequence variable is set to **USB**.	Yes	No	Yes
MinimumDriveSize Specifies the minimum available free disk space in gigabytes required for a drive to be available for storing user state migration data. The value of this variable acts as a filter, and you set it in the **Minimum Drive Size** text box in the **USB Combo Box** section on the wizard paged editor in the UDI Wizard Designer.	Yes	No	Yes
NetworkDrive Specifies the drive letter that this wizard page uses to map to the network shared folder in the **SMSConnectNetworkFolderPath** task sequence variable. The network shared folder mapping is used for capturing or restoring the user state migration data. Set the value of this variable in the **Mapped Drive Letter** box in the **Network Combo Box** section on the wizard paged editor in the UDI Wizard Designer. The drive letter specified must include the colon (:) after the drive letter and must not be in use on the target computer. For example, if the target computer has drives C: and D:, then C: and D: could not be used for this variable. Note:	Yes	No	Yes

Variable	Read	Write	Config
This variable is only valid if the **OSDUserStateMode** task sequence variable is set to **Network**.			
State Specifies whether the wizard page is being used for capturing or restoring the user state migration data. Set the value of this variable in the **Capture or Restore** box in the **Capture/Restore Location** section on the wizard paged editor in the UDI Wizard Designer. If the variable is set to: - **Capture**, then the wizard page is used to capture user state migration data - **Restore**, then the wizard page is used to restore user state migration data	Yes	No	Yes

VolumePage

Use this wizard page to configure the settings for the disk volume on the target computer on which the operating system will be deployed. These settings include selecting the target operating system, selecting the target drive, selecting any Windows installation, and determining whether the target drive should be formatted as a part of the deployment process.

Task Sequence Variables

Table 43 lists the **VolumePage** task sequence variables with a description and whether the variable is read by the wizard page, written by the wizard page, or can be configured in the UDI Wizard configuration file.

Table 43. VolumePage Task Sequence Variables

Expand table

Variable	Read	Write	Config
OSDDiskPart Specifies whether the drive selected for deploying the target operating system on the target computer should be partitioned and formatted prior to capturing user state migration data. The value of this variable is set by the one of the following check boxes on the wizard page:	Yes	Yes	Yes

Variable	Read	Write	Config
- **Clean the selected volume.** This check box appears when the UDI Wizard is running in a full Windows operating system. You can configure the text message using the **FormatFullOS** setter property for the wizard page in the UDI Wizard configuration file. - **Partition and format disk 0.** This check box appears when the UDI Wizard is running in Windows PE. You can configure the text message using the **FormatWinPE** setter property for the wizard page in the UDI Wizard configuration file. The code logic behind the UserStatePage wizard page uses this variable to determine which options are selected and enabled by default. If the variable is set to: - **TRUE**, then the drive is partitioned and formatted prior to deploying the target operating system - **FALSE**, then the drive is not partitioned and formatted prior to deploying the target operating system			
OSDImageIndex Specifies a numeric index of the operating system image in the .wim file, which is selected in the **Image Selection** combo box. You configure the list of possible operating system images in the **Image Selection** box in the **Image Combo Box Values** list in the **Image Combo Box** section on the **VolumePage** wizard page editor. The image index is configured as a part of each image in the **Image Combo Box Values** list.	Yes	Yes	Yes
OSDImageName Specifies the name of the operating system image in the .wim file, which is selected in the **Image Selection** box. The list of possible operating system images in the **Image Selection** combo box is configured in the **Image Combo Box Values** list in the **Image Combo Box** section on the **VolumePage** wizard page editor. The image name is configured as a part of each image in the **Image Combo Box Values** list.	No	Yes	No
OSDTargetDrive Specifies the drive letter for the volume selected in the **Volume** box on the wizard page. The value of this variable will be the drive letter, including the colon (:) suffix, such as *C:*.	No	Yes	No
OSDWinPEWindir	No	Yes	No

Variable	Read	Write	Config
Specifies the location of an existing installation of Windows on the target computer. Set the value of this variable in the **Windows Directory** box on the wizard page.			

Memory Variables

Table 44 lists the **VolumePage** memory variables with a description and whether the variable is read or written by the wizard page.

Table 44. VolumePage Memory Variables

Expand table

Variable	Read	Write
VolumeArchitecture Specifies the processor architecture of the operating system to be deployed, which is selected in the **Image Selection** box. The **VolumeArchitecture** wizard page consumes this variable to filter the architecture of applications displayed on that page. For example, if a 32-bit operating system is to be deployed, then the **VolumeArchitecture** wizard page removes (filters) any 64-bit applications from the list of available applications. If the variable is set to: - **x86**, then a 32-bit operating system was selected - **amd64**, then 64-bit operating system was selected	No	Yes

WelcomePage

Use this wizard page to provide information to the user about the UDI Wizard and the deployment process. You can configure the notification message using the UDI Wizard Designer.

UDI Build Your Own Page Toolbox Control Reference

The Build Your Own Page feature in UDI allows you to create custom wizard pages that you can use to collect additional deployment information for use in UDI. You can create custom wizard pages using the:

- **Build Your Own Page feature.** This feature allows you to create a custom wizard page for collecting deployment information without requiring you to write code or have developer skills. Use this feature if you need to collect basic information without advanced user interaction. For example, you cannot add any code or customize UI fonts using this feature.

- **UDI SDK and Visual Studio**. Use this SDK if you want to create an advanced, fully customized wizard page in Visual Studio for collecting deployment information. Although the UDI SDK allows you to create customized wizard pages, such as adding custom code or changing fonts, this method requires developer skills.

  For more information on using the UDI SDK to create custom wizard pages, see "Creating Custom UDI Wizard Pages" in the *User-Drive Installation Developers Guide*.

  The Build Your Own Page feature includes a toolbox of controls that you can add to your custom wizard page from the Build Your Own Page toolbox, which is displayed when you view the custom wizard page on the **Configure** tab in the UDI Wizard Designer.

  Table 45 lists the types of controls to your custom wizard page, which is illustrated in Figure 5. Each of these controls is discussed in further detail in a subordinate section.

Table 45. Types of Controls in the UDI Build Your Own Page Toolbox

Expand table

Control type	Description
Checkbox control	This control allows you select or clear a configuration option and behaves as a traditional UI check box.
Combobox control	This control allows you to select an item from a list of items and behaves as a traditional UI drop-down list.
Line control	This control allows you to add a horizontal line to divide one portion of the custom wizard page from another.

Control type	Description
Label control	This control allows you to add descriptive, read-only text to the wizard page.
Radio control	This control allows you to select one configuration option from a group of two or more options.
Bitmap control	This control allows you to add a bitmap graphic (.bmp file) to the custom wizard page.
Textbox control	This control allows you to enter text on the custom wizard page.

You can add any combination of these controls to your custom wizard page based on the information you want to collect. In addition, you can use the **Show Gridlines** check box to show or hide gridlines that can be used to assist in visually designing the custom wizard page.

Figure 5 provides an example of a custom wizard page and the Build Your Own Page toolbox.

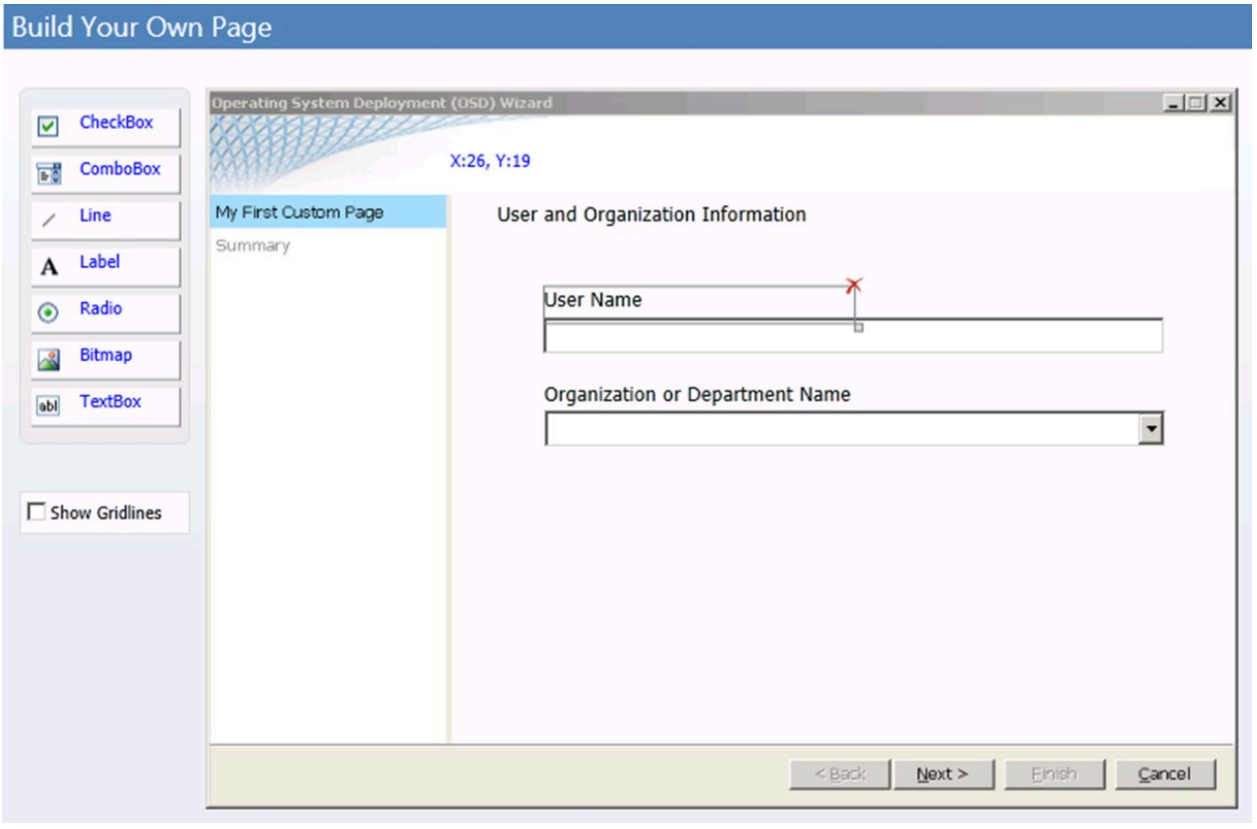


Figure SEQ Figure * ARABIC 5. Example custom wizard page

Checkbox Control

This control allows you select or clear a configuration option and behaves as a traditional UI check box. This control has a corresponding label that you can use to describe the purpose of the check box. The state of this control is True when the check

box is selected and False when the check box is cleared. The state of the check box is stored in the task sequence variable configured for this control.

Layout Properties

Layout properties are used to configure the UI characteristics of the control and are configured on the **Layout** tab in the UDI Wizard Designer. Table 46 lists the layout properties for the **Checkbox** control and provides a brief description of each property

Table 46. Checkbox Control Layout Properties

Expand table

Property	Description
X	Use this property to configure the horizontal position of the control.
Y	Use this property to configure the vertical position of the control.
Label	Use this property to configure the descriptive text associated with the check box.
Width	Use this property to configure the width of the control. **Note** If the text entered in the **Label** property is longer than the width of the control, the text is clipped and not displayed.
Height	Use this property to configure the height of the control. **Note** If the text entered in the **Label** property is taller than the height of the control, the text is clipped.

Settings Properties

Settings properties are used to configure the data initially shown in a control (the default value) and where the information collected from the user is saved. Table 47 lists the settings properties for the **Checkbox** control and provides a brief description of each property.

Table 47. Checkbox Control Settings Properties

Expand table

Property	Description
Default value	Use this property to configure the default value for the control. For a check box, the default value is **False**.
Task sequence variable name	Use this property to configure the task sequence variable where the information collected from the user is stored. If the task sequence variable: - Does not already exist, the task sequence variable is created and set to the value the user provides - Already exists, the existing value of the task sequence variable is overwritten with the value the user provides
Friendly display name visible in the summary page	Use this property to configure the descriptive name that appears on the **Summary** wizard page. This name is used to describe the value that was saved in the **Task sequence variable name** property for this control.
Unlocked	Use this property to configure whether the user is able to interact with the control. By default, the control is enabled. This button displays the following status: - **Unlocked.** The control is enabled, and users can enter information using it. - **Locked.** The control is disabled, and users are unable to enter information using it. **Note** If you disable (lock) a control, you must provide the information the control collected by configuring MDT properties in CustomSettings.ini or in the MDT DB. Otherwise, the UDI Wizard will not collect the necessary information, and the UDI deployment will fail.

Combobox Control

This control allows you to select an item from a list of items and behaves as a traditional UI drop-down list. This control allows you to add or remove items from the list and provide a corresponding value that will be set in the task sequence variable configured for this control.

Layout Properties

Layout properties are used to configure the UI characteristics of the control and are configured on the **Layout** tab in the UDI Wizard Designer. Table 48 lists the layout properties for the **Combobox** control and provides a brief description of each property.

Table 48. Combobox Control Layout Properties

Expand table

Property	Description
X	Use this property to configure the horizontal position of the control.
Y	Use this property to configure the vertical position of the control.
Width	Use this property to configure the width of the control. **Note** If the text entered in the control is longer than the width of the control, the text is not displayed.
Height	Use this property to configure the height of the control. **Note** If the text entered in the control is taller than the height of the control, the text is clipped.
Data Items	Use this property to configure the list of data items displayed in the control. Each data item has the following properties: - **Value.** The value stored in the task sequence variable when the data item is selected - **DisplayValue.** The value displayed to the user in the control You can: - Add data items to the list using the blue plus sign button immediately to the right of the list of data items - Remove data items from the list using the red **X** button immediately to the right of the list of data items **Note** You cannot change the sequence of the data item in the list after an item is added to the list. Ensure that you enter the data items in the order you wish them to appear in the control.

Settings Properties

Settings properties are used to configure the data that is initially shown in a control (the default value) and where the information collected from the user is saved. Table 49 lists the settings properties for the **Combobox** control and provides a brief description of each property.

Table 49. Combobox Control Settings Properties

Expand table

Property	Description
Task sequence variable name	Use this property to configure the task sequence variable where the information collected from the user is stored. If the task sequence variable: - Does not already exist, the task sequence variable is created and set to the value the user provides - Already exists, the existing value of the task sequence variable is overwritten with the value the user provides
Friendly display name visible in the summary page	Use this property to configure the descriptive name that appears on the **Summary** wizard page. This name is used to describe the value that was saved in the **Task sequence variable name** property for this control.
Unlocked	Use this property to configure whether the user is able to interact with the control. By default, the control is enabled. This button displays the following status: - **Unlocked.** The control is enabled, and users can enter information using it. - **Locked.** The control is disabled, and users are unable to enter information using it. **Note** If you disable (lock) a control, you must provide the information the control collected by configuring MDT properties in CustomSettings.ini or in the MDT DB. Otherwise, the UDI Wizard will not collect the necessary information, and the UDI deployment will fail.

Line Control

This control allows you to add a horizontal line to divide one portion of the custom wizard page from another. This control does not collect any configuration values but rather is used to visually enhance the UI.

Layout Properties

Layout properties are used to configure the UI characteristics of the control and are configured on the **Layout** tab in the UDI Wizard Designer. Table 50 lists the layout properties for the **Line** control and provides a brief description of each property.

Table 50. Line Control Layout Properties

Expand table

Property	Description
X	Use this property to configure the horizontal position of the control.
Y	Use this property to configure the vertical position of the control.
Width	Use this property to configure the width of the control.
Height	Use this property to configure the height of the control. **Note** Increasing this property does not increase the height or width of the line.

Settings Properties

The **Line** control has no settings properties.

Label Control

This control allows you to add descriptive, read-only text to the wizard page. This control does not collect any configuration values but rather is used to visually enhance the UI.

Layout Properties

Layout properties are used to configure the UI characteristics of the control and are configured on the **Layout** tab in the UDI Wizard Designer. Table 51 lists the layout properties for the **Label** control and provides a brief description of each property.

Table 51. Label Control Layout Properties

Expand table

Property	Description
X	Use this property to configure the horizontal position of the control.
Y	Use this property to configure the vertical position of the control.
Label	Use this property to configure the descriptive text associated with this control.

Property	Description
Width	Use this property to configure the width of the control. **Note** If the text entered in the **Label** property is longer than the width of the control, the text is clipped and not displayed.
Height	Use this property to configure the height of the control. **Note** If the text entered in the **Label** property is taller than the height of the control, the text is clipped.

Settings Properties

The **Label** control has no settings properties.

Radio Control

This control allows you to select one option from a group of two or more options. As with traditional radio buttons, you can group two or more of these controls; then, the user can select one of the options in the group.

A unique value is assigned to each radio button. The value assigned to the selected radio button control is saved in the task sequence variable configured for this control.

Layout Properties

Layout properties are used to configure the UI characteristics of the control and are configured on the **Layout** tab in the UDI Wizard Designer. Table 52 lists the layout properties for the **Radio** control and provides a brief description of each property.

Table 52. Radio Control Layout Properties

Expand table

Property	Description
X	Use this property to configure the horizontal position of the control.
Y	Use this property to configure the vertical position of the control.
Label	Use this property to configure the descriptive text associated with the radio button.

Property	Description
Width	Use this property to configure the width of the control. **Note** If the text entered in the **Label** property is longer than the width of the control, the text is clipped and not displayed.
Height	Use this property to configure the height of the control. **Note** If the text entered in the **Label** property is taller than the height of the control, the text is clipped.
RadioGroup	Use this property to group two or more radio buttons. When radio buttons belong to the same group, only one of the radio buttons within a group can be selected. If you need multiple groups of radio buttons, configure this property for each respective group of radio buttons.
Value	Use this property to configure the value stored in the task sequence variable when the radio button is selected.

Settings Properties

Settings properties are used to configure the data initially shown in a control (the default value) and where the information collected from the user is saved. Table 53 lists the settings properties for the **Radio** control and provides a brief description of each property.

Table 53. Radio Control Settings Properties

Expand table

Property	Description
Default value	Use this property to configure the default value for the control. By default, the value is set to the control ID.
Task sequence variable name	Use this property to configure the task sequence variable where the information collected from the user is stored. If the task sequence variable: - Does not already exist, the task sequence variable is created and set to the value the user provides - Already exists, the existing value of the task sequence variable is overwritten with the value the user provides

Property	Description
Friendly display name visible in the summary page	Use this property to configure the descriptive name that appears on the **Summary** wizard page. This name is used to describe the value that was saved in the **Task sequence variable name** property for this control.
Unlocked	Use this property to configure whether the user is able to interact with the control. By default, the control is enabled. This button displays the following status: - **Unlocked.** The control is enabled, and users can enter information using it. - **Locked.** The control is disabled, and users are unable to enter information using it. **Note** If you disable (lock) a control, you must provide the information the control collected by configuring MDT properties in CustomSettings.ini or in the MDT DB. Otherwise, the UDI Wizard will not collect the necessary information, and the UDI deployment will fail.

Bitmap Control

This control allows you to add a bitmap graphic (.bmp file) to the custom wizard page. This control does not collect any configuration values but rather is used to visually enhance the UI.

Layout Properties

Layout properties are used to configure the UI characteristics of the control and are configured on the **Layout** tab in the UDI Wizard Designer. Table 54 lists the layout properties for the **Bitmap** control and provides a brief description of each property.

Table 54. Bitmap Control Layout Properties

Expand table

Property	Description
X	Use this property to configure the horizontal position of the control.
Y	Use this property to configure the vertical position of the control.
Width	Use this property to configure the width of the control.

Property	Description
	Note If the graphic selected in the **Source** property is longer than the width of the control, the graphic is clipped.
Height	Use this property to configure the height of the control. **Note** If the graphic selected in the **Source** property is taller than the height of the control, the graphic is clipped.
Source	Use this property to configure the fully qualified path to the .bmp file, including the file name. The path to the .bmp file is relative to the location of the UDI Wizard (OSDSetupWizard.exe), which is on one of the following folders (where *mdt_tookit_package* is the location of the MDT toolkit package in Configuration Manager): - *mdt_tookit_package*\Tools\x86 - *mdt_tookit_package*\Tools\x64 To view the graphic when previewing the custom wizard page, the .bmp file must also be located in the following folders (where *mdt_install_folder* is the folder where you installed MDT): - *mdt_install_folder*\Template\Distribution\Tools\x86 - *mdt_install_folder* \Template\Distribution\Tools\x64

Settings Properties

The **Bitmap** control has no settings properties.

Textbox Control

This control allows you to enter text on the custom wizard page. The text typed into this control is saved in the task sequence variable configured for this control.

Layout Properties

Layout properties are used to configure the UI characteristics of the control and are configured on the **Layout** tab in the UDI Wizard Designer. Table 55 lists the layout properties for the **Textbox** control and provides a brief description of each property.

Table 55. Textbox Control Layout Properties

Expand table

Property	Description
X	Use this property to configure the horizontal position of the control.
Y	Use this property to configure the vertical position of the control.
Width	Use this property to configure the width of the control. **Note** If the text entered in the control is longer than the width of the control, the text is clipped and not displayed.
Height	Use this property to configure the height of the control. **Note** If the text entered in the control is taller than the height of the control, the text is clipped.

Settings Properties

Settings properties are used to configure the data that is initially shown in a control (the default value) and where the information collected from the user is saved. Table 56 lists the settings properties for the **Textbox** control and provides a brief description of each property

Table 56. Textbox Control Settings Properties

Expand table

Property	Description
Default value	Use this property to configure the default value for the control.
Task sequence variable name	Use this property to configure the task sequence variable where the information collected from the user is stored. If the task sequence variable: - Does not already exist, the task sequence variable is created and set to the value the user provides - Already exists, the existing value of the task sequence variable is overwritten with the value the user provides
Friendly display name visible in the summary page	Use this property to configure the descriptive name that appears on the **Summary** wizard page. This name is used to describe the value that was saved in the **Task sequence variable name** property for this control.

Property	Description
List of validators assigned to this control	This property contains a list of validators used to verify that the content entered in the text box. You can: - Add validators to the list using the blue plus sign button immediately to the right of the list of validators - Edit validators in the list using the pencil button immediately to the right of the list of validators - Remove validators from the list using the red **X** button immediately to the right of the list of validators
Unlocked	Use this property to configure whether the user is able to interact with the control. By default, the control is enabled. This button displays the following status: - **Unlocked.** The control is enabled, and users can enter information using it. - **Locked.** The control is disabled, and users are unable to enter information using it. Note: **Note** If you disable (lock) a control, you must provide the information the control collected by configuring MDT properties in CustomSettings.ini or in the MDT DB. Otherwise, the UDI Wizard will not collect the necessary information, and the UDI deployment will fail.

UDI Task Sequence Variables

The task sequence variables in this section are used only in User-Driven Installation (UDI) deployments. In addition to these task sequence variables, the following ZTI task sequence variables are also used by UDI and are documented in their respective sections earlier in this guide:

- KeyboardLocale
- OSDComputerName
- UILanguage
- UserLocale

OSDAddAdmin

This task sequence variable specifies a list of domain-based accounts or local accounts to be added to the Administrators local built-in group on the target computer.

Expand table

Value	Description
domain\account_name1; computer\account_name2	The format of the accounts to be made members of the Administrators group on the target computer in the format of *domain\account* and separated by semicolons, where *domain* can be the name of an Active Directory domain or the target computer name.

Expand table

Example
`OSDAddAdmin=domain\user01;Win7-01\LocalUser01`

OSDApplicationList

This task sequence variable specifies which applications should be selected by default on the **Install Software** page of the Operating System Deployment (OSD) Setup Wizard.

Expand table

Value	Description
app_id1;app_id2	A semicolon-delimited list of application to be selected by default on the **Install Software** page of the Operating System Deployment (OSD) Setup Wizard; each application is represented by an application ID and separated by a semicolon. The application ID is derived from the **Id** attribute of each application in the UDIWizard_Config.xml file. In the following excerpt from a UDIWizard_Config.xml file, the 2007 Microsoft Office system with SP2 application has an **Id** attribute of **1**: `<Application DisplayName="Office 2007 SP2" State="Disabled" Id="1">`

Expand table

Example
`OSDApplicationList=2;3`

OSDArchitecture

This task sequence variable specifies the processor architecture of the target operating system to be deployed.

Expand table

Value	Description
x86	The target operating system is a 32-bit operating system.
amd64	The target operating system is a 64-bit operating system.

Expand table

Example
`OSDArchitecture=amd64`

OSDBitlockerStatus

This task sequence variable specifies if BitLocker is enabled on the target computer by the BitLocker preflight check.

Expand table

Value	Description
PROTECTED	The target computer has BitLocker enabled.
Does not exist	If the target computer does not have BitLocker enabled, then the task sequence variable does not exist.

Expand table

Example
None

OSDDiskPart

This task sequence variable specifies whether the target disk partition should be formatted.

Expand table

Value	Description
TRUE	The target disk partition will be formatted.
FALSE	The target disk partition will not be formatted.

Expand table

Example
`OSDDiskPart=TRUE`

OSDDomainName

This task sequence variable specifies the name of the domain to which the target computer will be joined if the computer is configured to be a domain member.

Expand table

Value	Description
domain_name	The name of the domain to which the target computer will be joined. If you have configured the **Computer** wizard page in the Operating System Deployment (OSD) Setup Wizard to be **Silent**, the value in this task sequence variable must match the values specified in the UDI Wizard Designer. Otherwise, the wizard page will be displayed. Note: This task sequence variable is only necessary when you are creating a new computer account in the OU. If the computer account already exists, this variable is not needed.

Expand table

Example
`OSDDomainName=domain01`

OSDDomainOUName

This task sequence variable specifies the name of the OU in the domain to which the target computer account will be created when the computer joins a domain.

Expand table

Value	Description
ou_name	The name of the OU in the domain in which the computer account will be created Note: This task sequence variable is only necessary when you are creating a new computer account in the OU. If the computer account already exists, this variable is not needed.

Expand table

Example
`OSDDomainOUName=NewDeployOU`

OSDImageIndex

This task sequence variable specifies the index number of the target operating system in a WIM file.

Expand table

Value	Description
index_number	The index number of the target, which is starts with an index number of 1 for the first operating system in the WIM file

Expand table

Example
`OSDImageIndex=1`

OSDImageName

This task sequence variable specifies the name of the operating system image in the .wim file selected in the **Image Selection** box on the VolumePage wizard page. The list of possible operating system images in the **Image Selection** box is configured in the **Image Combo Box Values** list in the **Image Combo Box** section on the VolumePage wizard page editor. The image name is configured as a part of each image in the **Image Combo Box Values** list.

> **Note**
>
> **Note** This tasks sequence variable is set by the VolumePage wizard and should not be configured in the CustomSettings.ini file or in the MDT DB. However, this tasks sequence variable can be used to set conditions for task sequence steps, as described in the section, "Configure UDI Task Sequences to Deploy Different Operating Systems", in the MDT document *Using the Microsoft Deployment Toolkit*.

Expand table

Value	Description
image_name	The name of the operating system image in the .wim file selected in the **Image Selection** box on the VolumePage wizard page

Expand table

Example
None

OSDJoinAccount

This task sequence variable specifies the domain-based account used to join the target computer to the domain specified in the **OSDDomainName** task sequence variable. This task sequence variable is necessary if the target computer will be joined to a domain.

Expand table

Value	Description
account_name	The name of the account used to join the target computer to the domain in the format of *domain**account*

Expand table

Example
`OSDJoinAccount=domain\admin01`

OSDJoinPassword

This task sequence variable specifies the password for the domain-based account used to join the target computer to the domain specified in the **OSDJoinAccount** task sequence variable. This task sequence variable is necessary if the target computer will be joined to a domain.

Expand table

Value	Description
password	The password of the account used to join the domain

Expand table

Example
`OSDJoinPassword=P@ssw0rd10`

OSDLocalAdminPassword

This task sequence variable specifies the password for the Administrator local built-in account on the target computer.

Expand table

Value	Description
password	The password of the Administrator local built-in account on the target computer

Expand table

Example
`OSDLocalAdminPassword=P@ssw0rd10`

OSDNetworkJoinType

This task sequence variable specifies whether the target computer joins a domain or a workgroup.

Expand table

Value	Description
0	The target computer will join a domain. If you select this option and configure the corresponding Operating System Deployment (OSD) Setup Wizard page to be **Silent**, you must also provide values for the **OSDJoinAccount**, **OSDJoinPassword**, **OSDDomainName**, and **OSDDomainOUName** task sequence variables accordingly. In addition, you must select **Domain** in **Default Selection** in the Workspace pane on the **Computer Page** in the UDI Wizard Designer.
1	The target computer will join a workgroup. If you select this option and configure the corresponding Operating System Deployment (OSD) Setup Wizard page to be **Silent**, you must also provide a value for the **OSDWorkgroupName** task sequence variable. In addition, you must select **Workgroup** in **Default Selection** in the Workspace pane on the **Computer Page** in the UDI Wizard Designer.

Expand table

Example
`OSDNetworkJoinType=0`

OSDSetupWizCancelled

This task sequence variable specifies if the user canceled the Operating System Deployment (OSD) Setup Wizard.

Expand table

Value	Description
TRUE	The user canceled the Operating System Deployment (OSD) Setup Wizard.
Does not exist	If the wizard is not canceled, then the task sequence variable does not exist.

Expand table

Example
None

OSDTargetDrive

This task sequence variable specifies the disk volume where the target operating system will be deployed.

Expand table

Value	Description
disk_volume	The disk volume designation

Expand table

Example
`OSDTargetDrive=C:`

OSDWinPEWinDir

This task sequence variable specifies the folder in which the Windows operating system is currently installed on the target computer.

Expand table

Value	Description
windows_directory	The directory in which the Windows operating system is currently installed

Expand table

Example
`OSDWinPEWinDir=C:\Windows`

OSDWorkgroupName

This task sequence variable specifies the name of the workgroup to which the target computer will be joined if the computer is configured to be a workgroup member.

Expand table

Value	Description
workgroup_name	The name of the workgroup to which the target computer will be joined

Expand table

Example
`OSDWorkgroupName=WORKGROUP01`

OSDResults.exe.config File Element Values

The OSD Results program, OSDResults.exe, is run at the end of a UDI deployment and displays the results of the deployment process. The behavior of the OSD Results program can be customized by modifying the OSDResults.exe.config file element values. The OSDResults.exe.config file is stored in Tools\OSDResults in the MDT Package in the User Drive Installation Task Sequence.

backgroundOpacity

This XML element configures the opaqueness of the background wallpaper image specified as a decimal-formatted percentage in the **backgroundWallpaper** element.

Expand table

Value	Description
opacity_percent	The percentage of opaqueness of the **backgroundWallpaper** element specified in a decimal formatted percentage—for example, a value of **0.8** designates 80% opaqueness.

Expand table

Example
`<add key="backgroundOpacity" value="0.8"/>`

backgroundWallpaper

This XML element provides the file name and relative path to the image that is displayed as the background in the **OSD Results** dialog box. The path is relative to the Tools\OSDResults folder in the MDT Package.

Expand table

Value	Description
*path**file_name*	Includes the relative path and file name of the background image; the path is delimited with double forward slashes (//).

Expand table

Example
`<add key="backgroundWallpaper" value="images\\Wallpaper.jpg"/>`

completedText

This XML element provides the text that is displayed in the **OSD Results** dialog box when the deployment is complete.

Expand table

Value	Description
text	The text to be displayed in the **OSD Results** dialog box in quotation marks when deployment is complete

Expand table

Example
`<add key="completedText" value="Deployment Complete"/>`

headerImagePath

This XML element provides the file name and relative path to the image that is displayed in the header of the **OSD Results** dialog box. The path is relative to the Tools\OSDResults folder in the MDT Package.

Expand table

Value	Description
*path**file_name*	Includes the relative path and file name of the header image; the path is delimited with double backslashes (\\).

Expand table

Example
`<add key="headerImagePath" value="images\\Windows7_h_rgb.png"/>`

timeoutMinutes

This XML element configures how many minutes the **OSD Result** dialog box is displayed before the dialog box is automatically closed and the computer is restarted.

Expand table

Value	Description
Non-numeric value	The dialog box remains opens until **Start Windows** is selected.
Negative value	The dialog box remains opens until **Start Windows** is selected.
0	The dialog box remains opens until **Start Windows** is selected.
Include decimal point	The dialog box remains opens until **Start Windows** is selected.
1 - 10080	The number of minutes the dialog box will be displayed, with a minimum value of 1 minute and a maximum value of 10080 minutes (1 week).

Expand table

Example
`<add key="timeoutMinutes" value="30"/>`

welcomeText

This XML element provides the welcome text that is displayed in the **OSD Results** dialog box.

Expand table

Value	Description
welcome_text	The welcome text to be displayed in the **OSD Results** dialog box in quotation marks

Expand table

Example
`<add key="welcomeText" value="Congratulations, Windows 7 has been successfully deployed to your computer."/>`

Related articles

- Task Sequence Steps.
- Properties.
- Scripts.
- Support Files.
- Utilities.
- MDT Windows PowerShell Cmdlets.
- Tables and Views in the MDT DB.
- Windows 7 Feature Dependency Reference.

Feedback

Was this page helpful? Yes No

Provide product feedback

Troubleshooting Reference for the Microsoft Deployment Toolkit

Article • 02/11/2025

The deployment of operating systems and applications as well as the migration of user state can be a challenging endeavor, even when you are equipped with appropriate tools and guidance. This reference, which is part of Microsoft® Deployment Toolkit (MDT) 2013, provides information on current known issues, possible workarounds for those issues, and troubleshooting guidance.

> **Note**
>
> In this document, *Windows* applies to the Windows 8.1, Windows 8, Windows 7, Windows Server 2012 R2, Windows Server 2012, and Windows Server 2008 R2 operating systems unless otherwise noted. MDT does not support ARM processor–based versions of Windows. Similarly, *MDT* refers to MDT 2013 unless otherwise stated.

> **Note**
>
> The Microsoft Diagnostics and Recovery Toolset (DaRT) contains powerful tools for recovering and troubleshooting client computers that do not start or have become unstable. You can use DaRT to determine the cause of a crash, restore lost files, and so on. You can also use DaRT as a troubleshooting tool when developing and deploying a Windows operating system. For example, if a built image fails to start correctly, you can start the client computer containing the image by using ERD Commander—a diagnostic environment. Then, you can explore the client computer's hard disk, view the event log, remove updates, change operating system settings, and so on. DaRT is part of the Microsoft Desktop Optimization Pack for Software Assurance. For more information, see Diagnostics and Recovery Toolset 10.

Understanding Logs

Before effective troubleshooting of MDT can begin, you must have a clear understanding of the many .log files used during an operating system deployment. When you know which log files to research for what failure condition and at what time, issues that were once mysterious and difficult to understand may become clear and understandable.

The MDT log file format is designed to be read by CMTrace. Use this tool whenever possible to read the log files, because it makes finding errors much easier.

The rest of this section details the log files created during deployment as well as during Windows Setup. This section also provides examples of when to use the files for troubleshooting.

MDT Logs

Each MDT script automatically creates log files when running. The names of these log files match the name of the script—for example, ZTIGather.wsf creates a log file named *ZTIGather.log*. Each script also updates a common master log file (BDD.log) that aggregates the contents of the log files that MDT scripts create. MDT log files reside in C:\MININT\SMSOSD\OSDLOGS during the deployment process. Depending on the type of deployment being conducted, the log files are moved at the completion of the deployment to either %WINDIR%\SMSOSD or %WINDIR%\TEMP\SMSOSD. For Lite Touch Installation (LTI) deployments, the logs start in C:\MININT\SMSOSD\OSDLogs. They end up in %WINDIR%\TEMP\DeploymentLogs when the task sequence processing is complete.

MDT creates the following log files:

- **BDD.log**. This is the aggregated MDT log file that is copied to a network location at the end of the deployment if you specify the **SLShare** property in the Customsettings.ini file.
- **LiteTouch.log**. This file is created during LTI deployments. It resides in %WINDIR%\TEMP\DeploymentLogs unless you specify the **/debug:true** option.
- **Scriptname*.log**. This file is created by each MDT script. *Scriptname* represents the name of the script in question.
- **SMSTS.log**. This file is created by the Task Sequencer and describes all Task Sequencer transactions. Depending on the deployment scenario, it may reside in %TEMP%, %WINDIR%\System32\ccm\logs, or C:_SMSTaskSequence, or C:\SMSTSLog.
- **Wizard.log**. The deployment wizards create and update this file.
- **WPEinit.log**. This file is created during the Windows PE initialization process and is useful for troubleshooting errors encountered while starting Windows PE.
- **DeploymentWorkbench_*id*.log**. This log file is created in the %temp% folder when you specify **a /debug** when starting the Deployment Workbench.

Configuration Manager Operating System Deployment Logs

For information about which operating system deployment log files created by Microsoft System Center 2012 R2 Configuration Manager, see Technical Reference for Log Files in

Configuration Manager.

When running the Windows User State Migration Tool (USMT), MDT automatically adds the logging options to save the USMT log files to the MDT log file locations. The log files and when they are created are as follows:

- **USMTEstimate.log**. Created when estimating the USMT requirements
- **USMTCapture.log**. Created by the USMT when capturing data
- **USMTRestore.log**. Created by the USMT when restoring data

  The ZeroTouchInstallation.vbs script automatically scans the USMT progress log files for errors and warnings. The script generates event ID 41010 to Microsoft System Center Operations Manager with the following summary (where *usmt_type* is **ESTIMATE**, **SCANSTATE**, or **LOADSTATE**; *error_count* is the total number of errors found; and *warning_count* is the total number of warnings found):

```vbs
ZTI USMT <usmt_type> reported <error_count> errors and <warning_count> warnings
```

If the error count is greater than 0, this event is an Error type. If the warning count is greater than 0 with no errors, then the event is a Warning type. Otherwise, the event is an Informational type.

Identifying Error Codes

Table 1 lists the error codes that the MDT scripts create and provides a description of each error code. These error codes are recorded in the BDD.log file.

Table 1. Error Codes and Their Description

Expand table

Error code	Description
5201	A connection to the deployment share could not be made. The deployment will not proceed.
5203	A connection to the deployment share could not be made. The deployment will not proceed.
5205	A connection to the deployment share could not be made. The deployment will not proceed.

Error code	Description
5206	The Deployment Wizard was canceled or did not complete successfully. The deployment will not proceed.
5207	A connection to the deployment share could not be made. The deployment will not proceed.
5208	**DeploymentType** is not set. Must set some value for **SkipWizard**.
5208	Unable to find the SMS Task Sequencer. The deployment will not proceed.
5400	Create object: **Set *class_instance* = New *class_name***
5490	Create MSXML2.DOMDocument.
5495	Create MSXML2.DOMDocument.ParseErr.ErrCode.
5496	LoadControlFile.FindFile: *ConfigFile*
5601	Verify OS guid: %OSGUID% exists.
5602	Open XML with OSGUID: %OSGUID%.
5610	Verify file.
5630	Verify file: *ImagePath*.
5640	Verify file: *ImagePath*.
5641	FindFile: ImageX.exe.
5643	Find BootSect.exe.
5650	Verify directory: *SourcePath*.
5651	Verify directory: *SourcePath*\Platform.
5652	FindFile: bootsect.exe.
6001	Verify drive.
6002	Verify drive.
6010	Test for TSGUID.
6020	Robocopy returned value: *Value*.
6021	Robocopy returned value: *Value*.
6101	Check for file: *DeployCab*.
6102	Expand Sysprep files from DEPLOY.CAB.

Error code	Description
6111	Run Sysprep.exe.
6121	Run Sysprep.
6191	Test for **CloneTag** in registry to verify Sysprep completed.
6192	Test for **SystemSetupInProgress** in registry to verify Sysprep completed.
6401	Authorized DHCP server.
6501	Computer backup not possible, no network path (**BackupShare**, **BackupDir**) specified.
6502	ERROR - Unable to locate IMAGEX, unable to perform backup.
6601	GetObject(... root/wmi:BCDStore).
6602	BCD.OpenStore (*BCDStore*).
6701	Configured protectors.
6702	Moved boot files.
6703	Create BDE partition.
6704	Defragment drive.
6705	Shrink drive.
6706	Testing for more than 1 partition.
6707	Create boot files.
6708	Encrypt the disk.
6709	Connect to MicrosoftVolumeEncryption WMI provider.
6710	Encrypting the disk.
6711	**ProtectKeyWithTPM**.
6712	**ProtectKeyWithTPMAndPIN**.
6713	**ProtectKeyWithTPMAndStartupKey**.
6714	Save external key to file.
6715	Protect with external key.
6716	Save external key to file.
6717	Protect key with numerical password.

Error code	Description
6718	**GetKeyProtectorNumberialP@ssword.**
6718	Save password to file.
6719	Open *PasswordFile*.
6720	Encrypt the drive.
6721	Open *DiskPartFile*.
6722	Create partition.
6723	Get existing BDE drive.
6724	Open *DiskPartFile*.
6727	Attempt to open *DiskPartFile*.
6729	Create text file *DiskPartFile*.
6730	Execute **cmd /c DISKPART.EXE /s *DiskPartFile* >> *LogPath*\ZTIMarkActive_diskpart.log 2>&1**
6731	Find bcdboot.exe.
6732	Connect to Microsoft TPM provider.
6733	Get a TPM instance in the provider class.
6734	Get TPM instance.
6735	Check to see if TPM is enabled.
6736	Check to see if TPM is activated.
6737	Check to see if TPM is owned.
6738	Check to see if TPM ownership is allowed.
6739	Check to see if TPM is enabled.
6740	Check to see if TPM is activated.
6741	Check to see if TPM is owned and ownership is allowed.
6741	TPM Owner Password set
6742	TPM Owner P@ssword set to **AdminP@ssword**.
6743	Set TPM Owner P@ssword to value.
6744	Check to see if TPM is enabled.

Error code	Description
6745	Check TPM owner.
6746	Check for endorsement key pair.
6747	Check to see if TPM is activated.
6748	Check to see if TPM ownership is allowed.
6749	Convert owner p@ssword to owner authorization.
6750	Create endorsement key pair.
6751	Change owner authorization.
6752	Run **Cmd**.
6753	Validate TPM.
6754	Get BDE instance.
6755	Protect key with TPM.
6756	Check for removable media to configure. **ProtectKeyWithTpmAndStartupKey**.
6757	Protect key with TPM and startup key.
6758	Look for BDE pin.
6759	Protect key with TPM and Pin.
6760	Find removable media for **BDEKeyLocation**.
6761	Protect with external key.
6762	Recovery P@ssword being saved to *PasswordFile*.
6764	Configure BitLocker policy.
7000	Unable to locate ZTIConfigure.xml; aborting.
7001	Looking for unattend AnswerFile.
7100	ERROR - This script should only run in the full OS.
7101	ERROR - Not enough values supplied for generating DCPromo answer file.
7102	ERROR - Mandatory properties for creating a new replica DC were not specified.
7103	ERROR - Mandatory properties for creating a new child domain were not specified.
7104	ERROR - Mandatory properties for creating a new forest were not specified.

Error code	Description
7105	ERROR - Mandatory properties for creating a new forest were not specified.
7200	Unable to configure DHCP server because the service is not installed.
7201	Unable to read the scope details; `GetScopeDetails()` failed.
7202	Not enough values specified for scope creation.
7203	Not enough values provided to set the IP range for this scope.
7204	No value specified for scope exclusion range.
7300	Unable to issue DNS commands.
7700	Not a New Computer scenario; exiting disk partition.
7701	Disk is not large enough for System and BDE partitions, Required = 1.5 GB.
7702	Disk is not large enough for System and WinRE partitions, Required = 10 GB.
7703	DeployRoot is on disk # *DiskIndex*. Running an OEM Scenario: Skip.
7704	Running an OEM Scenario: Skip.
7704	Extended and logical partitions are not allowed with BitLocker.
7712	Verify Drive/*Volume Drive* is present. Format.
7900	FindFile: Microsoft.BDD.PnpEnum.exe.
7901	**AllDrivers.Exists("*GUID*").**
7904	**AllDrivers.Exists("*GUID*").**
9200	FindFile(PkgMgr.exe).
9601	ERROR - ZTITatoo state restore task should be running in the full OS; aborting.
9701	Nonzero return code from USMT estimate, rc = *Error*.
9702	User state capture not possible; insufficient local space and no network path (UDShare, UDDir) specified.
9703	Nonzero return code from USMT capture, rc = *Error*.
9704	No valid command line option was specified.
9801	ERROR - Attempting to deploy a client operating system to a machine running a server operating system.

Error code	Description
9802	ERROR - Attempting to deploy a server operating system to a machine running a client operating system.
9803	ERROR - Machine is not authorized for upgrading (OSInstall=*OSInstall*); aborting.
9804	ERROR - *Memory* MB of memory is insufficient. At least *Memory* MB of memory is required.
9805	ERROR - Processor speed of *ProcessorSpeed* MHz is insufficient. At least a *ProcessorSpeed* MHz processor is required.
9806	ERROR - insufficient space is available on *Drive*. An additional *Size* MB is required.
9807	ERROR - insufficient space is available on *Drive*. An additional *Size* MB is required.
9901	The ZTIWindowsUpdate script should not run in Windows PE.
9902	ZTIWindowsUpdate has run and failed too many times. Count = *Count*.
9903	Unexpected issue installing the updated Windows Update Agent, rc = *Error*.
9904	Failed to create object: **Microsoft.Update.Session**.
9905	Failed to create object: **Microsoft.Update.UpdateColl**.
9906	Critical file *File* was not found; aborting.
10000	Create object: **Set oLTICleanup = New LTICleanup**.
10201	Unable to Join Domain *Domain*. Stop installation.
10203	FindFile(LTISuspend.wsf).
10204	Run Program *LTISuspend*.
41024	Run ImageX.
52012	All the wizard parameters are not set.

Listing 1 provides an excerpt from a log file that illustrates how to find the error code. In this excerpt, the error code reported is 5001.

Listing 1. Excerpt from an SMSTS.log file that contains error code 5001

Output

```
.
.
.
The operating system installation failed. Please contact your system administrator
```

```
for assistance.

The action "Zero Touch Installation - Validation" failed with exit code 5001
.
.
.
```

Converting Error Codes

Many error codes presented in the log files seem cryptic and difficult to correlate to an actual error condition. However, the following process demonstrates how to convert an error code and obtain meaningful information that may assist in problem resolution.

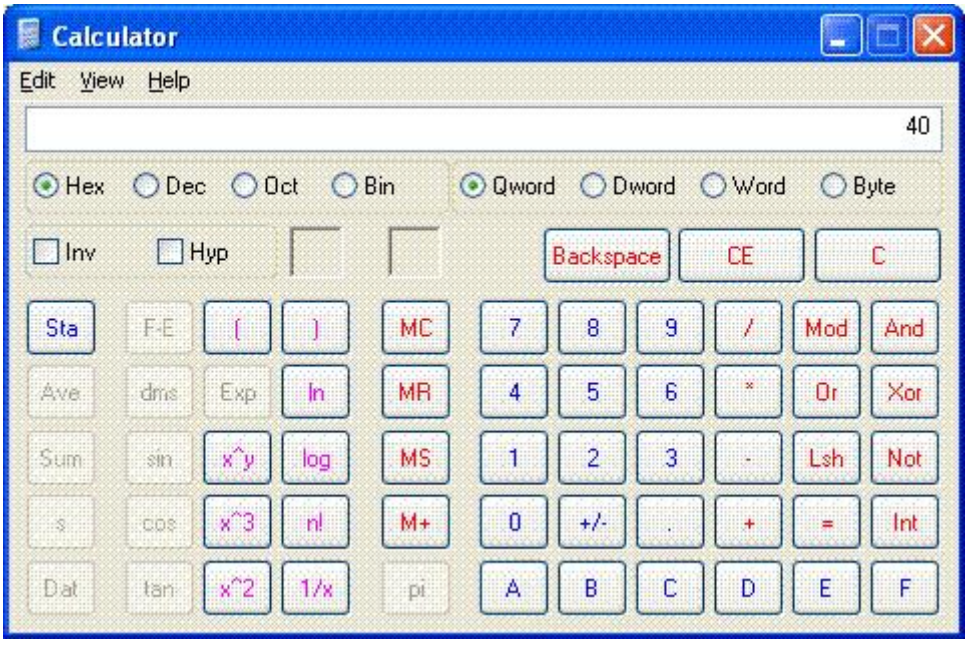
Problem: An image capture fails with error code 0x80070040.

Possible Solution 1: The error code presented is in hexadecimal format that you need to convert to decimal format. To do this, you need a scientific calculator, and the calculator included with Windows operating systems is well suited to this task.

To convert an error code

1. Click **Start**, and then point to **All Programs**. Point to **Accessories**, and then click **Calculator**.
2. From the **View** menu, click **Scientific**.
3. Select **Hex**, and then enter the last four digits of the code—in this case, **0040**, as shown in Figure 1.

   Figure 1. Error conversion



Notice that leading zeros are not displayed while the calculator is in Hexadecimal mode.

4. Select **Dec**.

   The hexadecimal value *40* is converted to a decimal value of *64*.

5. Open a Command Prompt window, type **NET HELPMSG 64**, and then press ENTER.

   The **NET HELPMSG** command translates the numerical error code into meaningful text. In the case of the error code provided here, it translates to "The specified network name is no longer available."

   This information indicates that a networking problem may exist on the target computer or between the target computer and the server on which the deployment share resides. These problems might include network drivers not being installed properly or a mismatch in speed and duplex settings.

Review of Sample Logs

MDT creates log files that you can use to troubleshoot problems in the MDT deployment process. The following sections provide examples of how to use the MDT log files to troubleshoot the deployment process:

- Problems that relate to failures accessing the MDT database (MDT DB), as described in Failure to Access the Database

Failure to Access the Database

Problem: An error occurs while running a deployment that used a CustomSettings.ini file containing numerous sections and specifying, with the **Priority** property, the priority of each section to be processed. BDD.log contains the following error messages:

- > ERROR - Opening Record Set (Error Number = -2147217911) (Error Description: The SELECT permission was denied on the object 'ComputerAdministrators', database 'AdminDB', schema 'dbo'.)

- > ADO error: The SELECT permission was denied on the object 'ComputerAdministrators', database 'AdminDB', schema 'dbo'. (Error #-2147217911; Source: Microsoft OLE DB Provider for SQL Server; SQL State: 42000; NativeError: 229

- > ERROR - Unhandled error returned by ZTIGather: Object required (424)

> ⓘ **Note**
>
> For clarity, the log file contents above have been represented as they appear while being viewed using CMTrace.

Possible Solution: The issue, as pointed out on the first line of the log file sample, is that permission to access the database was denied. Therefore, the script cannot establish a secure connection to the database, possibly because a user ID and password were not available. As a result, database access was attempted using the computer account. The easiest way to work around this issue is to grant everyone Read access to the database.

Troubleshooting

Prior to embarking on in-depth troubleshooting processes, review the following items and ensure that any associated requirements have been met:

- Installation issues can result if all software and hardware prerequisites have not been met.

Application Installation

Review the problems and solutions for application installation issues:

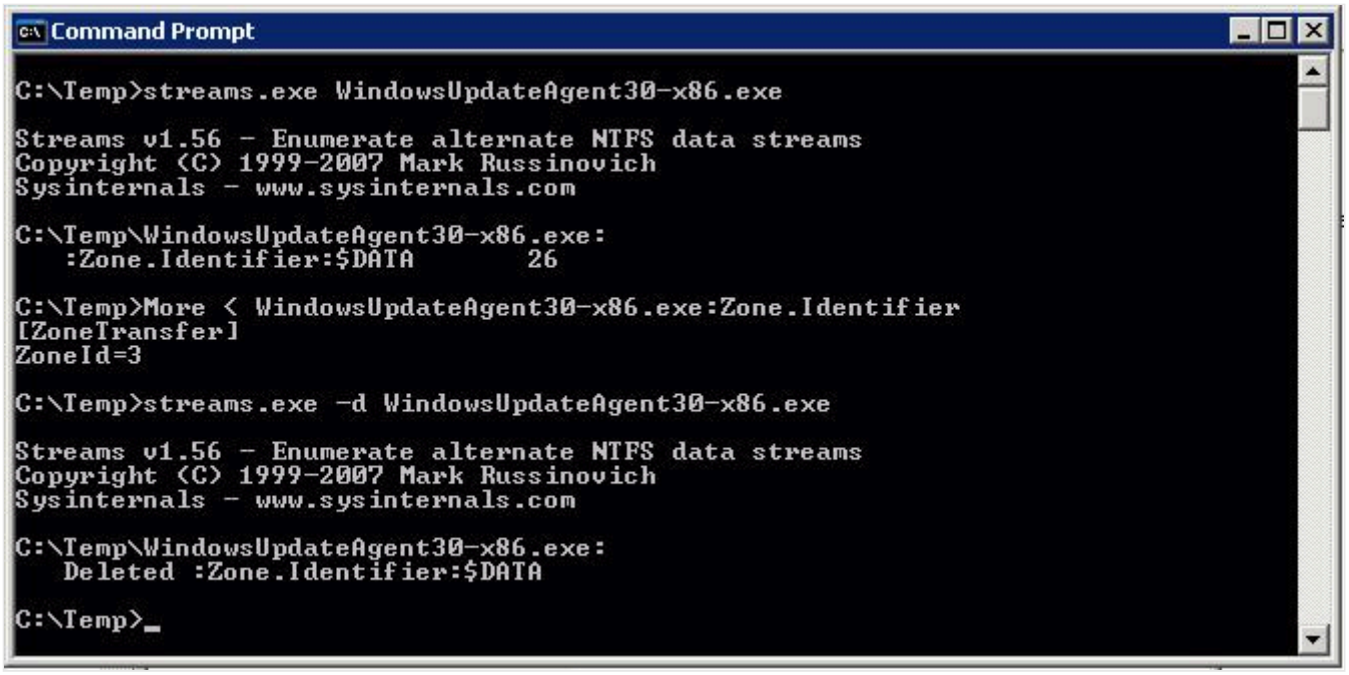
- Installation source files that are blocked for security reasons as described in Blocked Executables
- Loss of network connectivity as described in Lost Network Connections
- Installation error 30029 while installing the 2007 Microsoft Office system or related files as described in The 2007 Microsoft Office System

Blocked Executables

Problem: If installation source files are downloaded from the Internet, it is likely that they will be marked with one or more NTFS file system data streams. For more information about NTFS data streams, see File Streams. The existence of NTFS file system data streams might cause an **Open File – Security Warning** prompt to be displayed. The installation will not proceed until you click **Run** at the prompt.

Figure 2 shows, you can view NTFS file system data streams using the **More** command and the Streams utility.

Figure 2. NTFS data streams



Possible Solution 1: Right-click the installation source file, and then click **Properties**. Click **Unblock**, and then click **OK** to remove the NTFS file system data streams from the file. Repeat this process for each installation source file that is blocked by the existence of one or more NTFS file system data streams.

Possible Solution 2: Use the Streams utility, as REF _Ref308173670 \h Figure 2 shows, to remove the NTFS file system data streams from the installation source file. The Streams utility can remove NTFS file system data streams from one or more files or folders at once.

Lost Network Connections

Problem: An installation may fail if it installs device drivers or alters device and network configurations. These changes may result in a lapse in network connectivity that causes the installation to fail.

Possible Solution: Implement the ZTICacheUtil.vbs script to enable download and execution for the installation. This script is designed to tweak the advertisement to enable download and execute. The download uses Background Intelligent Transfer Service (BITS) if the Configuration Manager distribution point is Web-based Distributed Authoring and Versioning and BITS enabled. At the same time, it modifies Configuration Manager to run the ZTICache.vbs script first, which makes sure the program does not delete itself during the deployment process.

The 2007 Microsoft Office System

Problem: While deploying the 2007 Office system and including a Windows Installer patch (MSP) file, the installation may fail with error code 30029.

Further investigation in the ZTIApplications.log shows the following messages:

- > About to run command: `\\Server\Deployment$\Tools\X86\bddrun.exe \\Server\Share\Microsoft\Office\2007\Professional\setup.exe /adminfile \\Server\Share\Microsoft\Office\2007\Professional\file.msp`

- > ZTI Heartbeat: command has been running for 12 minutes (process ID 1600) Return code from command = 30029

- > Application Microsoft Office 2007 Professional returned an unexpected return code: 30029

  Possible Solution 1: Relocate the MSP file to the Updates directory, and then run setup.exe without specifying the **/adminfile** option. For more information about deploying updates during the installation, see Deploying the 2007 Office system.

  Possible Solution 2: Verify that the MSP file does not have the **Suppress modal** check box selected. For more information about configuring this setting, see Overview of 2007 Office System Deployment.

AutoLogon

Review the problems and solutions for automatic logon issues:

- Interruption of the LTI and Zero Touch Installation (ZTI) deployment processes because of logon security banners as described in Logon Security Banners
- Interruption of the LTI and ZTI deployment processes because of prompts for user credentials as described in Prompted for User Credentials

Logon Security Banners

Problem: MDT task sequences are processed during an interactive user session, which requires that the target computer be allowed to log on automatically using a specified administrative account. If a Group Policy object (GPO) is in place that enforces a logon security banner, this automatic logon will not be allowed to proceed, because the security banner halts the logon process while it waits for a user to accept the stated policy.

Possible Solution: Be sure that the GPO is applied to specific organizational units (OUs) and not included in the default domain GPO. When you add computers to the domain, specify that they be added to an OU that is not affected by a GPO that enforces a logon security banner. In the Task Sequence Editor, include as one of the last task sequence steps a script that relocates the computer account to the desired OU.

> **Note**
>
> If you are reusing existing Active Directory® Domain Services (AD DS) accounts, ensure that prior to deploying to the target computer you have relocated the target computer's account to an OU that is not affected by the GPO that enforces the security logon banner.

Prompted for User Credentials

Problem: You created an image of a computer that was joined to the domain. While deploying the new image to a target computer, the deployment process halts, because auto-logon does not occur and the user is prompted to enter appropriate credentials. The deployment process resumes when the credentials are provided and the user is logged on.

Possible Solution: When capturing images, the source computer should not be joined to a domain. If the computer was joined to a domain, join the computer to a workgroup, re-capture the image, and attempt the deployment to a target computer to determine whether the issue is resolved.

BIOS

Problem: While deploying to a target computer that is equipped with Intel vPro technology, the deployment may end with a stop error. Even though all updated drivers have been included as out-of-box drivers in the Deployment Workbench, the target computer does not start.

Possible Solution: Review the settings in the target computer's basic input/output system (BIOS) to determine whether the default Serial Advanced Technology Attachment mode is configured as Advanced Host Controller Interface (AHCI). Unfortunately, certain Windows operating systems do not support AHCI by default.

Database Problems

Review database-related problems and solutions:

- Errors generated as a result of improperly configured firewalls on database server as described in Blocked SQL Server Browser Requests
- Errors generated as a result of broken connections with the database server as described in Named Pipe Connections

Blocked SQL Server Browser Requests

Problem: During the MDT deployment process, information can be retrieved from Microsoft SQL Server® databases. However, errors might be generated that relate to an improperly configured firewall on the database server.

Possible Solution: The Windows Firewall in Windows Server helps prevent unauthorized access to computer resources. However, if the firewall is configured incorrectly, attempts to connect to a SQL Server instance may be blocked. To access an instance of SQL Server that is behind the firewall, configure the firewall on the computer that is running SQL Server. For more information on configuring firewall ports for SQL Server, see Configure the Windows Firewall to Allow SQL Server Access.

Named Pipe Connections

Problem: During the MDT deployment process, information can be retrieved from SQL Server databases. However, errors might be generated that relate to broken SQL Server connections. These can be caused by not enabling named pipe connections in Microsoft SQL Server.

Possible Solution: To resolve these problems, enable named pipes in SQL Server. Also, specify the **SQLShare** property, which it is required when making a connection to an external database using named pipes. When connecting using named pipes, use integrated security to make the connection to the database. In the case of LTI deployments, the user account that you specify makes the connection to the database. For ZTI deployments that use Configuration Manager, the network access account connects to the database. Because Windows PE has no security context by default, you must make a network connection to the database server to establish a security context for the user who will be making the connection.

The network share that the **SQLShare** property specifies provides a means to connect to the server to gain a proper security context. You must have Read access to the share. When the connection is made, you can then establish the named pipe connection to the database. The **SQLShare** property is not needed and should not be used when making a TCP/IP connection to the database.

Enable named pipe connections by performing the following tasks based on the version of SQL Server you are using:

- Enable named pipe connections for SQL Server 2008 R2 as described in Enable Named Pipe Connections in SQL Server 2008 R2.
- Enable named pipe connections for SQL Server 2005 as described in Enable Named Pipe Connections in SQL Server 2005.

Enable Named Pipe Connections in SQL Server 2008 R2

To enable named pipe connections in SQL Server 2008 R2, perform the following steps:

1. On the computer running SQL Server 2008 R2 that hosts the database to be queried, click **Start**, and then point to **All Programs**. Point to **Microsoft SQL Server 2008 R2**, and then click **SQL Server Management Studio**.
2. In the **Microsoft SQL Server Management Studio** console, in the **Object Explorer,** right-click ***sql_server_name***, and then click **Properties** (where *sql_server_name* is the name of the computer running SQL Server to be configured).
3. The Server Properties - ***sql_server_name*** dialog box is displayed.
4. In the **Server Properties** - ***sql_server_name*** dialog box, in **Select a page**, click **Connections**.
5. On the **Connections** page, ensure the **Allow remote connections to this server** check box is selected and then click **OK**.
6. Close the Microsoft SQL Server Management Studio console.
7. On the computer running SQL Server 2008 R2 that hosts the database to be queried, click **Start**, and then point to **All Programs**. Point to **Microsoft SQL Server 2008 R2**, point to **Configuration Tools**, and then click **SQL Server Configuration Manager**.
8. In the **Sql Server Configuration Manager** console, go to SQL Server Configuration Manager (Local) / SQL Server Network Configuration / Protocols for *sql_instance* (where *sql_instance* in the name of the SQL Server instance to be configured).
9. In the details pane, right-click **Named Pipes**, and then click **Enable**.

   The Warning dialog box appears indicating that the changes will be saved but will not take effect until the service is stopped and restarted.
10. In the **Warning** dialog box, click **OK**.
11. In the **Sql Server Configuration Manager** console, go to SQL Server Configuration Manager (Local) / SQL Server Services.
12. In the details pane, right-click **SQL Server*(sql_instance)***, *and then click *Restart* (where *sql_instance* in the name of the SQL Server instance that you configured in step 2).

    The SQL Server Configuration Manager progress bar is displayed that shows the status of restarting the services. After the service restarts, the progress bar closes.
13. Close the SQL Server Configuration Manager console.

For additional information, How to enable remote connections in SQL Server 2008.

Enable Named Pipe Connections in SQL Server 2005

To enable named pipe connections in SQL Server 2005, perform the following steps:

1. On the computer running SQL Server 2005 that hosts the database to be queried, click **Start**, and then point to **All Programs**. Point to **Microsoft SQL Server 2005**, point to **Configuration Tools**, and then click **SQL Server Surface Area Configuration**.

2. In the **SQL Server 2005 Surface Area Configuration** dialog box, click **Surface Area Configuration for Services and Connections**.

3. In the **Surface Area Configuration for Services and Connections – *server_name*** dialog box (where *server_name* is the name of the computer running SQL Server 2005), in **Select a component and then configure its services and connections**, go to MSSQLSERVER\Database Engine, and then click **Remote Connections**.

4. Click **Local and remote connections**, click **Using both TCP/IP and named pipes**, and then click **Apply**.

5. In the **Surface Area Configuration for Services and Connections – *server_name*** dialog box (where *server_name* is the name of the computer running SQL Server 2005), in **Select a component and then configure its services and connections**, go to MSSQLSERVER\Database Engine, and then click **Service**.

6. Click **Stop**.

   The MSSQLSERVER service stops.

7. Click **Start**.

   The MSSQLSERVER service starts.

8. Click **OK**.

9. Close SQL Server 2005 Surface Area Configuration.

   For additional information, see the Microsoft Support article How to configure SQL Server 2005 to allow remote connections

Deployment Scripts

Review MDT-related problems and solutions:

- Prompted for user credentials and may receive error 0x80070035 as described in Credentials_script
- Error message "Wuredist.cab not found" appears as described in ZTIWindowsUpdate

Credentials_script

Problem: During the last start-up of a newly deployed computer, the user is prompted to provide user credentials and may receive error 0x80070035, which indicates that the network path was not found.

Possible Solution: Be sure that the WIM file does not include a MININT or _SMSTaskSequence folder. To delete these folders, first use the ImageX utility to mount the WIM file, and then delete the folders.

> ⓘ **Note**
>
> If an Access Denied error occurs when you attempt to delete the folders from the WIM file, open a Command Prompt window, switch to the root of the image contained in the WIM file, and then run **RD MININT** and **RD _SMSTaskSequence**.

ZTIWindowsUpdate

Problem: If you use the ZTIWindowsUpdate.wsf script to apply software updates during deployment, note that this script may communicate directly with the Microsoft Update website to download and install the required Windows Update Agent binaries, scan for applicable software updates, download the binaries for the applicable software updates, and then install the downloaded binaries. This process requires that your networking infrastructure be configured to allow the target computer to gain access to the Microsoft Update website.

If the deployment share does not contain the Windows Update Agent installation files and the target computer does not have appropriate Internet access, error "wuredist.cab not found" is reported in the ZTIWindowsUpdate.log and BDD.log files.

Possible Solution: Follow the steps outlined in the section, "ZTIWindowsUpdate.wsf", in the MDT document *Toolkit Reference*.

Deployment Shares

Review deployment share–related problems and solutions:

- Updating WIM files fails when updating a deployment share as described in Failure to Update WIM Files.

Failure to Update WIM Files

In a "simple" environment:

- MDT typically picks up WIMGAPI.DLL from C:\Windows\system32 (always in the path). The version of this WIMGAPI.DLL must match the version (build) of the operating system.
- On a 64-bit operating system, MDT always uses the x64 WIMGAPI.DLL file; only that file should be in the system PATH. On a 32-bit operating system, MDT always uses the x86 WIMGAPI.DLL file; only that file should be in the system PATH. (Other products, such as Configuration Manager, use the 32-bit version of WIMGAPI.DLL, even on a 64-bit operating system, but they manage and install that version.)

  Problem: When attempting to update a deployment share, the user will be informed that the mounting of one or more .wim files did not succeed.

  Possible Solution: Open a Command Prompt window and run **where WIMGAPI.DLL**. For the first entry in the list (the first location found by searching the path), ensure that the **Version** property matches the build of the Windows Assessment and Deployment Kit (Windows ADK) that is installed. Also ensure that the property matches the operating system build number.

The Windows Deployment Wizard

Review Windows Deployment Wizard–related problems and solutions:

- Windows Deployment Wizard pages are displayed even when LTI is configured to skip the wizard pages as described in Wizard Pages Are Not Skipped.

Wizard Pages Are Not Skipped

Problem: A wizard page is displayed even though the MDT DB or CustomSettings.ini file specify that the wizard should be skipped.

Possible Solution: To properly skip a wizard page, include all properties that would be specified on that wizard page where appropriate in the MDT DB or CustomSettings.ini file along with appropriate values. If a property is configured improperly for a skipped wizard page, that page will be shown. For more information about which properties are required to ensure that a wizard page is skipped, see the section, "Providing Properties for Skipped Deployment Wizard Pages", in the MDT document *Toolkit Reference*.

Disks and Partitioning

Review disk partitioning problems and solutions:

- BitLocker® Drive Encryption issues as described in BitLocker Drive Encryption
- Disk partitioning errors as described in Disk Partitioning Errors
- Failures during Refresh Computer deployment scenarios caused by logical or dynamic disks as described in Support for Logical and Dynamic Disks

BitLocker Drive Encryption

Deploying BitLocker requires a specific configuration for proper deployment. The following potential problems may be related to the configuration of the target computer:

- In ZTI and UDI deployments, the ZTIBde.wsf Script Fails with the Error "Unable to open registry key 'HKEY_CURRENT_USER\Control Panel\International\LocaleName' for reading", as described in ZTIBde.wsf Script Fails with the Error "Unable to open registry key 'HKEY_CURRENT_USER\Control Panel\International\LocaleName' for reading".
- USB devices, CD drives, DVD drives, or other removable media devices on the target computer that appear as multiple drive letters, as described in Devices Appear as Multiple Drive Letters
- Shrinking drive C on the target computer to provide sufficient unallocated disk space as described in Problems with Shrinking Disks

ZTIBde.wsf Script Fails with the Error "Unable to open registry key 'HKEY_CURRENT_USER\Control Panel\International\LocaleName' for reading"

Problem: While trying to deploy BitLocker on the target computer in ZTI or UDI, the ZTIBde.wsf script fails with the error "Unable to open registry key 'HKEY_CURRENT_USER\Control Panel\International\LocaleName' for reading."

Possible Solution: Specify the locale in the `UILanguage` property. In ZTI and UDI, the ZTIBde.wsf script runs in the system control, so a full user profile is not loaded. When the ZTIBde.wsf script tries to read the locale information it is not in the registry, because the registry (user profile) is not fully loaded. As a workaround, specify the locale in the `UILanguage` property.

Devices Appear as Multiple Drive Letters

Problem: Some devices can appear as multiple logical drive letters, depending on how they are partitioned. In some cases, they can emulate a 1.44-megabyte (MB) floppy disk drive and a memory storage drive. Therefore, Windows may assign the same device drive letters A and B for floppy disk emulation and F for the memory storage drive. By default, MDT scripts use the lowest drive letter (in this example, A).

Possible Solution: Override the default setting on the **Specify the BitLocker recovery details** page in the Windows Deployment Wizard. The Windows Deployment Wizard summary page displays a warning to inform the user which drive letter was selected to store BitLocker recovery information. In addition, the BDD.log and ZTIBDE.log files record the removable media devices detected and which device was selected to store the BitLocker recovery information.

Problems with Shrinking Disks

Problem: Not enough unallocated disk space exists on the target computer to enable BitLocker. To deploy BitLocker on a target computer, at least 2 gigabytes (GB) of unallocated disk space is required to create the system volume. The *system volume* is the volume that contains the hardware-specific files needed to load Windows after the BIOS has booted the computer.

Possible Solution 1: On existing computers, use the Diskpart tool to shrink drive C so that the system volume can be created. In some instances, though, the Diskpart tool may not be able to shrink drive C sufficiently to provide 2 GB of unallocated disk space, possibly because of fragmented disk space within drive C.

One possible solution to this problem is to defragment drive C. To do so, perform the following steps:

1. Run the Diskpart `shrink querymax` command to identify the maximum amount of disk space that can be unallocated.
2. If the value returned in step 1 is less than 2 GB, clean drive C of any unnecessary files, and then defragment it.
3. Run the Diskpart `shrink querymax` command again to verify that more than 2 GB of disk space can be unallocated.
4. If the value returned in step 3 is still less than 2 GB, perform one of the following tasks:
    - Defragment drive C multiple times to ensure that it is fully optimized.
    - Back up the data on drive C, delete the existing partition, create a new partition, and then restore the data to the new partition.

Possible Solution 2: The ZTIBDE.wsf script runs the Disk Preparation Tool (bdehdcfg.exe) and configures the system volume partition size to 2 GB by default. You can customize the ZTIBDE.wsf script to change the default, if necessary. However, modifying the MDT scripts is not recommended.

Support for Logical and Dynamic Disks

Problem: When performing a Refresh Computer deployment scenario, the deployment process may fail when deploying to a target computer that is using logical drives or dynamic disks.

Possible Solution: MDT does not support deploying operating systems to logical drives or dynamic disks.

Domain Join

Problem: During deployment, you use the Windows Deployment Wizard to provide all the necessary information for the target computer, including credentials, domain join information, and static IP configuration. When Setup finishes, you can see that the system has not joined the domain and is still in a workgroup.

Possible Solution: An LTI deployment of MDT configures the static IP information after the operating system is up and running. If the target computer is located on a network segment that does not have Dynamic Host Configuration Protocol (DHCP), an automated domain join specified in Unattend.xml will fail when no DHCP is present.

Configure Unattend.xml to join a workgroup. Then, use the built-in **Recover from Domain** task sequence step to add a step in the task sequence to join the domain after the static IP has been applied.

Driver Installation

To ensure the best possible user experience, installation of hardware devices and software drivers should run as seamlessly as possible, with little or no user intervention. Microsoft provides tools and guidelines to help create installation packages that meet this goal. For general information about driver installation, see Device and Driver Installation.

Review device driver installation–related problems and solutions:

- Problems that occur when using OEM mass storage drivers with MDT as described in Combine OEM Mass Storage Drivers with MDT Mass Storage Logic
- Troubleshooting device driver installation issues using the SetupAPI.log as described in Troubleshoot Device Installation with SetupAPI.log

Troubleshoot Device Installation with SetupAPI.log

The white paper Troubleshooting Device Installation with the SetupAPI Log File provides information about debugging Windows device installation. Specifically, the paper provides guidelines for driver developers and testers to interpret the SetupAPI log file.

One of the most useful log files for debugging purposes is the SetupAPI.log file. This plain-text file maintains the information that SetupAPI records about device installation, service pack installation, and update installation. Specifically, the file maintains a record of device and driver changes as well as major system changes beginning from the most recent Windows installation. This paper focuses on using the SetupAPI log file to troubleshoot device installation; it does not describe the log file sections that are associated with service pack and update installations.

New Computer Deployments

Review the problems and solutions for New Computer deployment scenarios:

- Problems starting the deployment process using Pre-Boot Execution Environment (PXE) boot as described in PXE Boot

PXE Boot

In brief, the PXE protocol operates as follows: The client computer initiates the protocol by broadcasting a DHCP Discover packet containing an extension that identifies the request as coming from a client computer that implements the PXE protocol. Assuming that a boot server implementing this extended protocol is available, the boot server sends an offer containing the IP address of the server that will service the client. The client uses Trivial File Transfer Protocol to download the executable file from the boot server. Finally, the client computer runs the downloaded bootstrap program.

The initial phase of this protocol piggybacks on a subset of the DHCP messages to enable the client to discover a boot server (that is, a server that delivers executable files for new computer setup). The client computer may use the opportunity to obtain an IP address (which is the expected behavior) but is not required to do so.

The second phase of this protocol takes place between the client computer and a boot server and uses the DHCP message format as a convenient format for communication. This second phase is otherwise unrelated to the standard DHCP services. The next few pages outline the step-by-step process during PXE client computer initialization.

Review the following solutions for PXE boot issues:

- Disable Windows PE logging to SetupAPI.log as described in Disable Windows PE Logging in Windows Deployment Services.
- Ensure that DHCP is configured properly as described in Ensure the Proper DHCP Configuration.
- Improve the response times for assigning IP addresses to PXE client computers as described in Improve PXE IP Address Assignment Response Time.

Disable Windows PE Logging in Windows Deployment Services

The first procedure recommended is to make sure that logging to setupapi.log has been disabled.

Ensure the Proper DHCP Configuration

Depending on the router models in use, the specific router configuration of DHCP broadcast forwarding may be supported to either a subnet (or router interface) or a specific host. If the DHCP servers and the computer running Windows Deployment Services are separate computers, ensure that the routers that forward DHCP broadcasts are designed so that both the DHCP and Windows Deployment Services servers receive the client broadcasts; otherwise, the client computer does not receive a reply to its remote boot request.

Is there a router between the client computer and the remote installation server that is not allowing the DHCP-based requests or responses through? When the Windows Deployment Services client computer and the Windows Deployment Services server are on separate subnets, configure the router between the two systems to forward DHCP packets to the Windows Deployment Services server. This arrangement is necessary, because Windows Deployment Services client computers discover a Windows Deployment Services server by using a DHCP broadcast message. Without DHCP forwarding set up on a router, the client computers' DHCP broadcasts do not reach the Windows Deployment Services server. This DHCP forwarding process is sometimes referred to as *DHCP Proxy* or *IP Helper Address* in router configuration manuals. Refer to the router instructions for more information about setting up DHCP forwarding on a specific router.

Improve PXE IP Address Assignment Response Time

Check the following elements if it is taking a long time (15–20 seconds) for the PXE client computer to retrieve an IP address:

- Are the network adapter on the target computer and the switch or router set to the same speed (automatic, duplex, full, and so on)

- Is the IP address for the Windows Deployment Services server in the IP Helper file on the router through which the connection is made? If the list of IP addresses in the IP Helper file is long, can you move the address for the Windows Deployment Services server near the top

Restarting the Deployment Process

Problem: While testing and troubleshooting a new or modified task sequence, you may need to restart the target computer so that the deployment process can start over from the beginning. Unexpected results may occur, because MDT keeps track of its progress by writing data to the hard disk; any restart of the target computer has MDT resume where it left off at the previous restart.

Possible Solution: To allow the deployment process to restart from the beginning, delete the C:\MININT and C:_SMSTaskSequence folders prior to restarting the target computer.

Sysprep

Review Sysprep-related problems and solutions:

- The target computer is not appearing in the correct AD DS OU as described in The Computer Account Is in the Wrong OU.

The Computer Account Is in the Wrong OU

Problem: The target computer is properly joined to the domain, but the computer account is in the wrong OU.

Possible Solution 1: If an account pre-exists for the target computer, the account will remain in its original OU. To move the account to the specified OU, add a task sequence step that uses an automation tool, such as a Microsoft Visual Basic® Scripting Edition, to move the account.

Possible Solution 2: Verify that the specified OU is in the correct format and that it exists. The correct OU format should be `OU=Reception,OU=NYC,DC=Woodgrovebank,DC=com`.

Configuration Manager

Problem: The error message shown in REF _Ref308174600 \h Figure 3 is displayed when you attempt to create a Configuration Manager PXE service point using the **Create self-signed PXE certificate** option.

Figure SEQ Figure * ARABIC 3. PXE service point error

Possible Solution: If a PXE service point previously existed on the server you are configuring, the PXE service point may not have deleted the self-created certificates when you uninstalled it. Delete the PXE certificate folder from C:\Documents and Settings*user_name*\Application Data\Microsoft\Crypto\RSA, where *user_name* is the name of the user performing the current configuration or who performed the previous configuration. The New Site Role Wizard in the Configuration Manager console should successfully finish when you have deleted the folder.

Task Sequences

Review task sequence–related problems and solutions:

- Task sequence does not finish successfully or has unpredictable behavior as described in The Task Sequence Does Not Finish Successfully.
- Original equipment manufacturer (OEM) task sequences in LTI are listed on boot images with the opposite processor architecture as described in The OEM Task Sequence Incorrectly Appears for a Boot Image Created for a Different Processor Architecture.
- The Windows Deployment Wizard displays the error message "Bad Task Sequence Item (Invalid OS GUID)" as described in Bad Task Sequence Item (Invalid OS GUID) Message in the Windows Deployment Wizard.
- While configuring a network connection name, the message "Please enter a valid name for the network adapter" is displayed as described in Apply Network Settings.
- Problems that may occur as a result of improper configuration of continue on error configuration settings for task sequence steps as described in Use Continue on Error.

The Task Sequence Does Not Finish Successfully

Problem: Task sequence may not finish successfully or has unpredictable behavior.

Possible Solution: The **Install Operating System** task sequence step (for LTI) or the **Apply Operating System Image** task sequence step (for UDI and ZTI) may have been modified after the creation of the task sequence step can lead to unpredictable results. For example, if a task sequence was created to deploy a 32-bit Windows 8.1 image, and then later the **Install Operating System** task sequence step or the **Apply Operating System Image** task sequence

step was changed to reference a 64-bit Windows 8.1 image, the task sequence may not run successfully.

It is recommended that a new task sequence is created to deploy a different operating system image.

The OEM Task Sequence Incorrectly Appears for a Boot Image Created for a Different Processor Architecture

Problem: A task sequence based on a LTI OEM task sequence template is showing up for a boot image with a different processor architecture. For example, an OEM task sequence that deploys a 64-bit operation system is showing on a 32-bit boot image.

Possible Solution: This is expected behavior as OEM task sequences in LTI are not considered to be "platform-specific" will always be listed, regardless of the processor architecture of the boot image.

Bad Task Sequence Item (Invalid OS GUID) Message in the Windows Deployment Wizard

Problem: When running the Windows Deployment Wizard, the wizard displays the error message "Bad Task Sequence Item (Invalid OS GUID)." The operating system is listed in the OperatingSystem.xml file; however, the operating system is not displayed in the Deployment Workbench.

Possible Solution: The original operating system source has two or more WIM files associated. A SKU that is associated with a task sequence is deleted; however, other SKUs for the operating system source still exist. When the task sequence that references the deleted SKU is selected on the **Select a task sequence to execute on this computer** wizard page in the Windows Deployment Wizard, the error message "Bad Task Sequence Item (Invalid OS GUID)" is displayed after you click **Next** on the wizard page.

To resolve this problem, perform one of the following tasks:

- Remove all SKUs from the operating system source. The Windows Deployment Wizard behaves normally, and the error message is not displayed.
- Change the task sequence to use a different operating system image.

Apply Network Settings

Problem: When configuring the network connection name in the Deployment Workbench, a validation error prompts you with the message, "Please enter a valid name for the network

adapter."

Possible Solution: Remove any spaces and invalid characters from the specified connection name.

Use Continue on Error

If a MDT task sequence is configured not to continue on error and that task sequence returns an error, all remaining task sequences in that task sequence group are skipped. However, the remaining task sequence groups are processed. Consider the following:

Two task sequence groups have been created, and either group contains more than one task sequence step:

- Group A
  - Step A
  - Step B
- Group B
  - Step A
  - Step B

  If Group A\Step A is configured not to continue on error, then Group A\Step B will not be processed. However, all task sequence steps in Group B will be processed.

The User State Migration Tool

Review USMT-related problems and solutions:

- Shortcuts that point to documents stored in network shared folders may not be restored properly as described in Missing Desktop Shortcuts.

Missing Desktop Shortcuts

Problem: While using USMT to migrate user data, shortcuts that point to network documents may not be restored. The shortcuts are captured during Scanstate; however, they are never restored to the target computer during Loadstate.

Possible Solution: Edit the MigUser.xml file and comment out the following line:

Original:

```
XML
```

```
<include> filter='MigXmlHelper.IgnoreIrrelevantLinks()'>
```

Modified:

```
XML
<include> <!-- filter='MigXmlHelper.IgnoreIrrelevantLinks()'> -->
```

Windows Imaging Format Files

Review WIM-related problems and solutions:

- LTI and ZTI deployments fail with WIM file errors in the BDD.log file as described in Corrupt WIM File.

Corrupt WIM File

Problem: When deploying an image, the deployment fails with the following entries in the BDD.log file:

- > The image `\\Server\Deployment$\Operating Systems\Windows\version1.wim` was not applied successfully by ImageX, rc = 2

- > LTIApply COMPLETED. Return Value = 2

- > ZTI ERROR - Non-zero return code by LTIApply, rc = 2

  Investigate the issue by mounting the WIM file using ImageX results in the error, "The data is invalid." Further investigation shows that the date stamp of the .wim file is many years before the current date. It is possible that another process, such as a virus scanner, was holding the .wim file open after it was previously closed at the conclusion of a Read or Write process.

  Possible Solution: Restore the .wim file from backup media.

Windows PE

Review Windows PE–related problems and solutions:

- The LTI or ZTI deployment process is not initiated because of insufficient RAM or wireless network adapters as described in Deployment Process Not Initiated—Limited RAM or Wireless Network Adapter.
- The LTI or ZTI deployment process is not initiated because of missing Windows PE components as described in Deployment Process Not Initiated—Missing Components.
- The LTI or ZTI deployment process is not initiated because of missing or incorrect device drivers as described in Deployment Process Not Initiated—Missing or Incorrect Drivers.

Deployment Process Not Initiated—Limited RAM or Wireless Network Adapter

Problem: When deploying an image to certain target computers, Windows PE starts, runs **wpeinit**, opens a Command Prompt window but does not actually start the deployment process. Troubleshooting the problem by mapping a network drive from the target computer indicates that the network adapter drivers are not loaded.

Possible Solution 1: The Deployment Wizard is not starting, because there is insufficient RAM. Verify that the target computer has at least 512 MB of RAM and that no shared video memory consumes more than 64 MB of the 512 MB.

The versions of Windows PE that MDT supports are unable to run on a target computer that has less than 512 MB of RAM.

Possible Solution 2: Do not include the wireless drivers in the Windows PE image.

Deployment Process Not Initiated—Missing Components

Problem: When troubleshooting a failed deployment, a review of the BDD.log file lists the following entry:

> ERROR - Unable to create ADODB.Connection object, impossible to query SQL Server: ActiveX component can't create object (429).

Possible Solution: This error may indicate that the Windows PE image was not created using MDT. If you are using Configuration Manager, do not use one of the existing Windows PE images that Configuration Manager created; instead, create an image using the Import Microsoft Deployment Task Sequence Wizard.

> ⓘ **Note**

The Windows PE images that Configuration Manager creates contain components that support scripting, XML, and Windows Management Instrumentation (WMI), but they do not contain components that support Microsoft ActiveX® Data Objects (ADO).

Deployment Process Not Initiated—Missing or Incorrect Drivers

Problem: When deploying to certain target computers, Windows PE starts, runs **wpeinit**, opens a Command Prompt window, but does not actually start the deployment process. Troubleshooting by mapping a network drive from the target computer indicates that the network adapter drivers are not loaded. A review of the SetupAPI.log file located in *X*:\Windows\System32\Inf indicates that Windows PE generates errors when it is configuring the network adapter, one of which is, "This driver is not meant for this platform." The drivers in the **Out-of-Box Drivers** list have been injected into the image.

Possible Solution: It is possible that Windows PE is having a driver conflict with another driver. When configuring the settings for the Windows PE image in the Deployment Workbench, create a Windows PE drivers group that contains only network adapter and storage drivers, and then configure the deployment share to use only the Windows PE driver group.

Deployment Process Flow Charts

This section provides two sets of MDT flow charts: one for LTI deployments and one for ZTI deployments with Configuration Manager. Each flow chart illustrates the tasks run during that deployment type.

Familiarize yourself with the deployment process flow charts by:

- Reviewing the LTI deployment process flowcharts as described in LTI Deployment Process Flowcharts
- Reviewing the ZTI deployment process flowcharts as described in ZTI Deployment Process Flowcharts

LTI Deployment Process Flowcharts

Flow charts are provided for the following phases:

- Validation (Figure 4)
- State Capture (Figure 5 and Figure 6)
- Preinstall (Figure 7, Figure 8, and Figure 9)

- Install (Figure 10)
- Postinstall (Figure 11 and Figure 12)
- State Restore (Figure 13, Figure 14, Figure 15, and Figure 16)

Figure 4. Flow chart for the Validation Phase

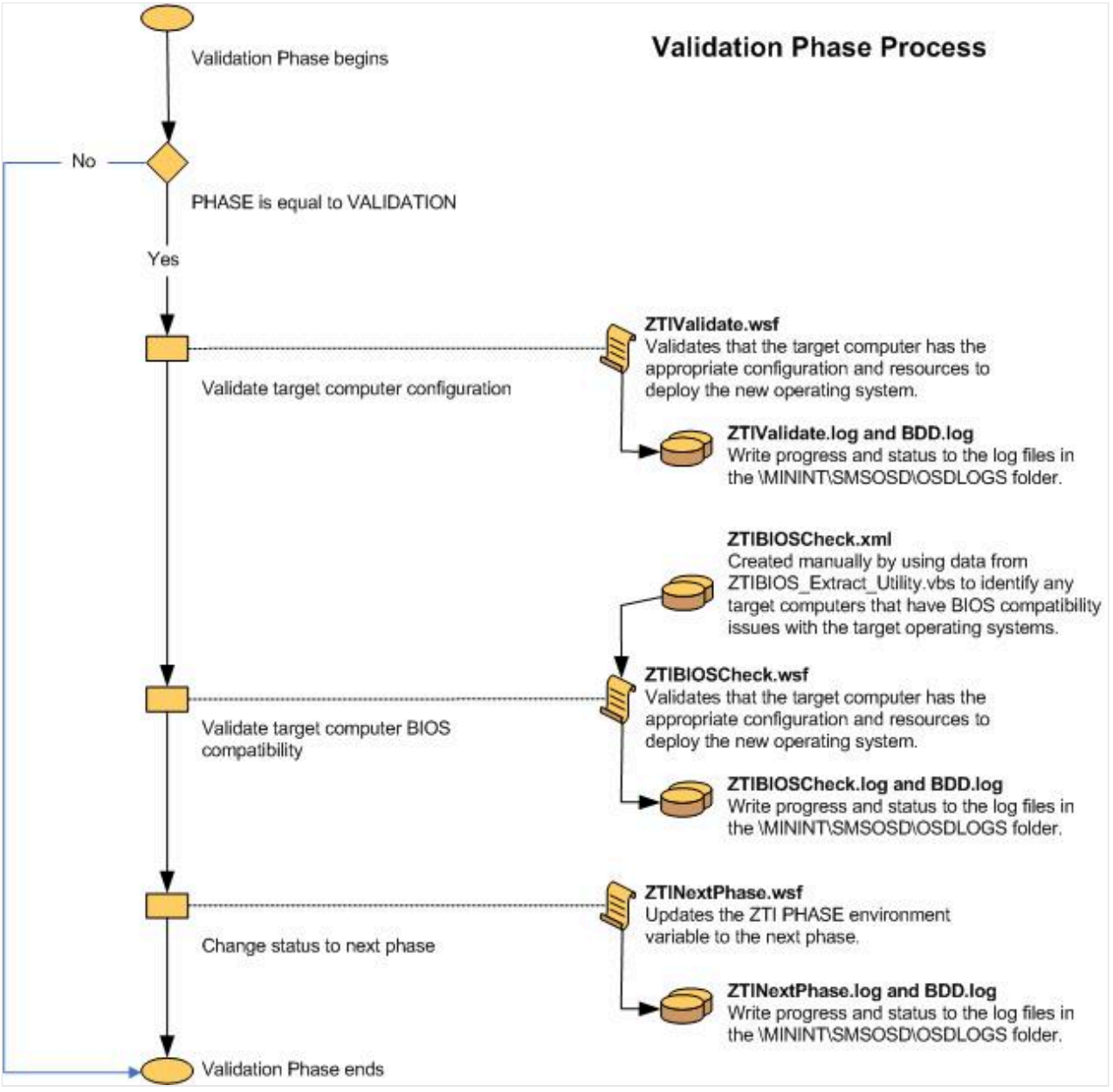


Figure 5. Flow chart for the State Capture Phase (1 of 2)

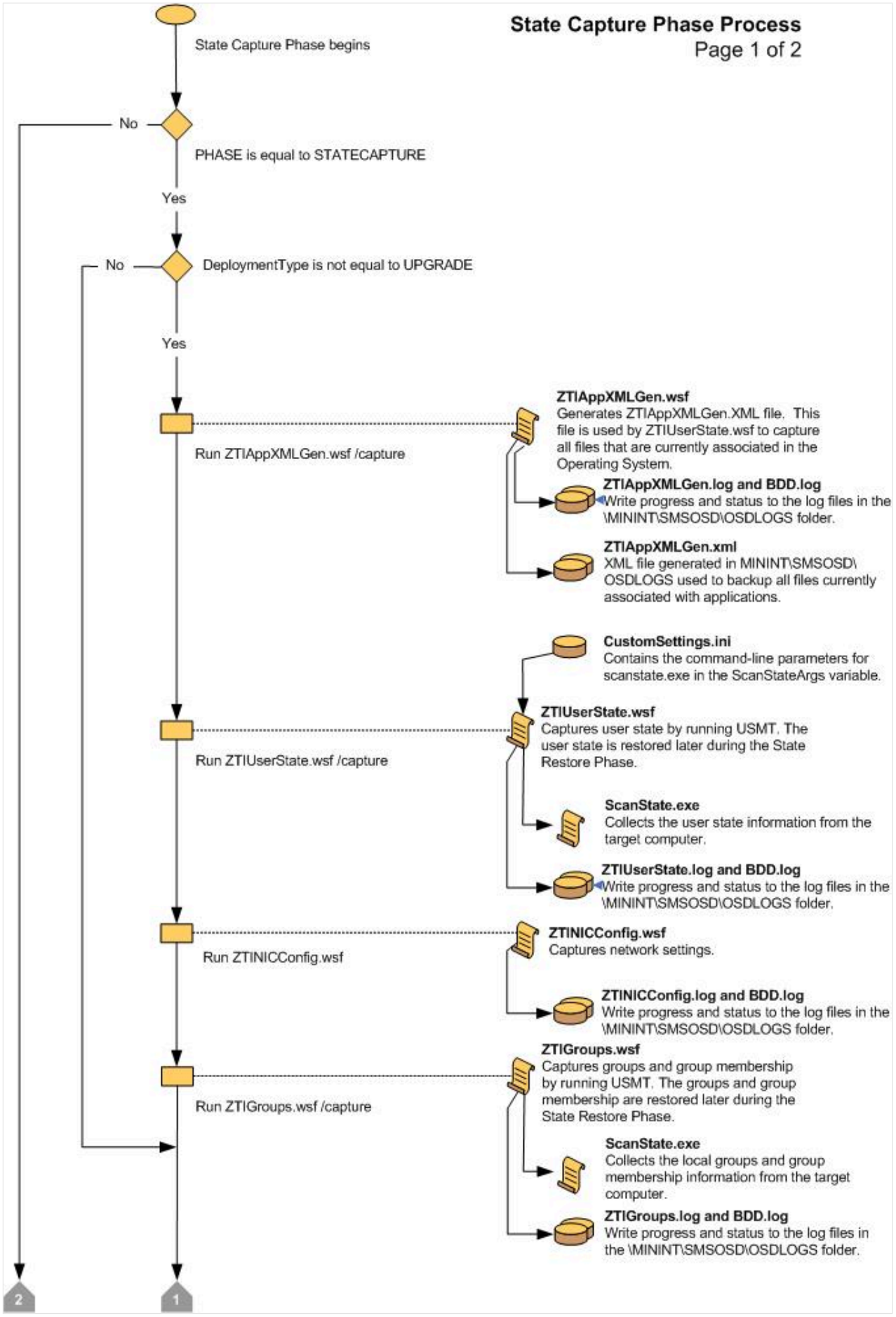


Figure 6. Flow chart for the State Capture Phase (2 of 2)

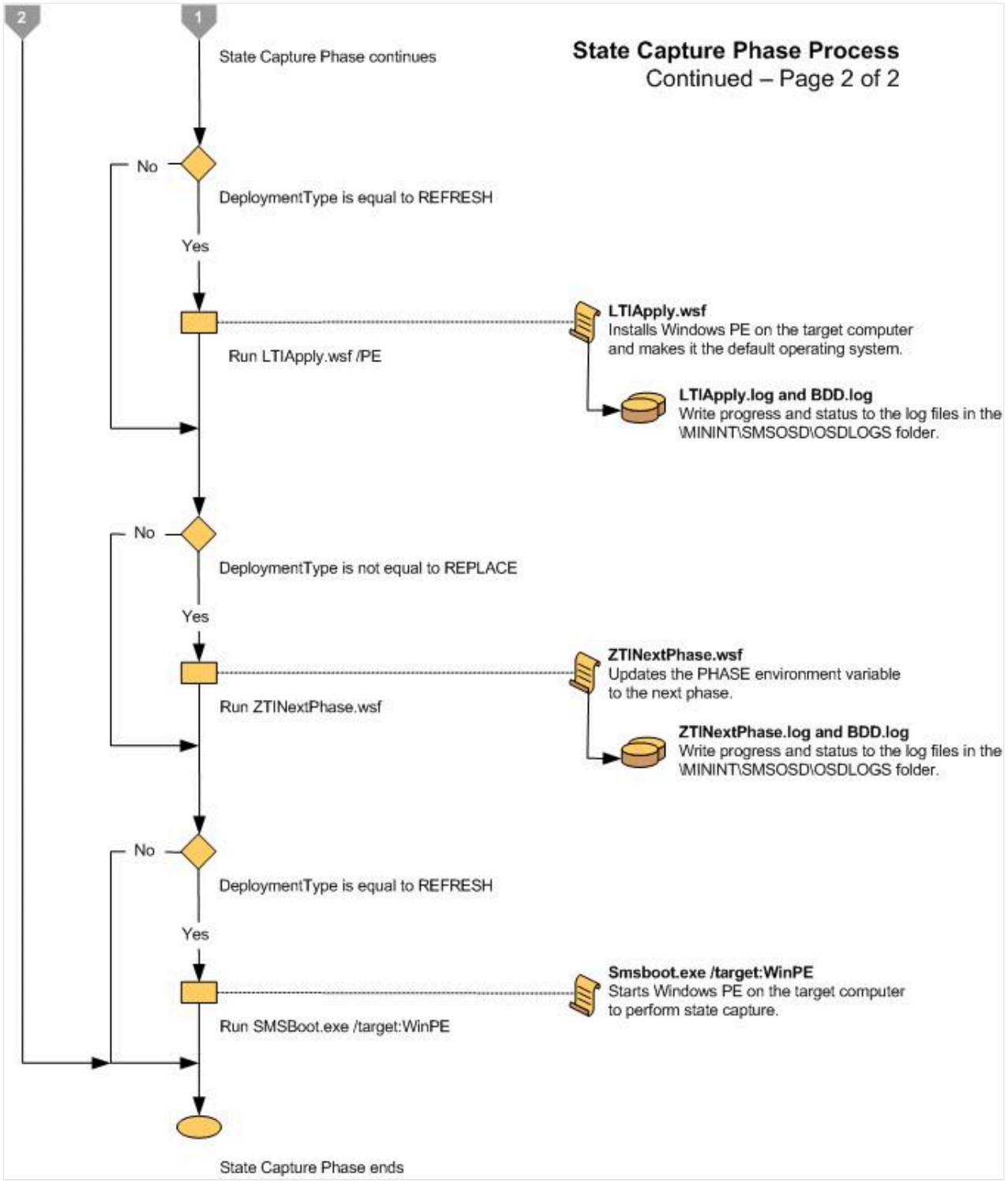


Figure 7. Flow chart for the Preinstall Phase (1 of 3)

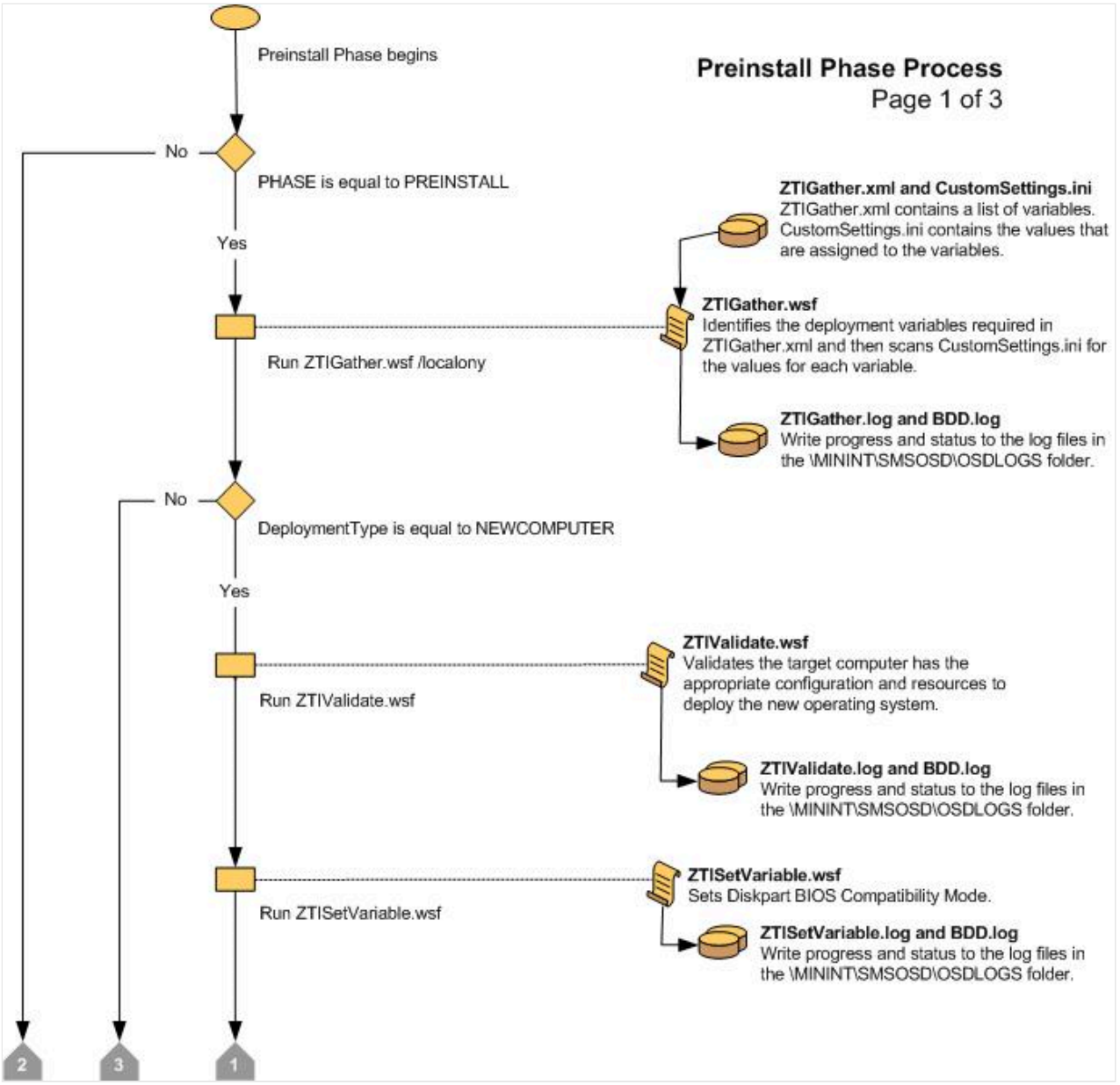


Figure 8. Flow chart for the Preinstall Phase (2 of 3)

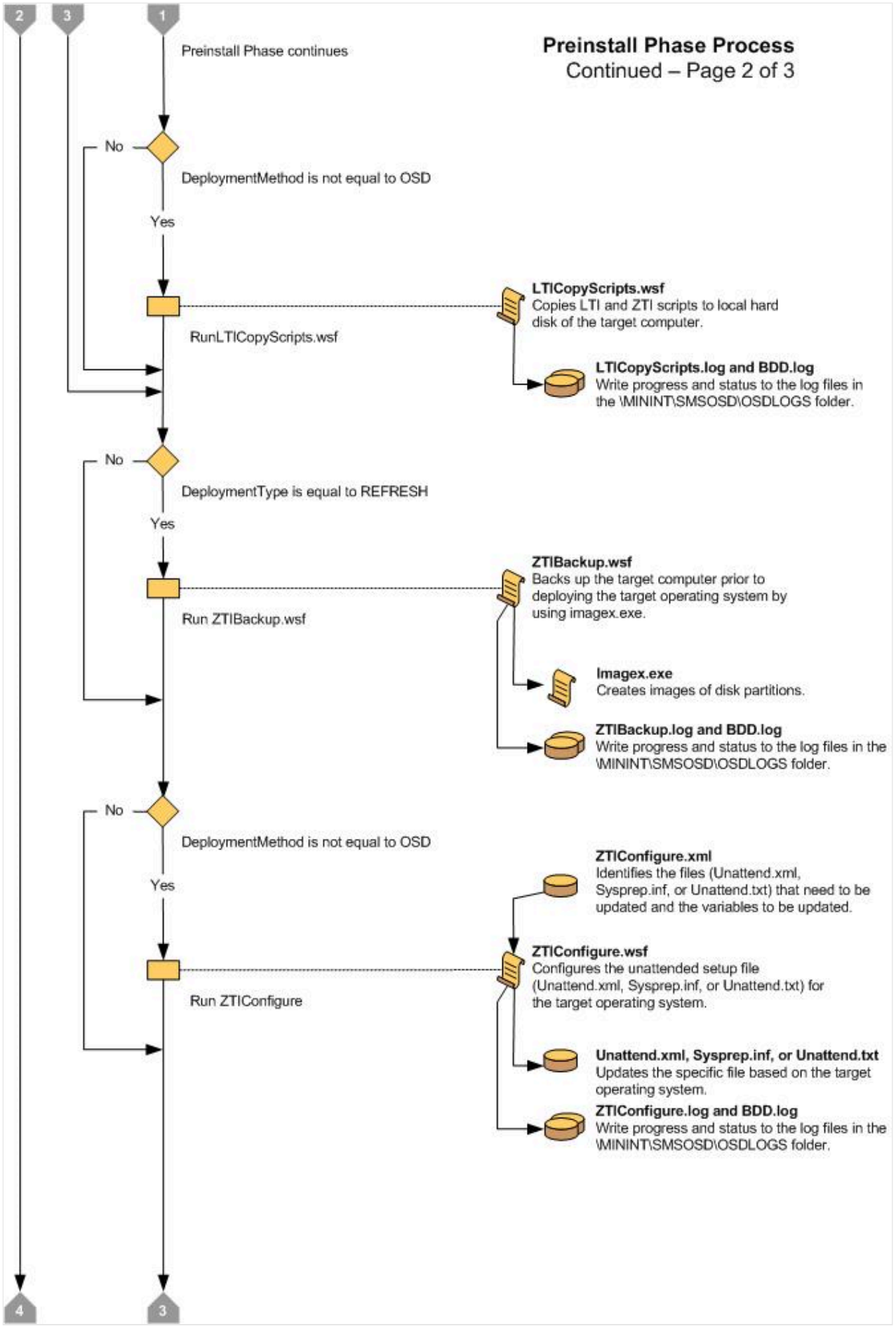


Figure 9. Flow chart for the Preinstall Phase (3 of 3)

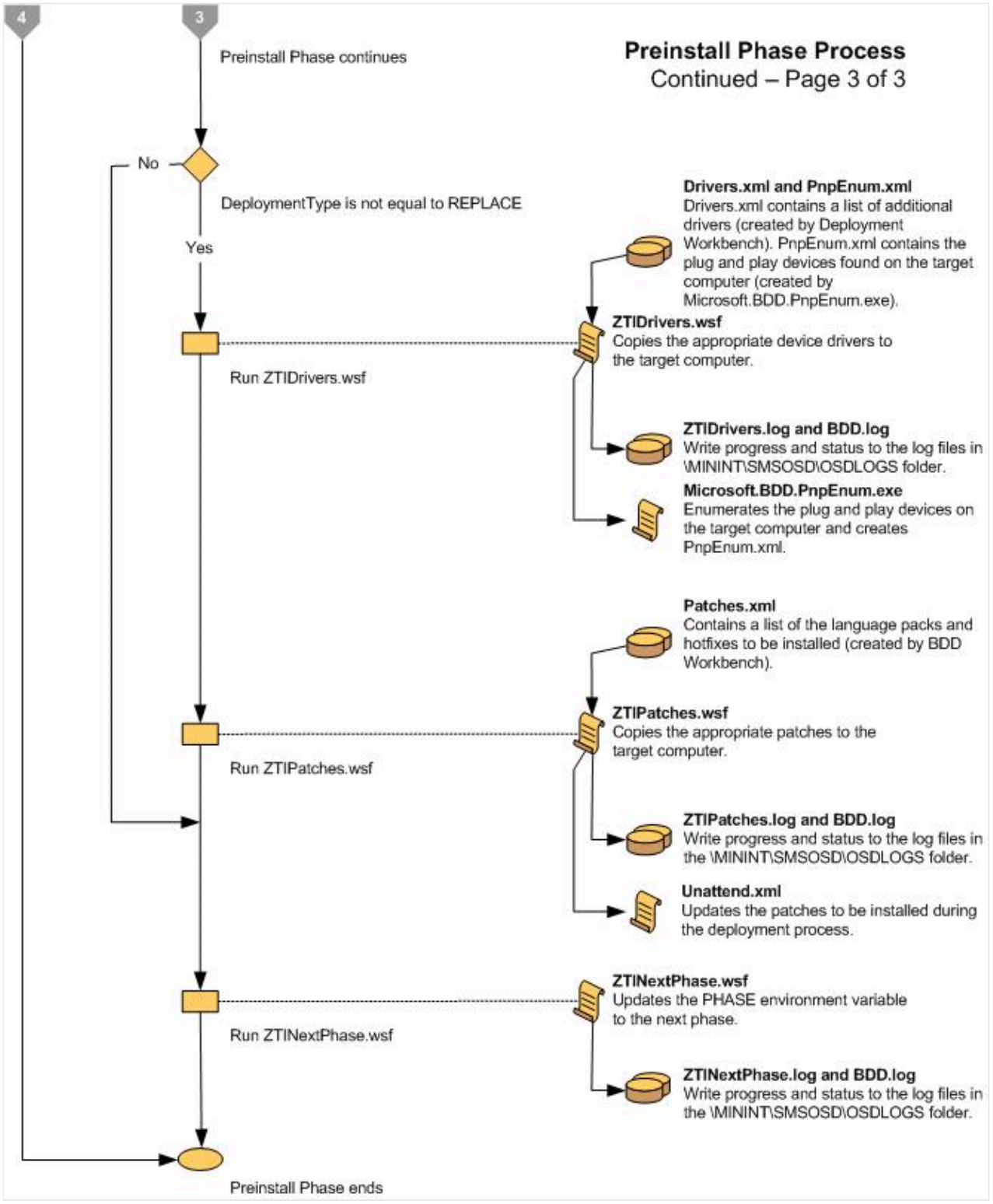


Figure 10. Flow chart for the Install Phase

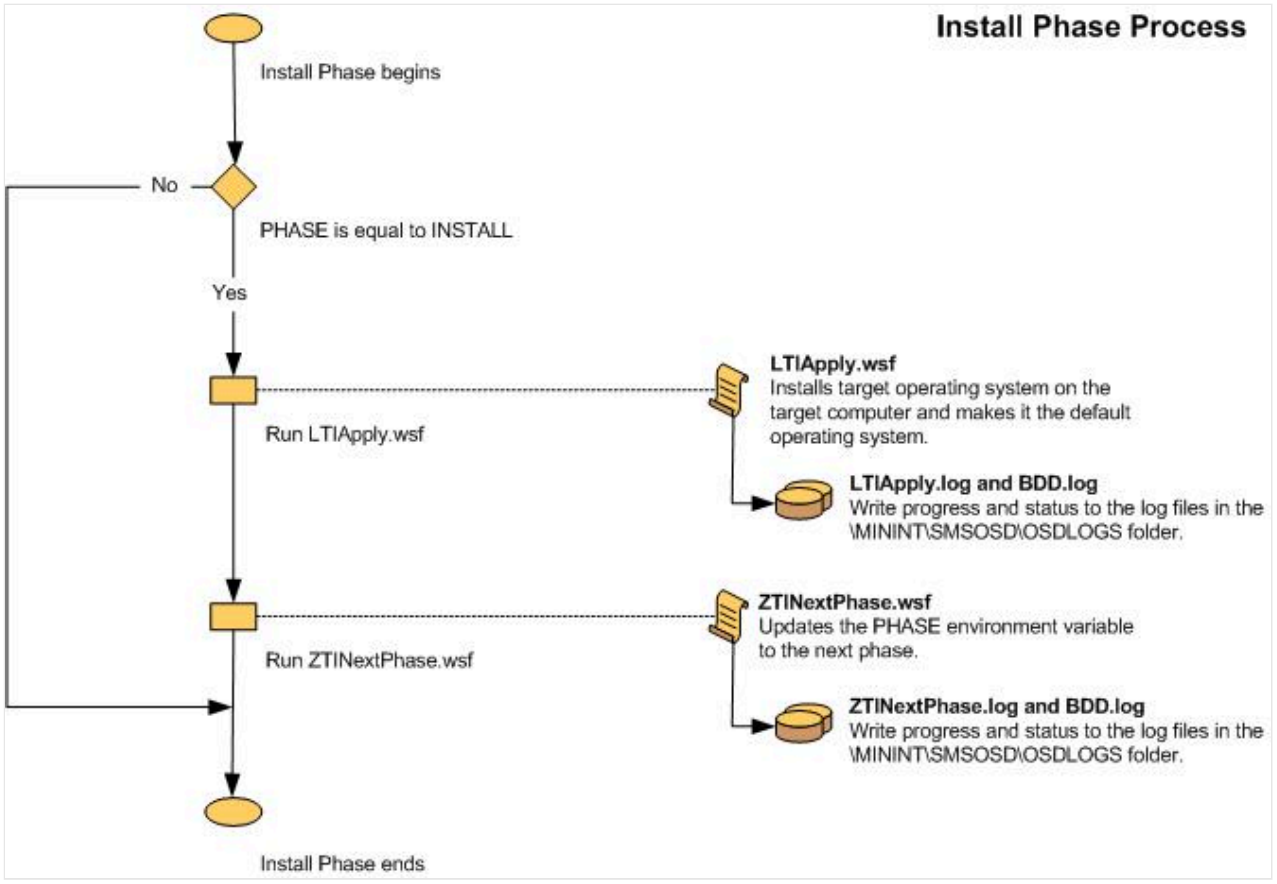


Figure 11. Flow chart for the Postinstall Phase (1 of 2)

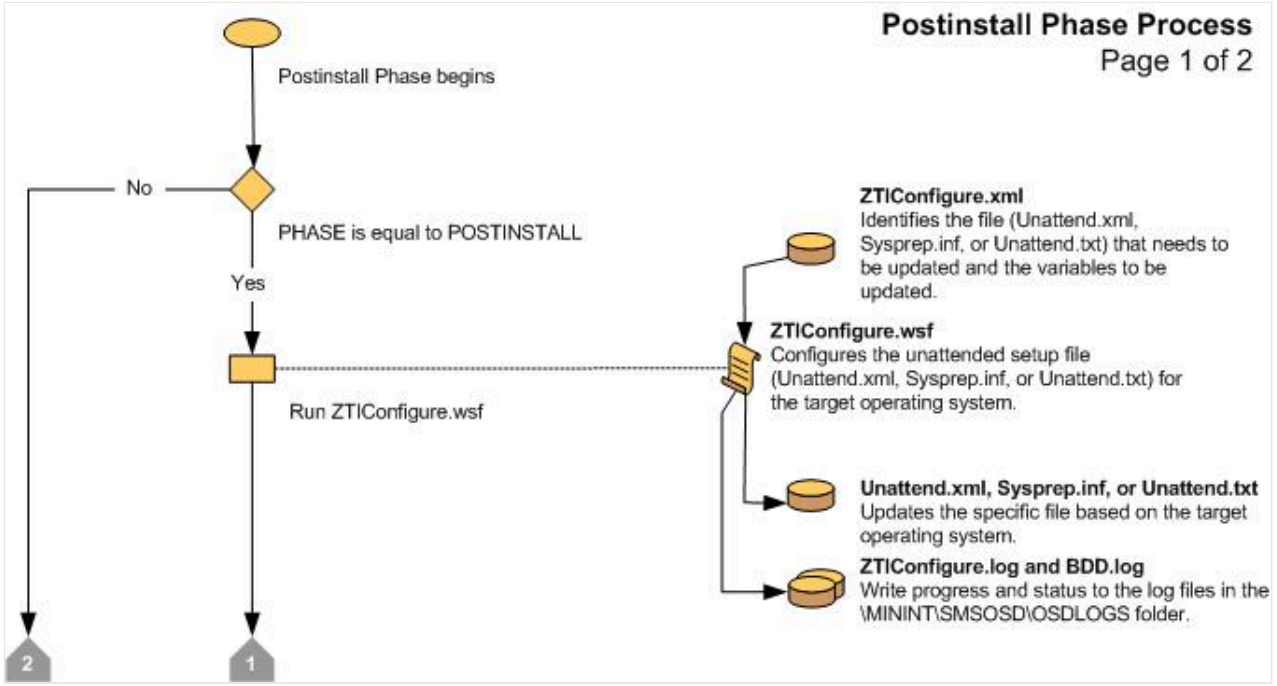


Figure 12 Flow chart for the Postinstall Phase (2 of 2)

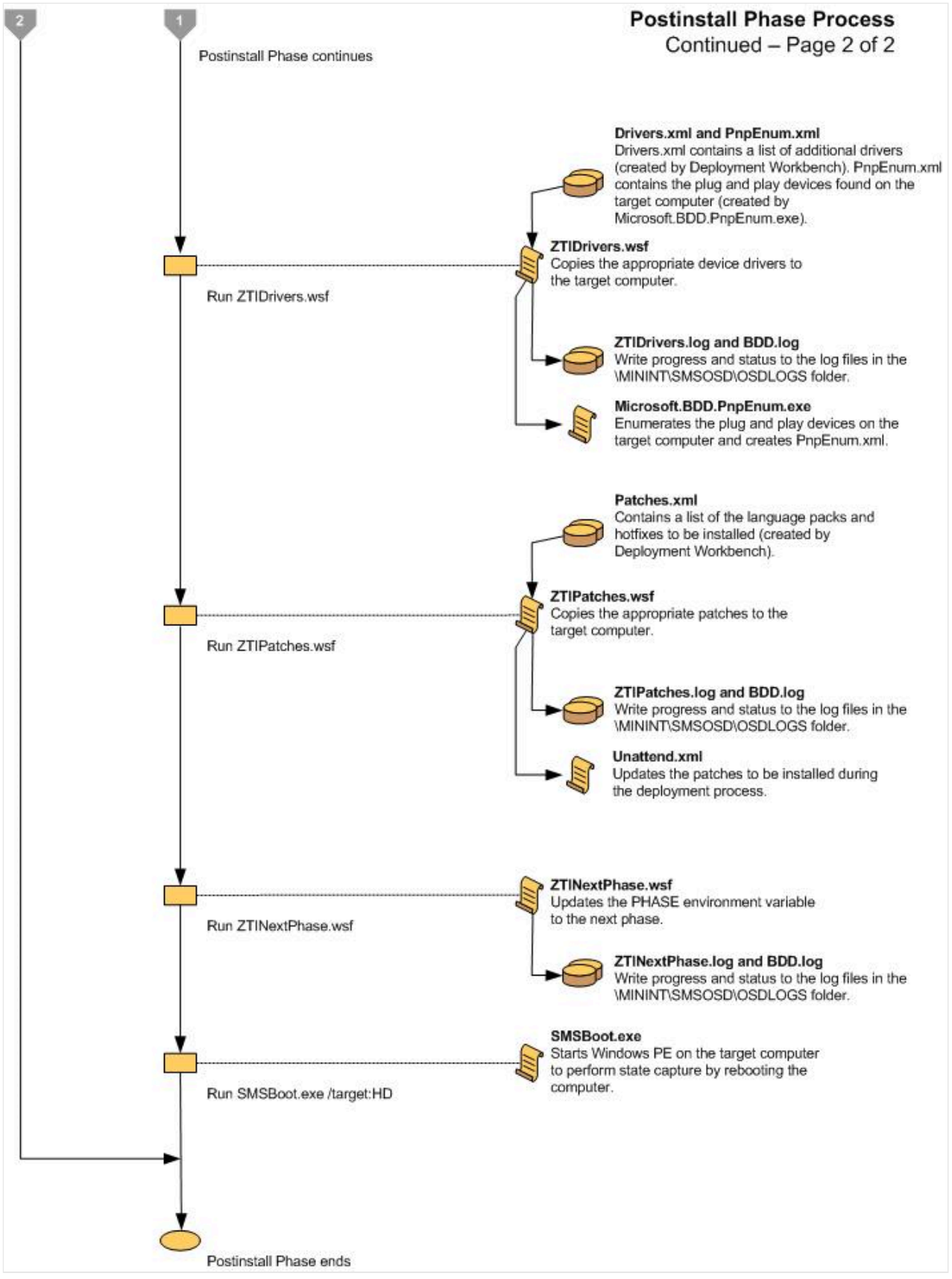


Figure 13. Flow chart for the State Restore Phase (1 of 4)

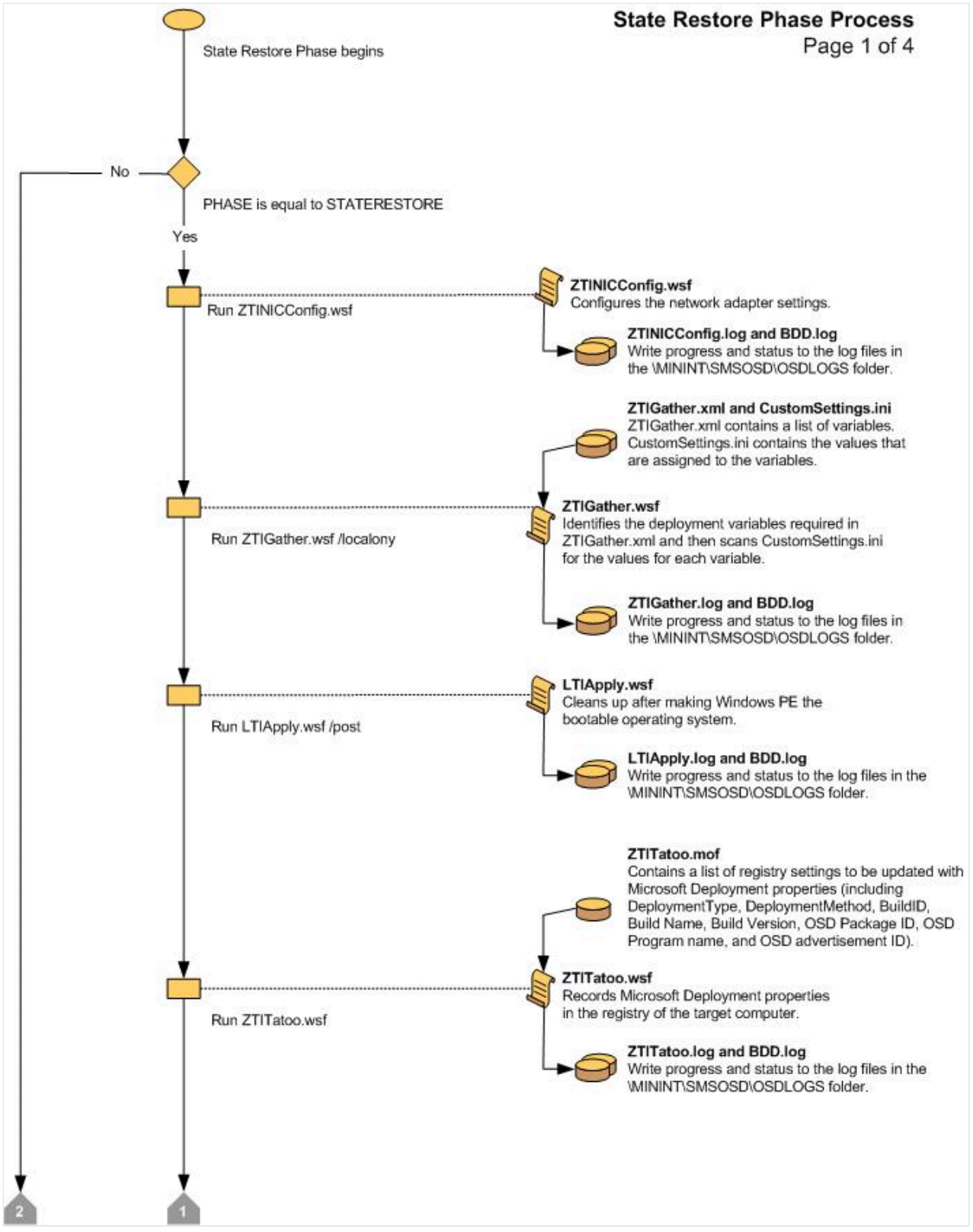


Figure 14. Flow chart for the State Restore Phase (2 of 4)

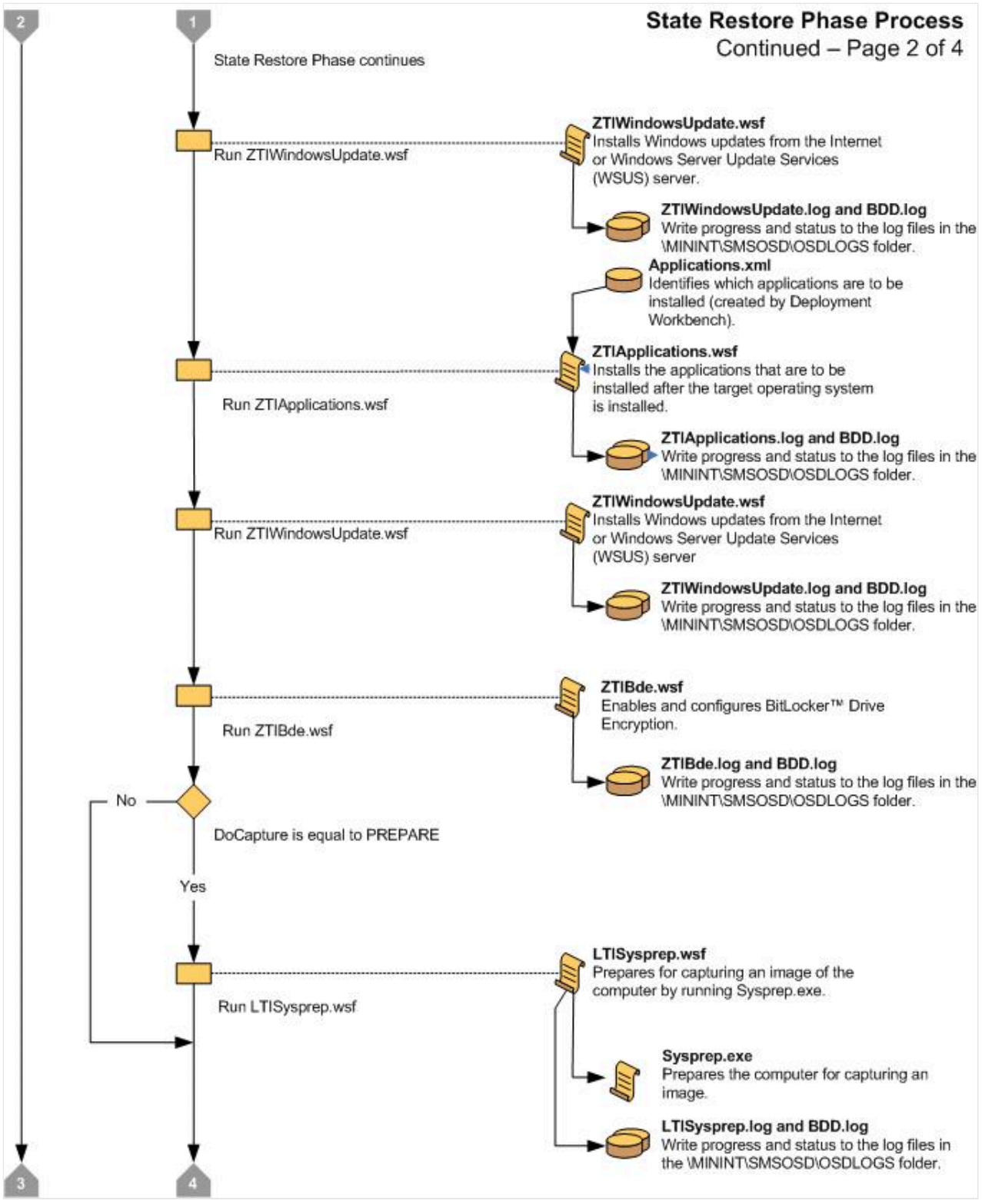


Figure 15. Flow chart for the State Restore Phase (3 of 4)

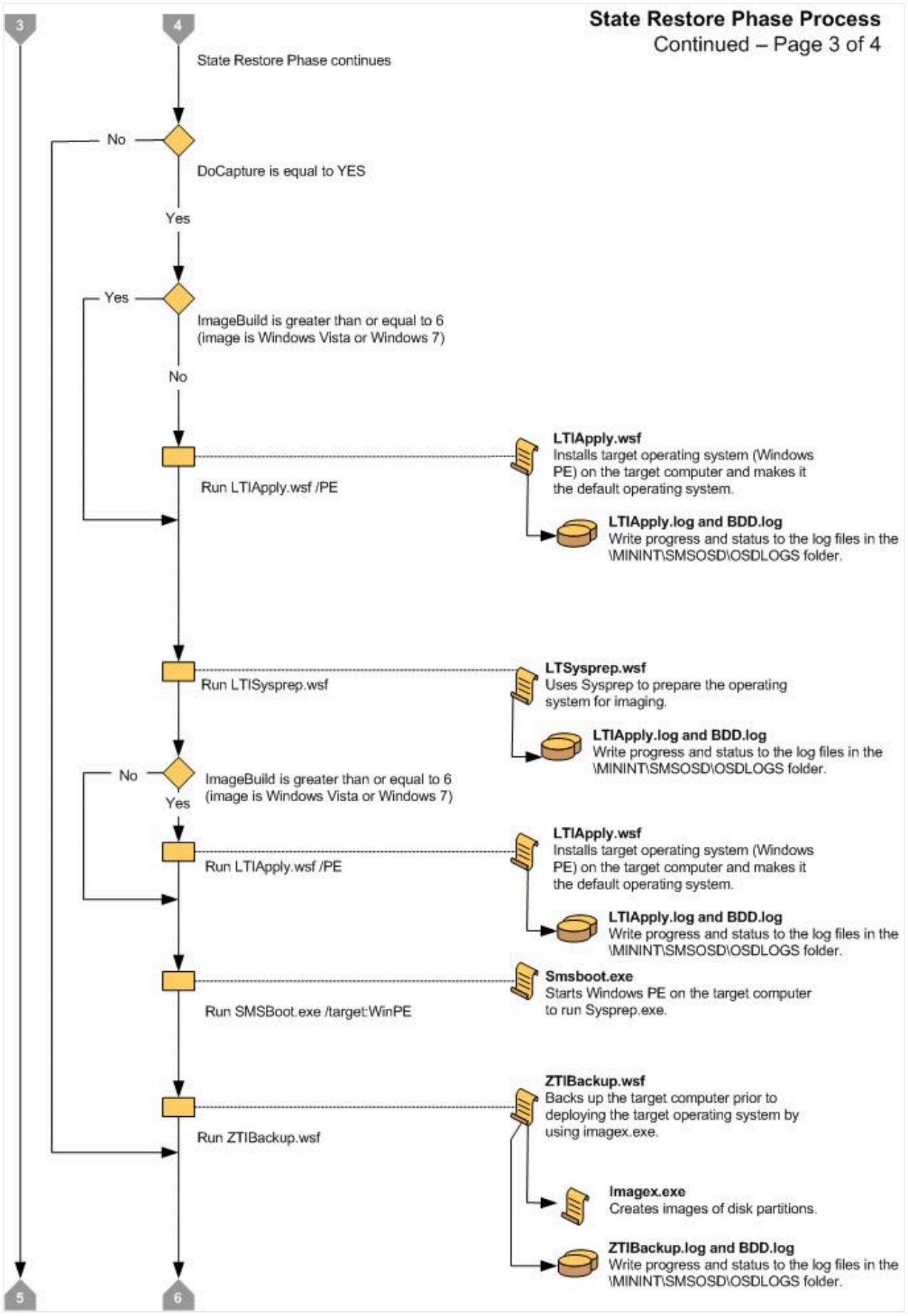
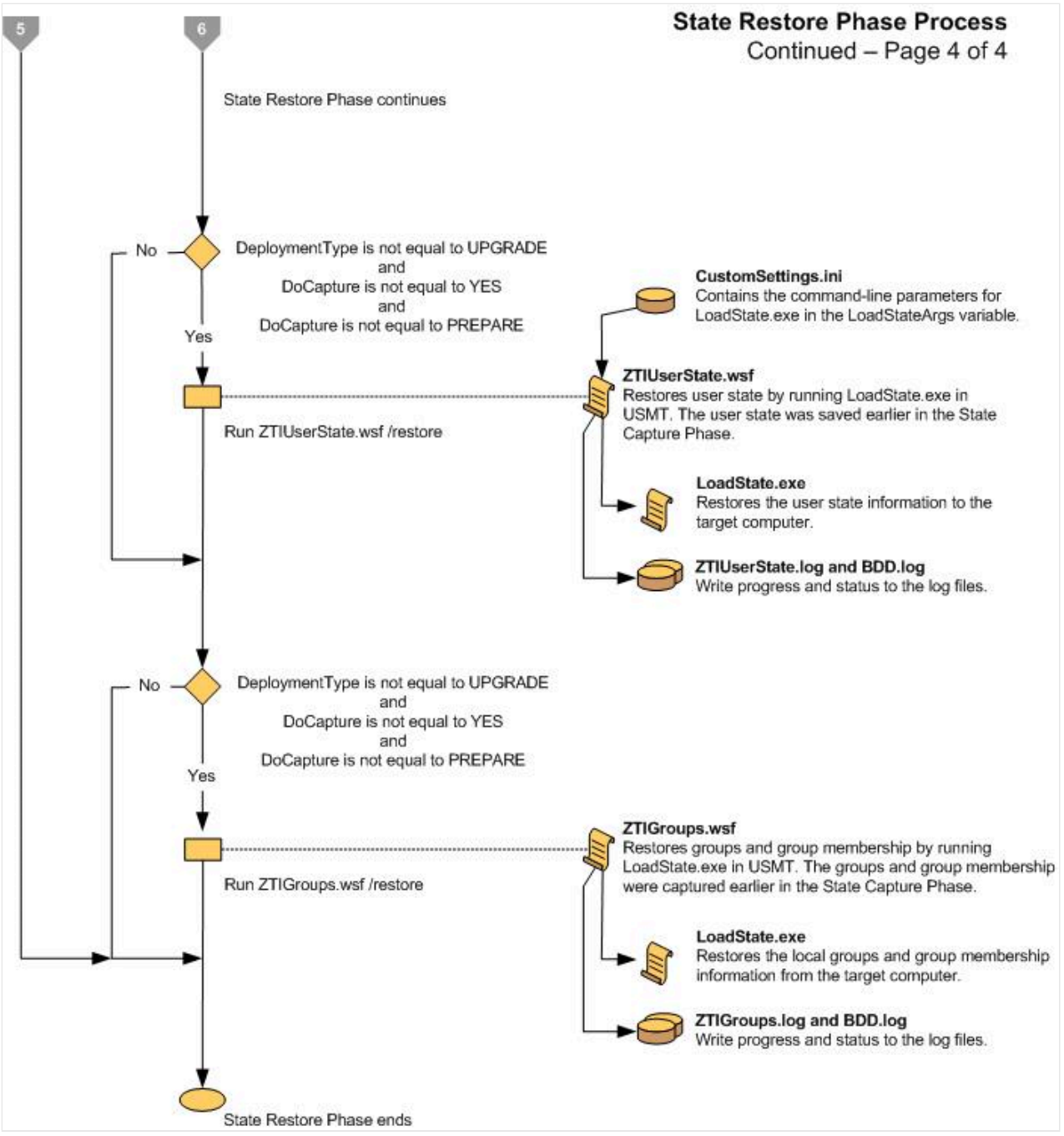


Figure 16. Flow chart for the State Restore Phase (4 of 4)



ZTI Deployment Process Flowcharts

Flow charts are provided for the following phases of ZTI deployment with Configuration Manager:

- Initialization (Figure 17)
- Validation (Figure 18)
- State Capture (Figure 19)
- Preinstall (Figure 20)
- Install (Figure 21)

- Postinstall (Figure 22)
- State Restore (Figure 23 and Figure 24)
- Capture (Figure 25)

Figure 17. Flow chart for the Initialization Phase

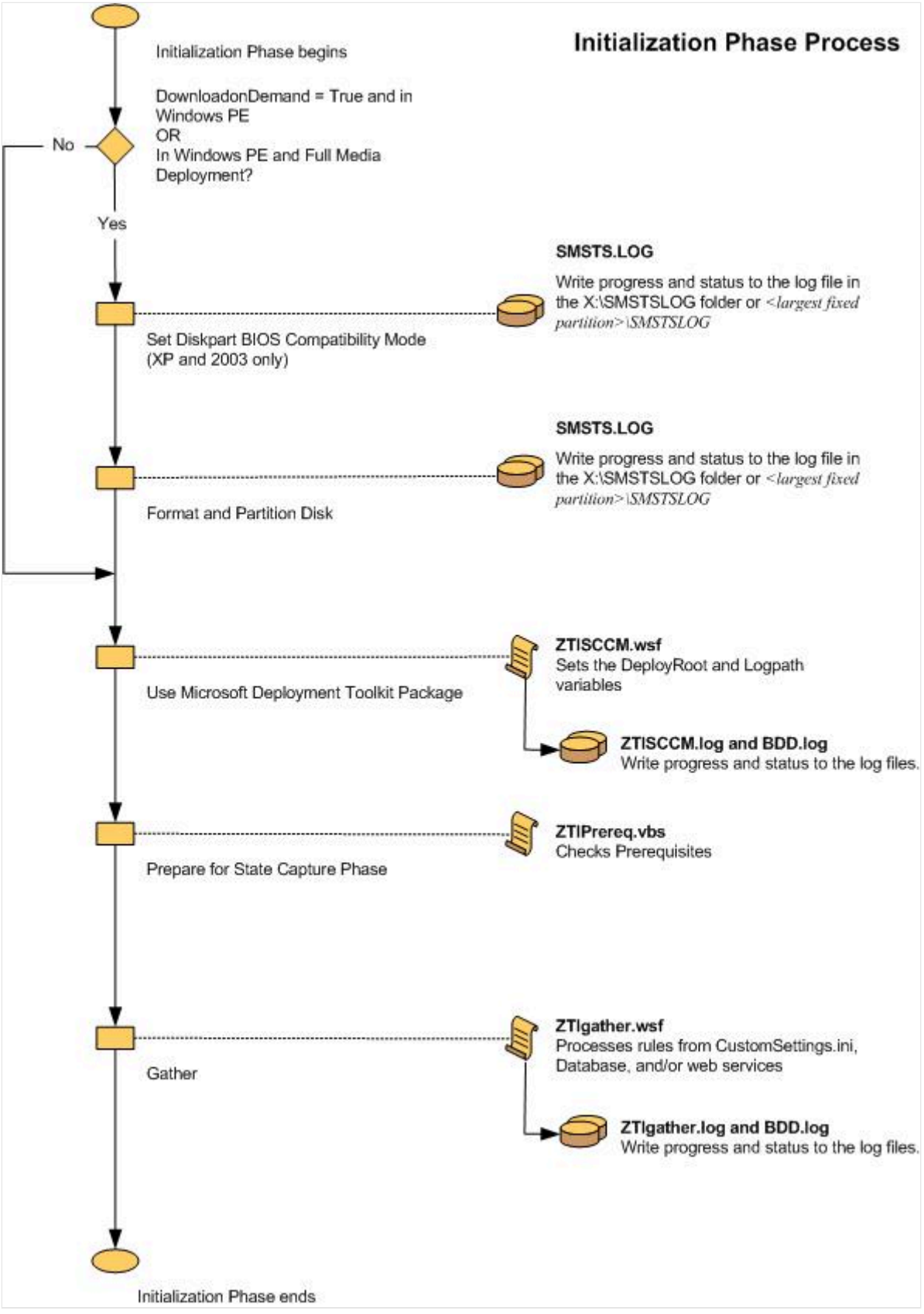


Figure 18. Flow chart for the Validation Phase

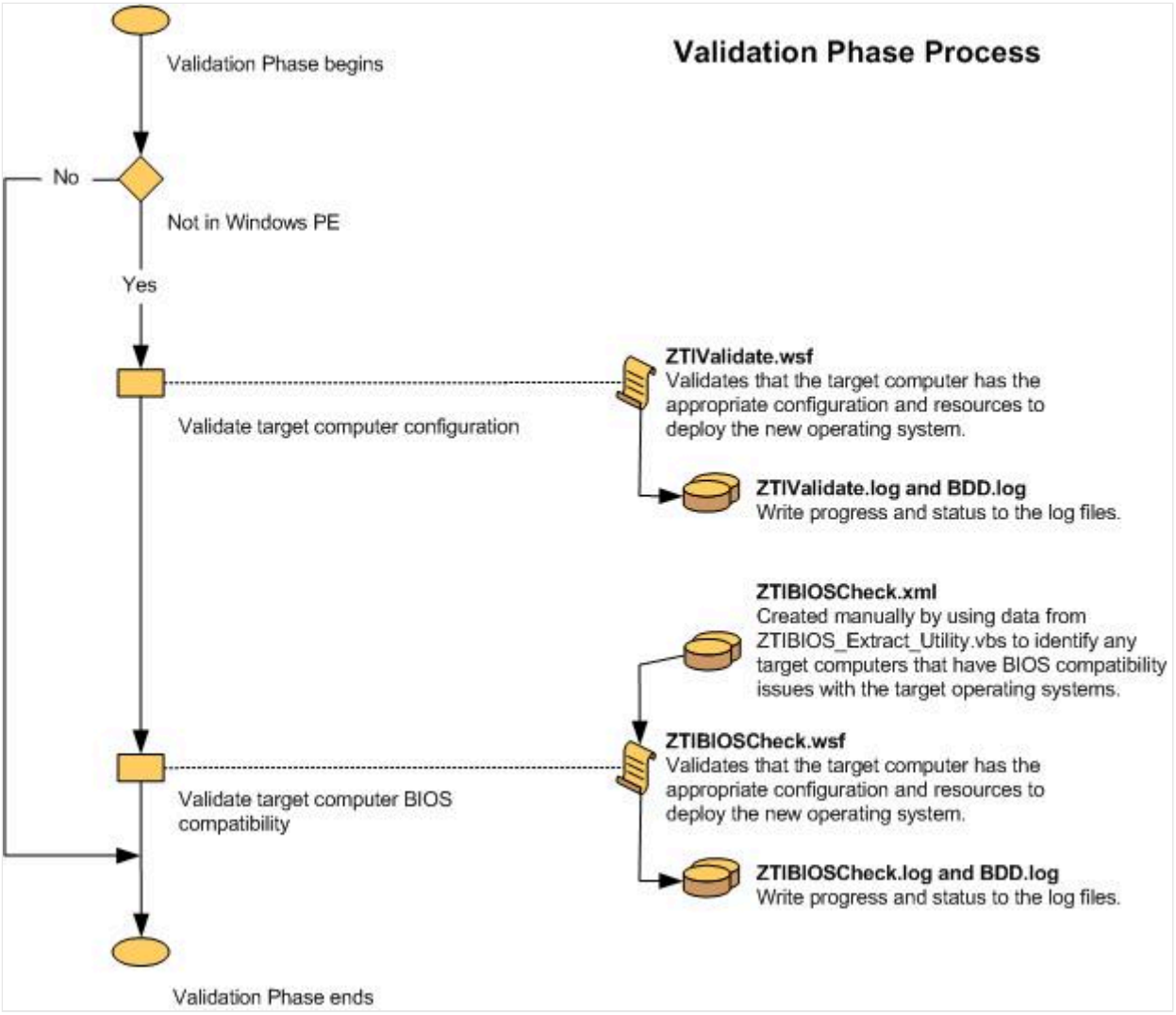


Figure 19. Flow chart for the State Capture Phase

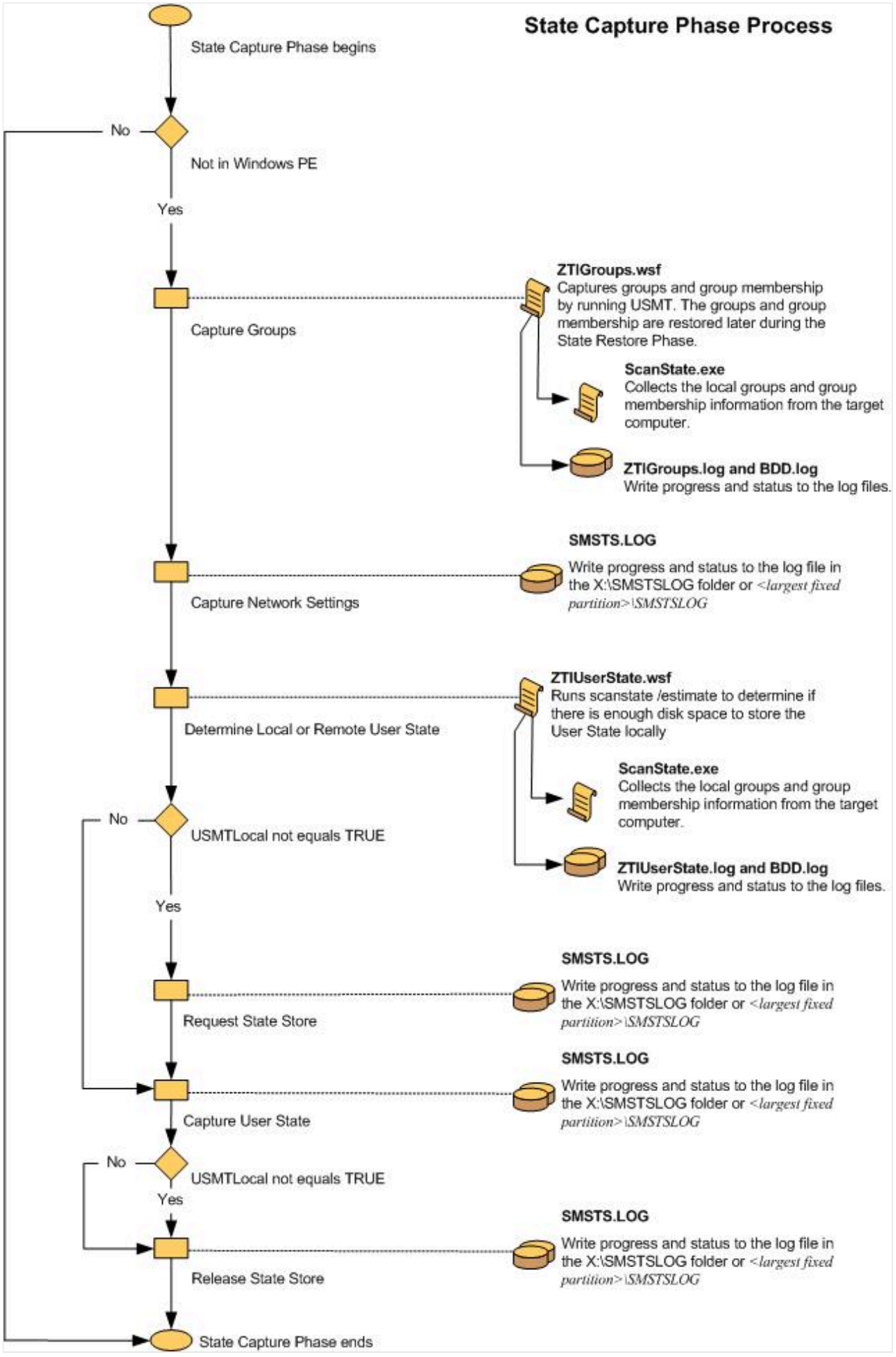


Figure 20. Flow chart for the Preinstall Phase

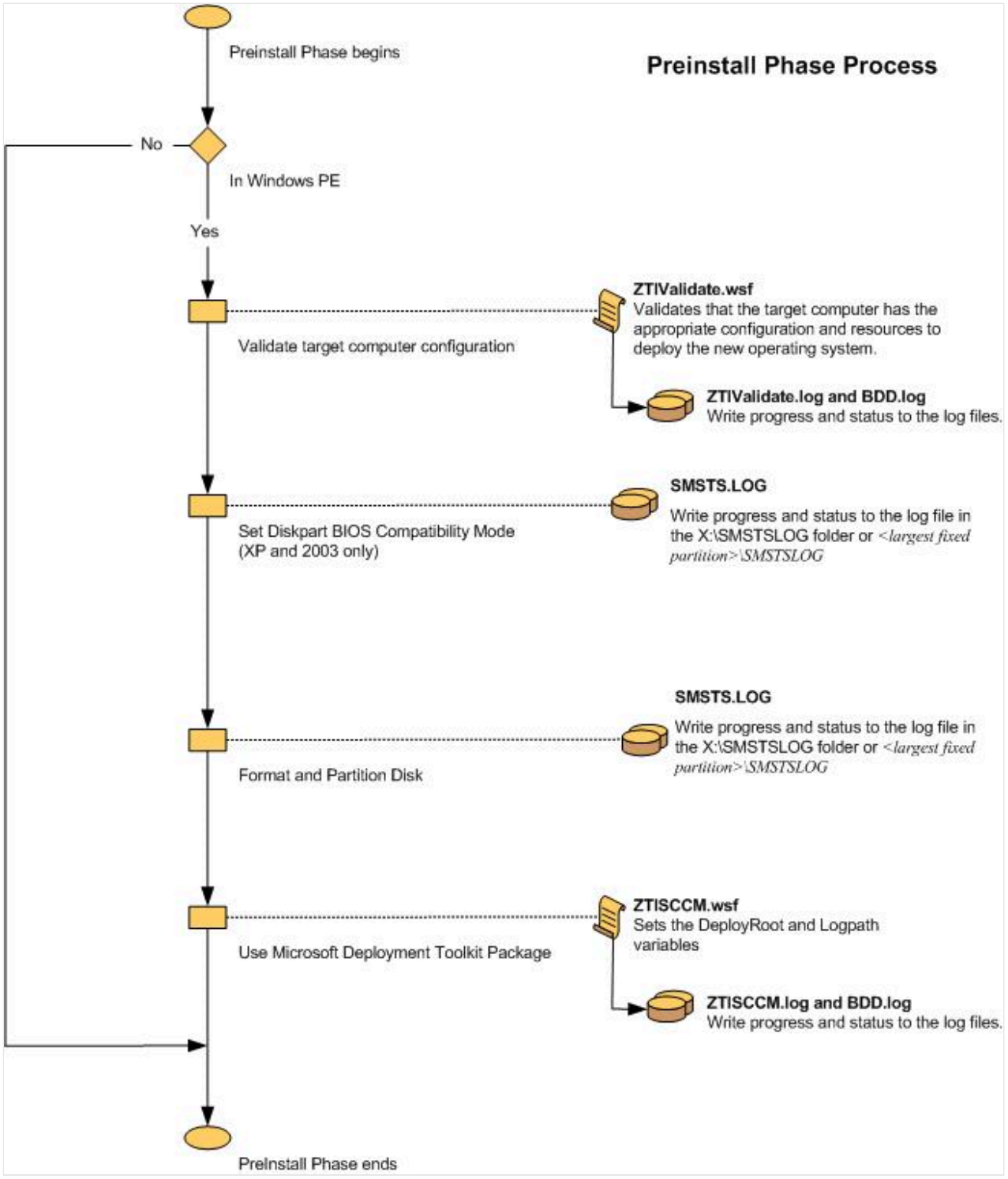


Figure 21. Flow chart for the Install Phase

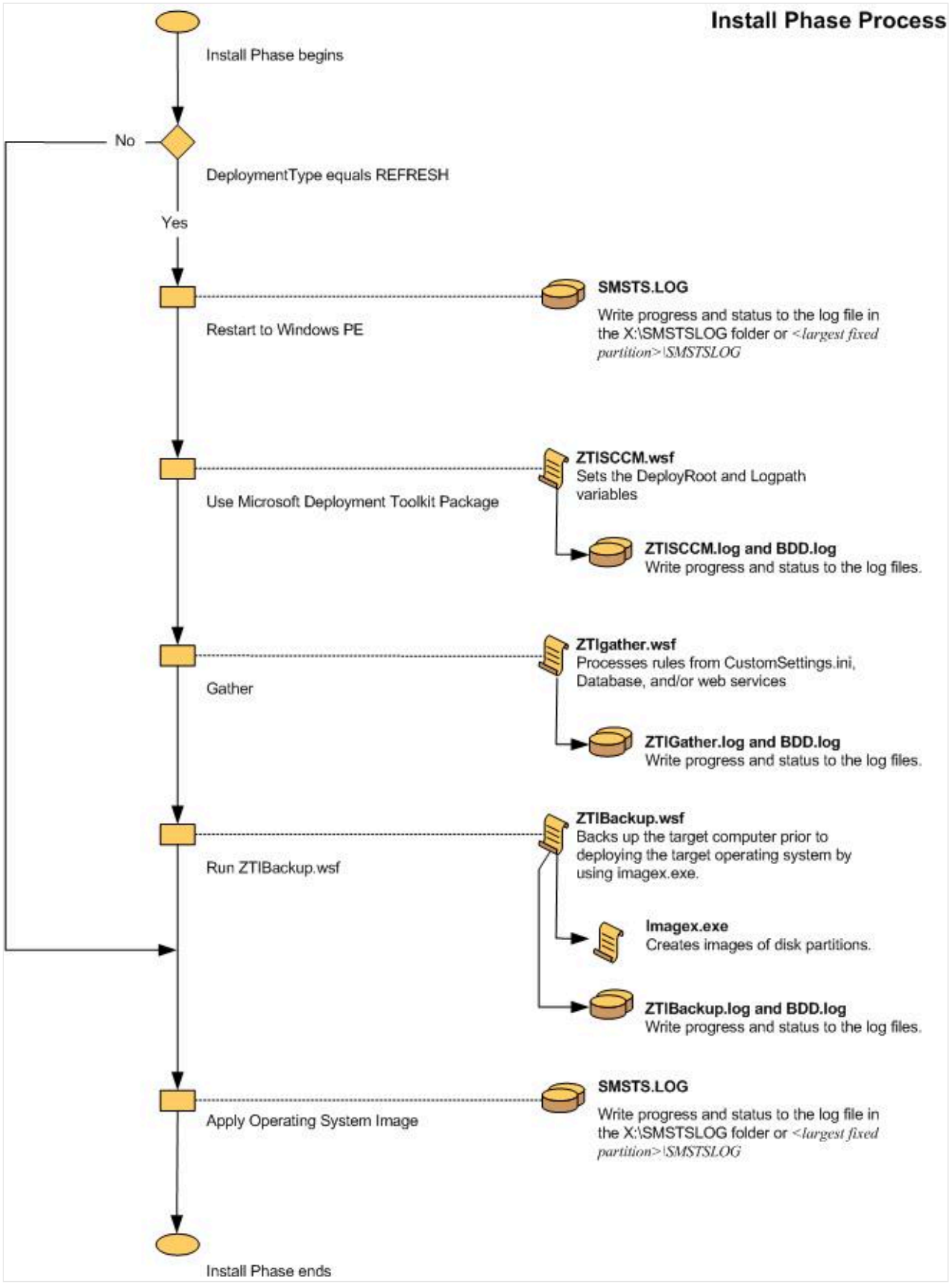


Figure 22. Flow chart for the Postinstall Phase

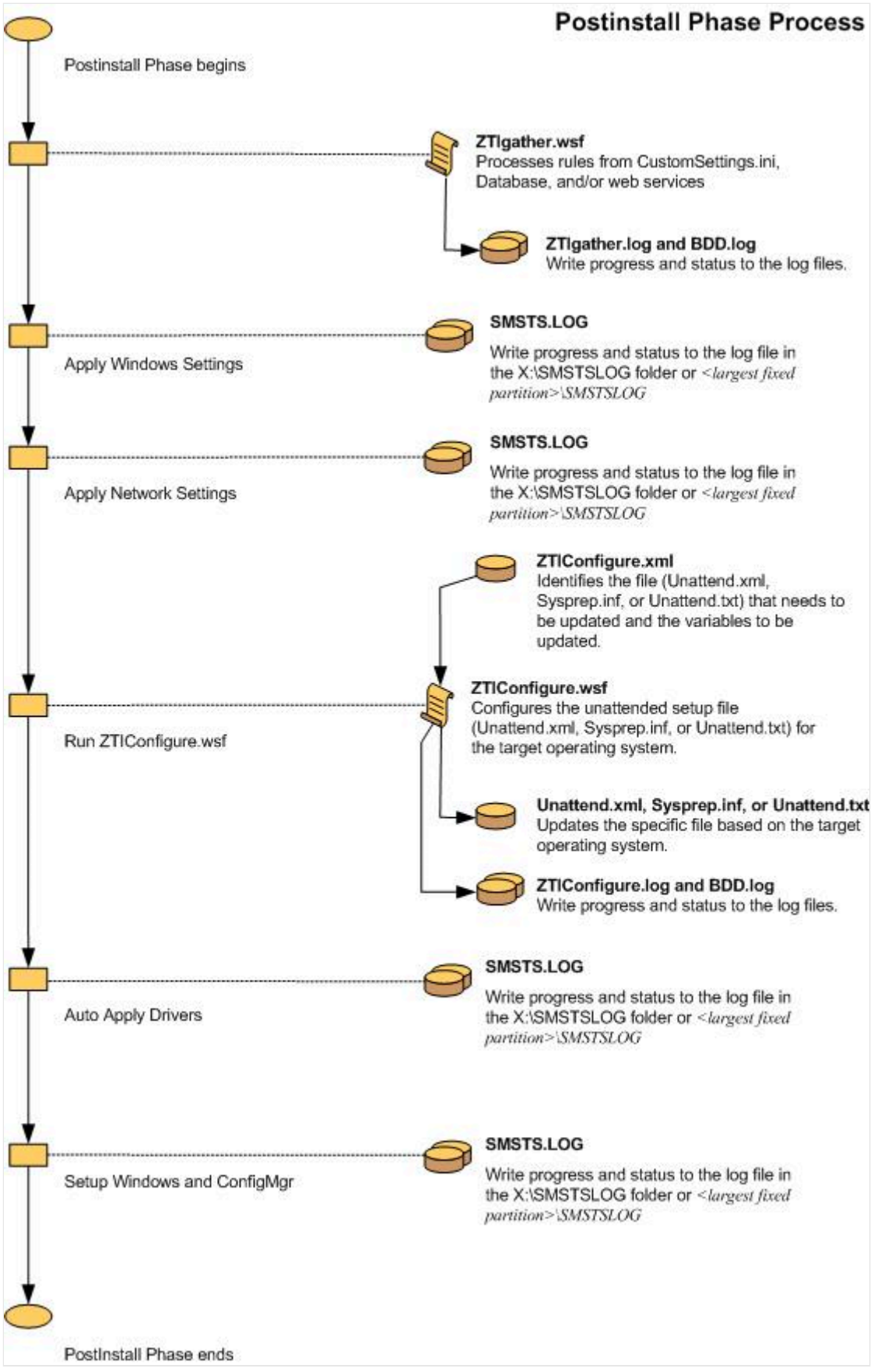


Figure 23. Flow chart for the State Restore Phase (1 of 2)

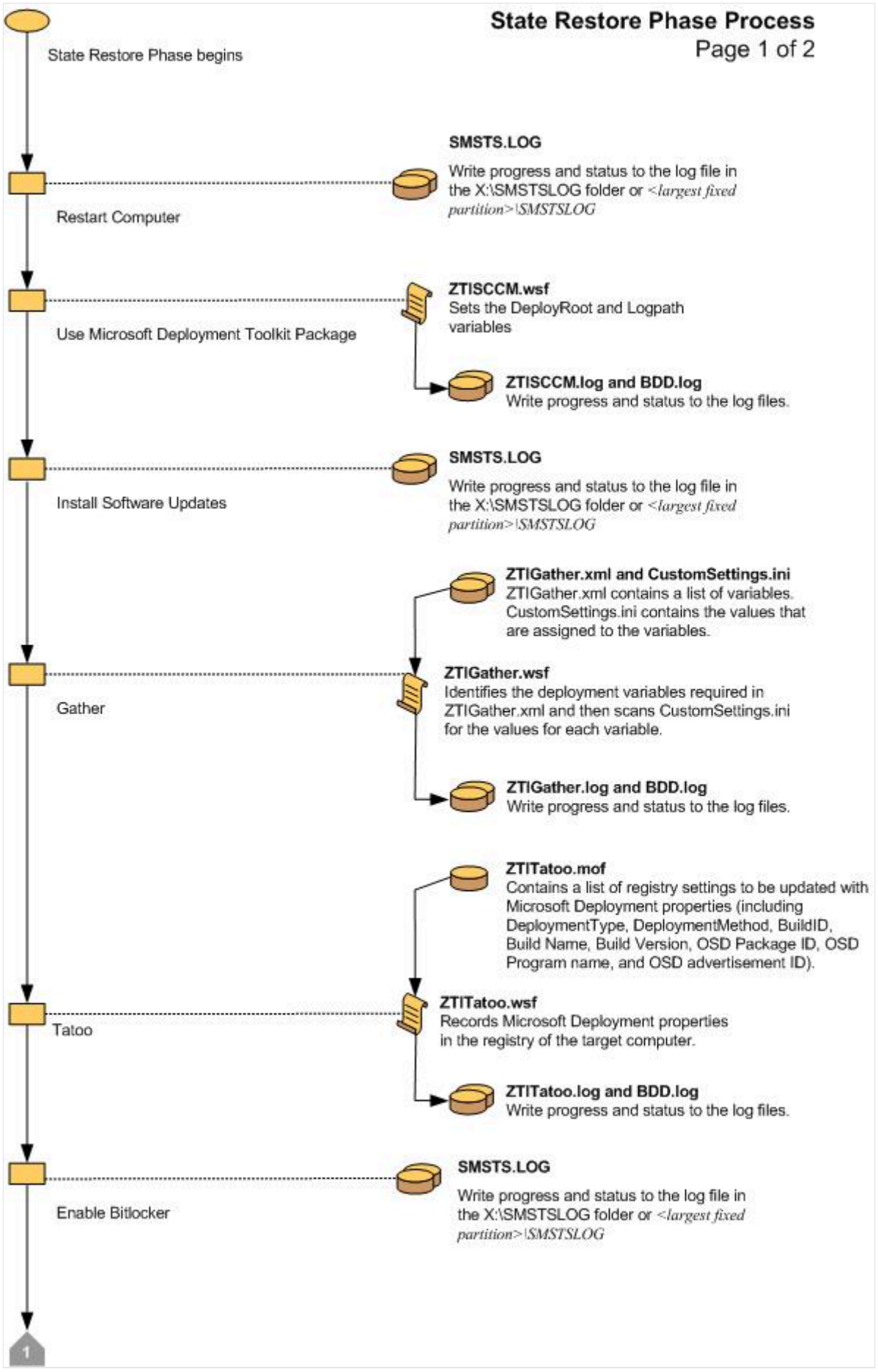


Figure 24. Flow chart for the State Restore Phase (2 of 2)

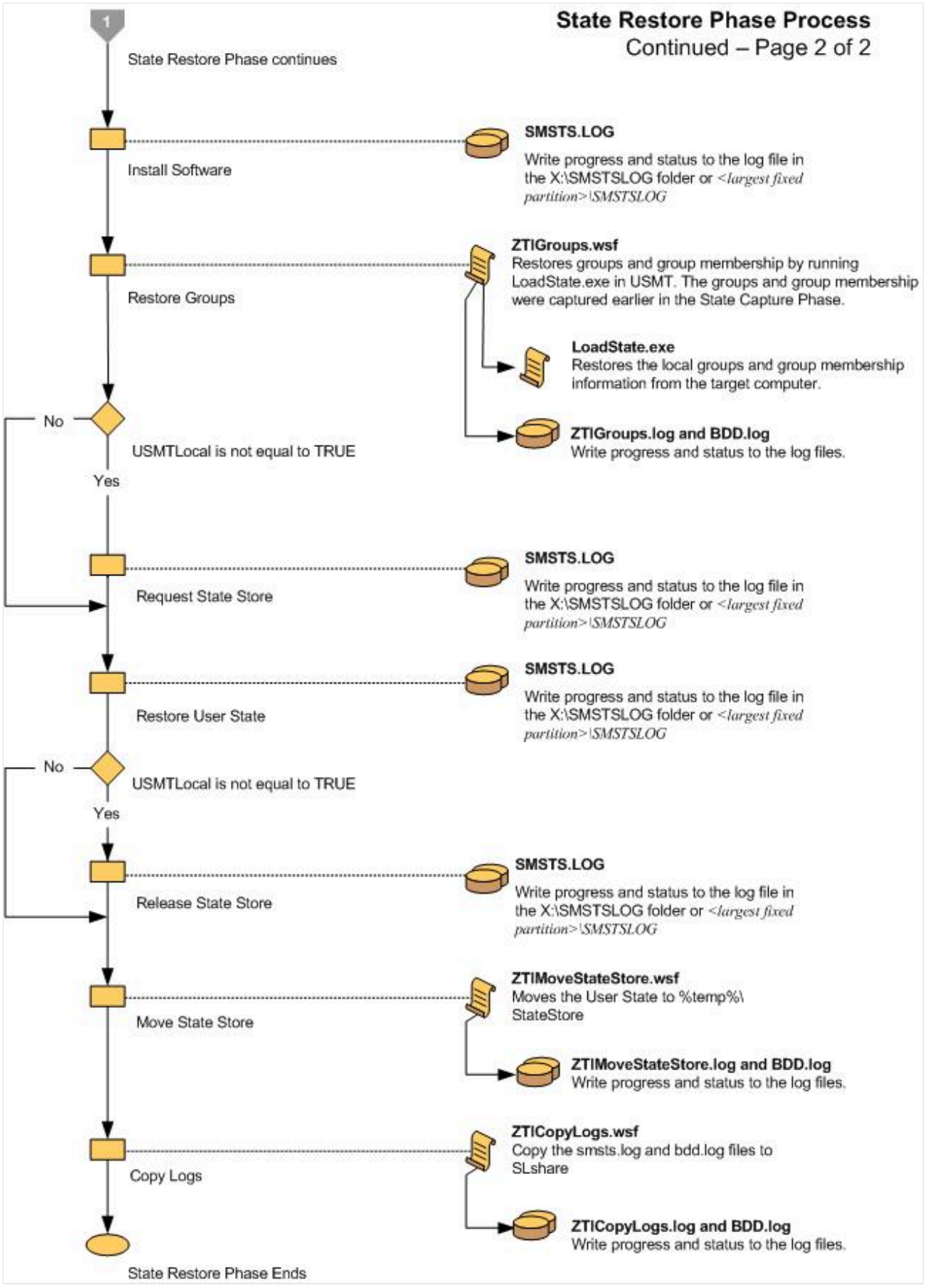
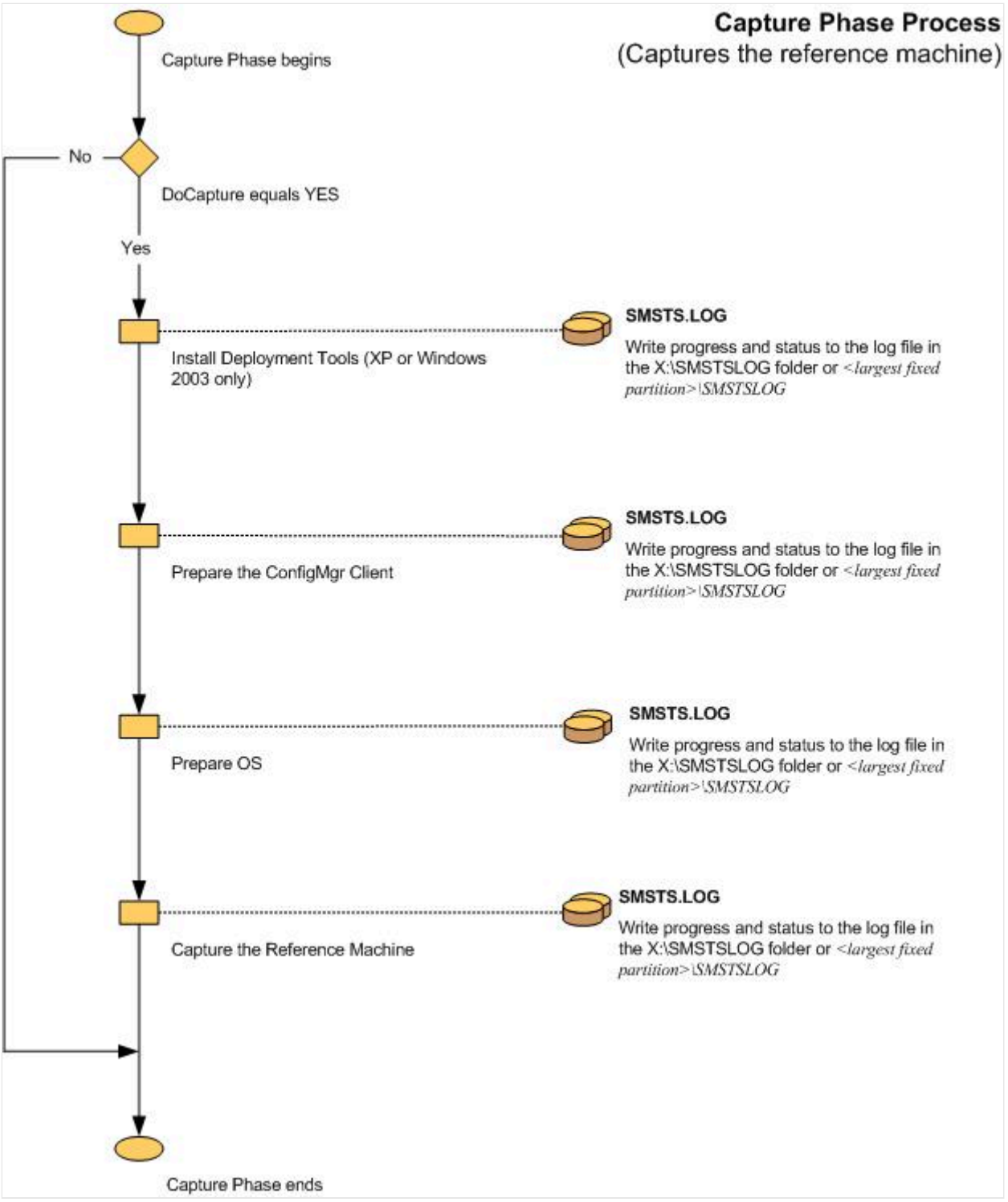


Figure 25. Flow chart for the Capture Phase



Microsoft Support

Microsoft provides Premier and Professional level support for Microsoft Deployment Toolkit.

- Professional level support
- Premier level support

> ⓘ **Note**

When contacting support, be clear that the issue is with MDT and the specific version.

User Driven Installation - Developers Guide

Article • 10/04/2022

User Driven Installation (UDI) helps simplify the deployment of Windows® client operating systems, such as Windows 8.1, to computers using the operating system deployment (OSD) feature in Microsoft® System Center 2012 R2 Configuration Manager. UDI is part of the Microsoft Deployment Toolkit (MDT).

Introduction

Typically, when deploying operating systems using the OSD feature, you must provide all the necessary information for deploying the operating system. The information is configured in configuration files or in databases (such as the CustomSettings.ini file or the MDT database [MDT DB]). You must provide all configuration settings before you can initiate the deployment.

UDI provides a wizard-driven interface that allows you to provide configuration information immediately prior to performing the deployment. This behavior allows you to create generic OSD task sequences, and then provide computer-specific information at the time of deployment, which provides greater flexibility in the deployment process.

Target Audience

This guide is written for the developers who create custom wizard pages for the UDI Wizard and custom wizard page editors for the UDI Wizard Designer. This guide assumes that you are familiar with the development of Windows applications using:

- C++, which is used to create custom wizard pages
- Microsoft .NET Framework, which is used to create custom wizard page editors
- Windows Presentation Foundation (WPF), which is used to create custom wizard page editors
- Languages that WPF supports, such as C#, C++, or Microsoft Visual Basic® .NET, which are used to create custom wizard page editors

About This Guide

This guide provides the necessary reference information to help you customize UTI for your organization. This guide does not discuss administrative or operational topics, such as installing MDT (which includes UDI), configuring UDI to deploy operating systems and applications, or performing deployments using the UDI Wizard. For more information on those topics, see the UDI topics in *Using the Microsoft Deployment Toolkit*, which is included with MDT.

UDI Development Overview

UDI development allows you extend the features that UDI provides. Typically, UDI development is required when you want to collect additional information that the UDI deployment process consumes. This additional information is usually saved as task sequence variables that task sequence steps in a UDI task sequence in Configuration Manager read.

UDI Architecture

The high-level goal of UDI development is to create custom wizard pages that can be displayed in the UDI Wizard. By creating custom wizard pages, you can extend the existing features of UDI to meet the business and technical requirements of your organization. A custom wizard page collects information in addition to or in place of the wizard pages that UDI provides.

Figure 1 illustrates the relationship between the UDI Wizard Designer and the UDI Wizard.

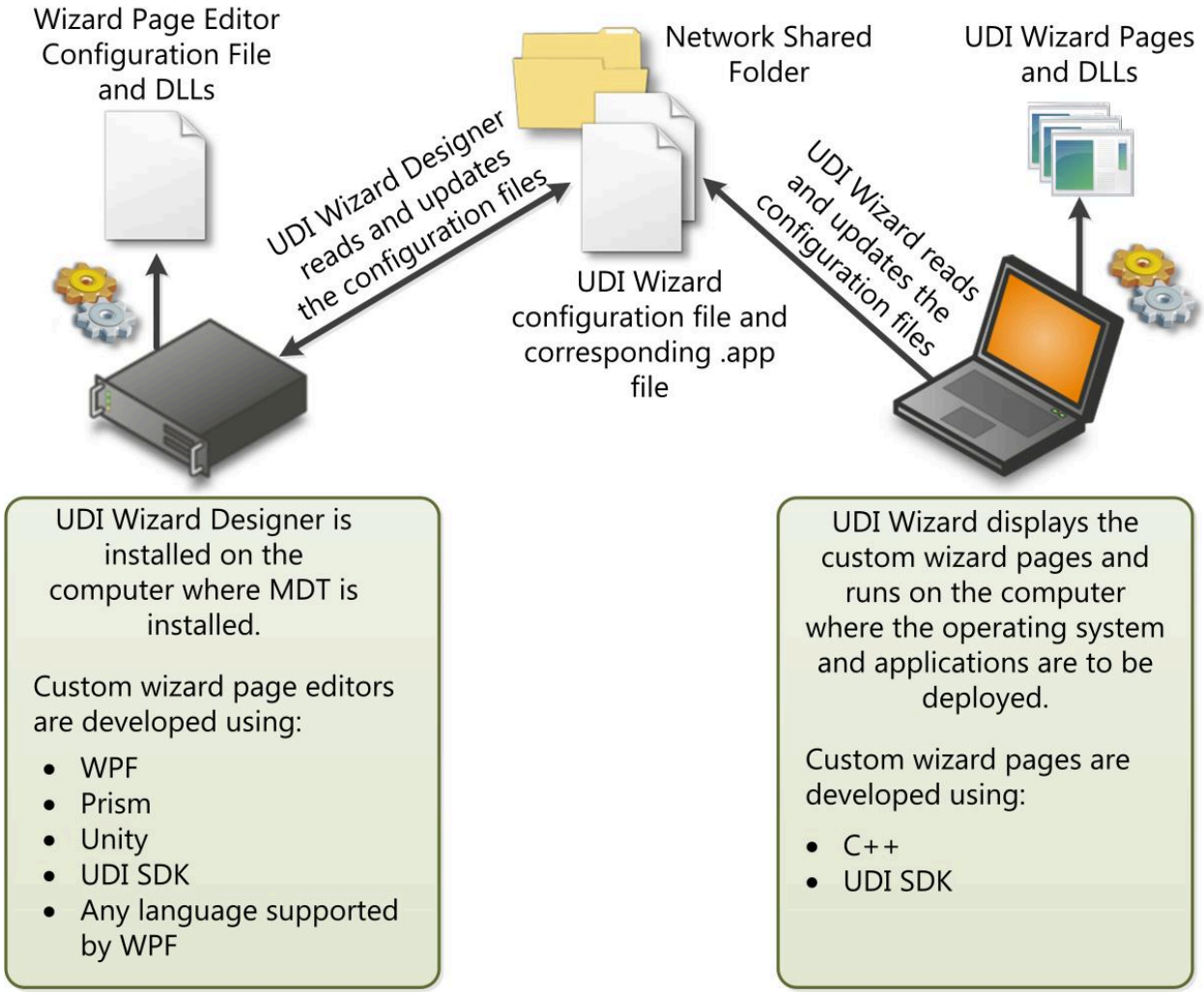


Figure 1. Relationship between the UDI Wizard and UDI Wizard Designer

Figure 1. Relationship between the UDI Wizard and UDI Wizard Designer

At a conceptual level, UDI development includes the creation of:

- **Custom wizard pages**. Wizard pages are displayed in the UDI Wizard and collect the information required to complete the deployment process. You create wizard pages using C++ in Microsoft Visual Studio®. The custom wizard pages are implemented as DLLs that the UDI Wizard reads. The UDI software development kit (SDK) includes an example of how to create custom wizard pages.

- **Custom wizard page editors**. You use wizard page editors to configure the behavior of your custom wizard page. The custom wizard page editors are implemented as DLLs that the UDI Wizard Designer reads. You create wizard page editors using:

  - WPF version 4.0

  - Microsoft Prism version 4.0

  - Microsoft Unity Application Block (Unity) version 2.1

MDT includes all the assemblies necessary to create a custom wizard page editor for use in the UDI Wizard Designer. The UDI SDK includes an example of how to create custom wizard page editors.

In addition, the UDI Wizard Designer consumes supporting wizard page editor configuration files. You create the wizard page editor configuration files as a part of the process for creating your custom wizard pages and custom wizard page editors. The UDI Wizard Designer creates the necessary XML information in the UDI Wizard configuration file and corresponding .app file.

Preparing the UDI Development Environment

Before you begin creating your own custom wizard pages and wizard page editors, perform the following steps to prepare the UDI development environment:

1. Prepare the UDI development environment prerequisites as described in Prepare the UDI Development Environment Prerequisites.
2. Configure the UDI development environment as described in Configure the UDI Development Environment.
3. Verify that the UDI development environment is configured correctly as described in Verify the UDI Development Environment.

Prepare the UDI Development Environment Prerequisites

To prepare the UDI development environment prerequisites, perform the following steps:

1. Prepare the UDI development environment hardware perquisites as described in Prepare the UDI Development Environment Hardware Prerequisites.
2. Prepare the UDI Development environment software perquisites as described in Prepare the UDI Development Environment Software Prerequisites.

Prepare the UDI Development Environment Hardware Prerequisites

The UDI development environment hardware prerequisites are the same hardware requirements for the edition of Microsoft Visual Studio you are using. For more information about these requirements, see the system requirements for each edition in the Visual Studio Documentation.

Prepare the UDI Development Environment Software Prerequisites

The UDI development environment has the following software prerequisites:

- Any Windows operating system that Visual Studio 2010 supports (Windows 7 or Windows Server® 2008 R2 is recommended.)

  You will need a Windows operating system that supports the processor architecture for which you want to develop. You can perform 32-bit and 64-bit UDI development using a 64-bit operating system. You only do 32-bit UDI development on 32-bit operating systems. For this reason, you should use a 64-bit operating system.

  > ⓘ **Note**
  >
  > IntelItanium versions (IA-64) of Windows operating system are not supported for UDI development environments.

  For more information about the operating systems that Visual Studio 2010 supports, see the system requirements for each edition in the Visual Studio Documentation.

- Microsoft .NET Framework version 4.0 (required by Visual Studio 2010)
- C++ language (the language used in extending UDI Wizard pages)
- Other languages that WPF supports, such as C#, Visual Basic .NET, or C++/Common Language Infrastructure, which are used to extend UDI Wizard Designer wizard page editors

  > ⓘ **Note**
  >
  > The sample source code for the UDI Wizard Designer wizard page editors is written in C#. Install the C# language if you want to use the sample source code.

Configure the UDI Development Environment

After then UDI development environment prerequisites are met, perform the following steps to configure the UDI development environment:

1. Install Visual Studio 2010.

Ensure that you install the C++ language and any other language that WPF supports.

> **Note**
>
> The sample source code for the UDI Wizard Designer editor pages is written in C#. Install the C# language if you want to use the sample source code.

For more information about installing Visual Studio 2010, see Installing Visual Studio.

2. Install MDT.

   For more information about how to install MDT, see the section, "Installing or Upgrading to MDT", in the MDT document *Using the Microsoft Deployment Toolkit*.

3. In Windows Explorer, create *local_folder* (where *local_folder* is any folder located on a local drive on the development computer).

4. Copy the *installation_folder*\SDK folder to *local_folder* (where *installation_folder* is the folder in which you installed MDT and *local_folder* is any folder located on a local drive on the development computer).

   You copy the SDK folder to another location because MDT is installed in the Program Files folder, which cannot be written to without elevated permissions. Copying the SDK folder to another location allows you to modify the files in the SDK folder without requiring elevated permissions.

5. Copy the *installation_folder*\Templates\Distribution\Tools folder to *local_folder* (where *installation_folder* is the folder in which you installed MDT and *local_folder* is the folder you created earlier in the process).

6. Rename the *local_folder*\Tools folder to *local_folder*\OSDSetupWizard(where *local_folder* is the folder you created earlier in the process).

   When completed, the folder structure beneath *local_folder* should look like the folder structure illustrated in Figure 2 (where *local_folder* is the folder you created earlier in the process and is shown as *UDIDevelopment* in the figure).

Figure 2. Folder structure for UDI development

Figure 2. Folder structure for UDI development

Verify the UDI Development Environment

When the UDI development environment is configured, verify that the UDI development environment is configured correctly by ensuring that the sample projects build correctly in Visual Studio 2010.

Verify that the UDI development environment is configured correctly by determining whether:

- The SamplePage project builds correctly as described in Verify That the SamplePage Project Builds Correctly
- The SampleEditor project builds correctly as described in Verify That the SampleEditor Project Builds Correctly

Verify That the SamplePage Project Builds Correctly

The SamplePage project provides an example of how to create a custom wizard page for the UDI Wizard. For more information about the SamplePage project, see Review the SamplePage Visual Studio Solution.

To verify that the SamplePage project builds correctly

1. Start Visual Studio 2010.
2. Open the SamplePage project.

   The SamplePage project resides in the *local_folder*\SDK\UDI\SamplePage folder (where *local_folder* is the folder you created earlier in the process).
3. In Visual Studio 2010, in Solution Explorer, right-click the SamplePage project, and then select **Properties**.

   The **SamplePage Property Pages** dialog box appears.
4. In the **SamplePage Property Pages** dialog box, go to Configuration Properties/Debugging.
5. In the Debugging properties, under **Configuration**, select **All Configurations**.
6. In the Debugging properties, under **Command**, type **$(TargetDir)\OSDSetupWizard.exe.**
7. In the Debugging properties, under **Working Directory**, type **$(TargetDir)**.
8. In the **SamplePage Property Pages** dialog box, go to Configuration Properties/Build Events/Post-Build Event.
9. In the Post-Build Event properties, under **Command Line**, type the following:

   Windows Command Prompt

   ```
   copy /y "$(ProjectDir)..\..\..\..\OSDSetupWizard\x86\*.*"
   "$(TargetDir)"
   xcopy /y /i "$(ProjectDir)..\..\..\..\OSDSetupWizard\x86\en-us"
   "$(TargetDir)en-us"
   ```

```
copy /y
"$(ProjectDir)..\..\..\..\OSDSetupWizard\OSDResults\Images\UDI_Wizard_B
anner.bmp" "$(ProjectDir)header.bmp"
copy /y "$(ProjectDir)Config.xml" "$(TargetDir)"
copy /y "$(ProjectDir)header.bmp" "$(TargetDir)header.bmp"
```

10. In the **SamplePage Property Pages** dialog box, select **OK**.
11. Save the project.
12. From the **Debug** menu, select **Start Debugging**.

    The **Microsoft Visual Studio**dialog box appears indicating that the source is out of date and asks whether you want to build the project.
13. In the **Microsoft Visual Studio** dialog box, select **Yes**.

    The **No Debugging Information** dialog box appears informing you that no debugging information is available for OSDSetupWizard.exe.
14. In the **No Debugging Information** dialog box, select **Yes**.

    The UDI Wizard opens with the custom wizard page displayed.
15. Verify that you can select a value in **Choose your location**.
16. In the **Wizard with sample page** form, select **Cancel**.

    The **Cancel Wizard** dialog box appears.
17. In the **Cancel Wizard** dialog box, select **Yes**.
18. Close Visual Studio 2010.

Verify That the SampleEditor Project Builds Correctly

The SampleEditor project provides an example of how to create a custom wizard page editor for the UDI Wizard Designer. For more information about the SampleEditor project, see Review the SamplePage Visual Studio Solution.

To verify that the SampleEditor project builds correctly

1. Start Visual Studio 2010.
2. Open the SampleEditor project.

   The SampleEditor project resides in the *local_folder*\SDK\UDI\SampleEditor folder (where *local_folder* is the folder you created earlier in the process).

3. In Visual Studio 2010, in Solution Explorer, select the SampleEditor project.
4. From the **Project** menu, select **Add Reference**.

   The **Add Reference** dialog box opens.
5. In the **Add Reference** dialog box, select the **Browse** tab.
6. On the **Browse** tab, go to *installation_folder*\Bin (where *installation_folder* is the folder in which you installed MDT). Select the following files, and then select **OK**:

   - Microsoft.Enterprise.UDIDesigner.Common.dll
   - Microsoft.Enterprise.UDIDesigner.DataService.dll
   - Microsoft.Enterprise.UDIDesigner.Infrastructure.dll
   - Microsoft.Practices.Prism.dll
   - Microsoft.Practices.ServiceLocation.dll
   - Microsoft.Practices.Unity.dll
   - RibbonControlsLibrary.dll

   > **Note**
   >
   > You can select multiple files on the **Browse** tab by holding down the CTRL key while you select the files.
7. In Solution Explorer, go to SampleEditor/References.
8. Verify that none of the references have any warnings or errors.
9. In Solution Explorer, right-click the SampleEditor project, and then select **Properties**.

   The **SampleEditor Property Pages** dialog box appears.
10. In the **SampleEditor Property Pages** dialog box, select the **Debug** tab.
11. On the **Debug** tab, select **Start external program**.
12. In **Start external program**, type ***installation_folder*\Bin\UDIDesigner.exe** (where *installation_folder* is the folder in which you installed MDT), and then select **OK**.

> **Tip**
>
> You can select the ellipsis (...) button to browse to the folder and select UDIDesigner.exe.

13. From the **File** menu, select **Save All**.
14. Copy the *local_folder*\SDK\SamplePage\SamplePage.dll.config file to the *installation_folder*\Bin\Config folder (where *local_folder* is the folder you created on the development computer earlier in the configuration process and*installation_folder* is the folder in which you installed MDT).
15. In Visual Studio 2010, from the **Debug** menu, select **Start Debugging**.

    The UDI Wizard Designer starts.
16. In the UDI Wizard Designer, on the Ribbon, select **Open**.

    The **Open** dialog box appears.
17. In the **Open** dialog box, open the *local_folder*\SDK\SamplePage\SamplePage\Config.xml file (where *local_folder* is the folder you created on the development computer earlier in the configuration process).

    The Config.xml file opens, and the Custom StageGroup is displayed in the details pane.
18. In the details pane, select the **Configure** tab.
19. Review the configuration information for the **Location** box, including the following:

    - **Unlocked** button, with which you enable or disable the **Location** box
    - **Default value** box, in which you enter a default value to be displayed in the **Location** box
    - **Friendly display name visible in summary page**, in which you enter the caption for the information displayed on the **Summary** page
    - **Location** list box, which includes a list of possible locations
20. Close the UDI Wizard Designer.
21. Close Visual Studio 2010.

Reviewing the UDI SDK Examples

Before beginning development, review the examples provided in the UDI SDK. Use the information in this guide and the source code in the examples to help you create your own UDI custom wizard pages and wizard page editors.

Go through the UDI SDK examples by reviewing the:

- Contents of the SDK folder that you copied earlier in the installation process as described in Review the Contents of the SDK Folder
- Custom UDI wizard page example as described in Review the SamplePage Visual Studio Solution
- Custom UDI wizard page editor example as described in Review the SampleEditor Visual Studio Solution

Review the Contents of the SDK Folder

During configuration of the UDI development environment, you copied the SDK folder from the folder in which you installed MDT to another folder that you created. Table 1 lists the folders immediately beneath the SDK folder and provides a brief description of each.

Table 1. Folders in the UDI SDK

Expand table

Folder	This folder contains
Includes	The C++ header files necessary for creating custom wizard pages for the UDI Wizard
Libs	The C++ library files that will be linked to your custom page; there are 32-bit and 64-bit versions of the static link libraries. **Note:** Itanium versions of the libraries (IA-64) are not available.
SampleEditor	A Visual Studio project for building a custom editor used to edit the SamplePage page in UDI Wizard Designer, which is written in C#
SamplePage	A Visual Studio project for building a custom UDI wizard page, which is written in Visual C++

Review the SamplePage Visual Studio Solution

Before you begin creating your custom wizard pages and wizard page editors, perform the following tasks to prepare the UDI development environment:

- Review the stages in the life cycle of a UDI wizard page as described in Review the Wizard Page Life Cycle.
- Review the Visual Studio solution for the SamplePage example in the UDI SDK as described in Review the SamplePage Example.

Review the Wizard Page Life Cycle

A UDI wizard page has methods that correspond to each stage (or phase) of the life cycle of the page. As a part of creating your custom wizard page, you need to override these methods with your code. Table 2 lists the methods that you will need to override and provides a brief description of each method, including when to use the method in the wizard page life cycle.

Table 2. Methods in a Wizard Page Life Cycle

Expand table

Method	Description
OnWindowCreated	This method is called once, after the page's window has been created. For this method, write code that initializes the page for the first time and only needs to be performed once. For example, use this method to initialize fields or to read configuration information from the **Setter** elements in the UDI Wizard configuration file.
OnWindowShown	This method is called each time the page is displayed (shown) in the UDI Wizard. It is called the first time the page is displayed and each time you navigate to the page by selecting **Next** or **Back** in the wizard. For this method, write code that prepares the page to be displayed—for example, reading memory variables, task sequence variables, or environment variables, and then updating the page based on any changes to those variables.
OnCommonControlEvent	This method can be called anytime the wizard page is displayed and receives a WM_NOTIFY message from a child (typically, common controls). For this method, write code that handles WM_NOTIFY based on the

Method	Description
	notification message. For example, you may want to respond to events from a common control, such as responding to select or double-click events for a **TreeView** control.
OnUnhandledEvent	This method is called anytime an unhandled window message occurs for your wizard page. This method provides the opportunity to intercept and handle these otherwise unhandled window messages. For this method, write code that handles the window messages that are pertinent to your wizard page. Typically, you will not need to override this method.
OnNextSelected	This method is called when you select **Next** in the wizard. For this method, write code that performs any necessary actions before moving to the next wizard page—for example, performing validation that can take a long time. If the validation fails, you can cancel the **Next** request and display a message.
OnWindowHidden	This method is called each time the page is hidden when either the previous or next wizard page is shown. For this method, write code that performs any actions before the page is hidden, prior to another page being shown. Typically, you will not need to override this method.

Review the SamplePage Example

Review the SamplePage example using the following list, which represents the sequence of events during the wizard page life cycle of the SamplePage example:

1. The UDI Wizard, OSDSetupWizard.exe, reads the configuration information from the UDI Wizard configuration file in the example (the Config.xml file) as described in Step 1: The UDI Wizard (OSDSetupWizard.exe) Reads the Config.xml File.

2. The UDI Wizard loads the DLLs required for each wizard page listed in the UDI Wizard configuration file as described in Step 2: The UDI Wizard Loads the DLL for the Custom Wizard Page.

3. The UDI Wizard displays the custom wizard page and allows for the desired control interaction as described in Step 3: The UDI Wizard Displays the Custom Wizard Page.

4. When the custom wizard page has collected the information, perform any tasks necessary before selecting **Next** to proceed to the next wizard as described in Step

4: The Next Button Is Selected in the Custom Wizard Page.

Step 1: The UDI Wizard (OSDSetupWizard.exe) Reads the Config.xml File

When the UDI Wizard (OSDSetupWizard.exe) starts, by default it reads the UDI Wizard configuration file, which is the UDIWizard_Config.xml file—the primary configuration file for the UDI Wizard.

> **Note**
>
> The example uses the Config.xml file as the configuration file. In MDT, the default configuration file is the UDIWizard_Config.xml file, which resides in the Scripts folder in the MDT Files package for configuration.

You can override the default configuration file that the UDI Wizard uses by modifying the UDI Wizard task sequence step to use the **/definition** parameter. For more information about overriding the default configuration file that the UDI Wizard uses, see "Override the Configuration File That the UDI Wizard Uses".

The top-level elements in the Config.xml file are the

- DLLs element
- Style element
- Pages element
- StageGroups element

  For more information about the schema of the UDI Wizard configuration file and each of these elements, see UDI Wizard Configuration File Schema Reference.

  The UDI Wizard scans the **DLLs** element looking for the .dll files to load. In the example, two .dll files are listed: SamplePage.dll and SharedPages.dll. These .dll files must reside in the same folder as OSDSetupWizard.exe—the Tools*platform* folder (where *platform* is x86 for the 32-bit version or x64 for the 64-bit version).

  The UDI Wizard scans the **Pages** element looking for the pages that are defined. In the example, two pages are defined: **Custom** and **SummaryPage**. The **Type** attribute of the **Page** element is defined in the PageClassIDs.h file and uniquely defines the type of your custom page.

In the example, the defined type is **Microsoft.SamplePage.LocationPage**. For your custom page, substitute the following to avoid any potential conflicts with other pages you may create in the future:

- Your organization name in the place of **Microsoft**.
- Your project name in the place of **SamplePage**.
- Your custom wizard page name in the place of **LocationPage**.

Step 2: The UDI Wizard Loads the DLL for the Custom Wizard Page

When the UDI Wizard loads your DLL, it calls the **RegisterFactories** function, which must be implemented in your .dll file. In the example, this function is implemented in the dllmain.ccp file. Each wizard page you create must implement the **RegisterFactories** function.

The **RegisterFactories** function is used to register the factory class of your wizard page with the class factory registry for the UDI Wizard. *Class factories* are classes that can create an instance of another class. The **RegisterFactories** function creates a new instance of a factory class and passes that class to the class factory registry for the UDI Wizard, which makes that factory class available to the wizard. The UDI Wizard looks for a factory class registered with an ID that matches the **Type** attribute of the **Page** element for the custom wizard page.

In the example, the ID is defined as **ID_Location** in the PageClassIds.h file as **Microsoft.SamplePage.LocationPage**, which matches the **Type** attribute for the **Page** element in the Config.xml file. **ID_Location** is passed as a parameter in the **RegisterFactories** function implemented in the dllmain.ccp file.

You can create a function using the Register_*name* function template to simplify the creation of a new factory instance and register the newly created instance. The **name** value provided using the Register function template must implement the **iClassFactory** interface. The ClassFactoryImpl Class handles most of the details for implementing a class factory.

You can also use the **RegisterFactories** function to register task types and validator types. For more information, see the following:

- Creating Custom UDI Tasks
- Creating Custom UDI Validators

> ⓘ **Note**

The example contains and registers only the one custom wizard page. The example does not include custom tasks or validators and so does not register any custom tasks or validators.

Step 3: The UDI Wizard Displays the Custom Wizard Page

The custom wizard page in the example is defined in the LocationPage.cpp file. Wizard pages are derived from template classes that provide much of the functionality a page has. All wizard pages should derive from the WizardPageImpl Template Class, which implements the IWizardPage Interface. Each wizard page can implement other optional template classes and corresponding interfaces based on the needs of the page.

The WizardPageImpl Template Class has several useful interfaces that can help you write custom wizard pages. Implement the WizardPageImpl Template Class as the base class for your custom wizard page.

For a list of the available:

- Template classes for wizard pages, see Wizard Page Helper Classes
- Interfaces for the wizard page template classes, see Wizard Page Interfaces

  The custom wizard page in the example is derived from the WizardPageImpl Template Class and implements the IWizardPage Interface. In addition, the custom wizard page implements the **IFieldCallback** interface. Both of these are implemented in the LocationPage.cpp file.

  The example custom wizard page overrides the following methods:

- **OnWindowCreated**. The **OnWindowCreated** method in the example wizard page calls the following methods:

  - AddField. This method relates the **IDC_COMBO_LOCATION** box control in the **IDD_LOCATION_PAGE** resource with the Data element named **Location** in the Config.xml file.

    In addition to the **AddField** method, you could use the AddRadioGroup and AddToGroup methods to support other controls and behaviors.

    > ⓘ **Note**
    >
    > Ensure that you call the AddField, AddRadioGroup, or AddToGroup method prior to calling the InitFields method.

- InitFields. Use this method to initialize the fields (controls) that you have added to the form. The pointer of the page is a parameter. In the example, the **this** pointer is passed, which refers to the current page.

  > ⓘ Note
  >
  > To support the use of the **this** pointer, you must implement the **IFieldCallback** interface in addition to the interfaces that the WizardPageImpl Template Class supports.

  The **IFieldCallback** interface calls the **SetFieldDefault** method, which is used to set the default values for controls other than text box and check box controls. In the example, the **SetFieldDefault** method sets the initial index of the combo box control based on the default value specified in the **Default** element for the Field element in the Config.xml file.

  The **OnWindowCreated** method sets up the form controller using the IFormController interface. For more information about setting up the form controller, see Setting up the Form.

- **InitLocations**. This method populates the combo box from the list of locations in the Config.xml file. The Data element and child DataItem elements the Confg.xml file provide the list of possible values.

- **OnNextSelected**. This method performs the following tasks:

  - Updates the **TSLocation** task sequence variable with the value selected in the combo box using the **SaveFields** method

  - Adds information that will be shown on the **Summary** page using the **SaveFields** method

Step 4: The Next Button Is Selected in the Custom Wizard Page

When the user completes the fields on the custom wizard page, they select **Next**, which calls the **OnNextSelected** method. The **OnNextSelected** method performs any necessary tasks before proceeding to the next wizard page, such as recording any configuration changes made on the custom wizard page.

For the example custom wizard page, the override for the **OnNextSelected** method is implemented in the LocationPage.ccp file. In the **OnNextSelected** method in the example custom wizard page, the following methods are called:

1. InitSection. This method initializes the header (label caption) for the summary data displayed on the **Summary** page. Typically, you can set this value using the **DisplayName()** function. The data associated with this caption is saved using the SaveFields method.

2. SaveFields. This method saves field values to task sequence variables and to the data displayed on the **Summary** page.

Review the SampleEditor Visual Studio Solution

Before you begin creating your own custom wizard pages and wizard page editors, perform the following steps to prepare the UDI development environment:

- Review the architecture of the UDI Wizard Designer as described in Review the UDI Wizard Designer Architecture.
- Review the components of a UDI Wizard page that can be customized using the UDI Wizard configuration file as described in Review Configurable Components of a UDI Wizard Page.
- Review the EditorPage example provided in the UDI SDK as described in Review the EditorPage Example.

Review the UDI Wizard Designer Architecture

The UDI Wizard Designer was developed using WPF, Prism, and Unity. The UDI Designer is used to edit the UDI Wizard configuration file (UDIWizard_Config.xml), which the UDI Wizard (OSDSetupWizard.exe) reads at runtime. The Pages element in the UDI Wizard configuration file contains a list of pages that has a separate Page element for each wizard page.

When you edit the configuration settings for a wizard page, the UDI Wizard Designer loads the custom page editor that corresponds to the wizard page type. The custom wizard page editors are developed as WPF user controls. The custom wizard page editor pages use the Model-View-ViewModel (MVVM) design pattern for WPF.

The MVVM design pattern helps separate the user interface (UI; presentation) from the data being presented. The data is a façade over the Page element in the UDI Wizard configuration file (the Config.xml file in the example), which is accessed using the CurrentPage property of the IDataService interface.

The UDI Wizard Designer uses the **DependencyAttribute** to obtain access to the **DataService** class based on the dependency injection framework in Unity. For more

information about the dependency interjection framework in Unity, see Inject Some Life into Your Applications—Getting to Know the Unity Application Block.

Review Configurable Components of a UDI Wizard Page

As you create your custom wizard page, some of the configuration settings may be set in code and cannot be changed after you have compiled the page. However, for other configuration settings, you will need to allow those configuration settings to be changed using the UDI Wizard Designer.

Typically, the configuration settings that you want to configure using the UDI Wizard Designer are saved in the UDI Wizard configuration file (the Config.xml file in the example). However, you can also create your own separate configuration file, if necessary. One example of using a separate configuration file is the UDIWizard_Config.xml.app file, which the **Application Discovery** task and the **ApplicationPage** wizard page type use.

The following is a list of the typical configuration settings that you can manage using the UDI Wizard Designer:

- **Field**. Use fields allow users to provide input. Fields appear as Field elements in the UDI Wizard configuration file (UDIWizard_Config.xml), which contains the configuration settings for each field. The corresponding wizard page editor needs to provide a method for editing the field configuration settings for the field using the FieldElementControl.

- **Properties**. Setters help create properties for entities on the page, such as pages in the Page element, fields in the Field element, or data in the Data or DataItem elements. You configure properties in the Setter elements. Add a separate Setter element for each property you want to define. You edit the properties using the SetterControl and configure other Setter elements using other controls.

- **Data**. Data is used to store information for use by the wizard page and other components. You can define data for pages or fields using the Data or DataItem elements. The data can be defined in a flat or hierarchical structure through the proper use of the Data or DataItem elements. The Config.xml in the example in the SDK shows how to build flat data structures.

  The custom wizard page editor that you create must be able to manage these configuration settings.

Review the EditorPage Example

The EditorPage example is used to configure the configuration settings for the **SamplePage** wizard page in the UDI Wizard configuration file. The EditorPage example has the following primary components:

- UI to configure the **Location** combo box settings
- UI to add or edit a location in the list of possible locations, which are shown in the **Location** combo box
- Configuration settings read from and saved to the UDI Wizard configuration file
- Supporting code for the other components

Review the EditorPage example in Visual Studio by performing the following steps:

1. Review how the **SampleEditor** wizard page editor is loaded and initialized in the UDI Wizard Designer as described in Review Wizard Page Editor Loading and Initialization.
2. Review the UI used to edit the **Location** combo box in the LocationPageEditor.xaml and LocationPageEditor.xaml.cs files as described in Review the User Interface Used to Configure the Location Combo Box.
3. Review the UI used to add or edit locations to the list in the AddEditLocationView.xaml and AddEditLocationView.xaml.cs files as described in Review the User Interface Used to Modify the List of Possible Locations.
4. Review the code used to manage configuration information saved in the UDI Wizard configuration file as described in Review the Code Used to Manage Configuration Information.

Review Wizard Page Editor Loading and Initialization

Custom wizard page editors are loaded as required by the UDI Wizard Designer. The UDI Wizard Designer configuration files are loaded when the UDI Wizard Designer starts. The UDI Wizard Designer scans the *install_folder*\Bin\Config folder (where *install_folder* is the name of the folder where MDT is installed) for files that have a .config file extension.

During the configuration of the UDI development environment, you copied the SamplePage.dll.confg file to the *install_folder*\Bin\Config folder. When you start the UDI Wizard Designer, the SamplePage.dll.confg file is found and loaded.

The UDI Wizard Designer uses the following attributes of the Page element in the SamplePage.dll.confg file to load and initialize the EditorPage example:

- **DesignerAssembly**. This attribute determines the name of the DLL to be loaded. This DLL needs to be placed in the same folder as the UDIDesigner.exe file, which is the *install_folder*\Bin folder (where *install_folder* is the name of the folder in which MDT is installed).
- **DesignerType**. This attribute is the Microsoft .NET type name of the class that contains the WPF user control.
- **Type**. Use this attribute to configure the page type of the custom wizard page, which the UDI Wizard loads. The UDI Wizard Designer uses this attribute to locate the appropriate Page element in the UDI Wizard configuration file.
- **Dll**. Use this attribute to configure the DLL element in the UDI Wizard configuration file, which the UDI Wizard Designer creates.
- **Description**. Use this attribute to provide information about the wizard page editor. The value of this attribute is shown in the **Add New Page** dialog box in the UDI Wizard Designer, which is used to add the wizard page to the "Page Library".
- **DisplayName**. Use this attribute to provide the name of the custom wizard page that is displayed in the UDI Wizard Designer. The value of this attribute is shown in the **Add New Page** dialog box in the UDI Wizard Designer, which is used to add the wizard page to the "Page Library".

  In the example, the type of the **SamplePage** custom wizard page is **Microsoft.SamplePage.LocationPage**, which is saved in the Config.xml file. The Config.xml file resides in the *local_folder*\SDK\SamplePage\SamplePage folder to (where *local_folder* is the folder you created on the development computer earlier in the configuration process).

Review the User Interface Used to Configure the Location Combo Box

When the wizard page editor is loaded and initialized, the SampleEditor wizard page editor is loaded when a page with a type of **Microsoft.SamplePage.LocationPage** is edited. The UI for the page editor is stored in the LocationPageEditor.xaml file.

If you examine the UI on the **Design** tab and the code on the **XAML** tab, you can see the relationship between the graphical UI and the elements and attributes in the Extensible Application Markup Language (XAML).

For example, if you review the **Controls:FieldElementControl** element in the XAML you can see how that relates to the layout of the corresponding UI. Use the **Controls:FieldElementControl** element to define the FieldElementControl control.

The **Binding** parameters in the XAML file bind the fields on the sample page editor with the information in the UDI wizard configuration file. For example, the following code ties the **Default value** text box with the Default element in the UDI wizard configuration file (Config.xml in the example):

```
XML
<TextBox Text="{Binding FieldData.DefaultValue,
 UpdateSourceTrigger=PropertyChanged,
 Mode=TwoWay}"/>
```

For more information, see How to: Make Data Available for Binding in XAML.

Use the **Views:CollectionTControl.ColumnCollectionView** element in the XAML to edit the list of available locations in the grid view. You use the CollectionTControl control to display the grid view and bind the grid view to the Data element with the name **Location** in the UDI configuration file.

Review the User Interface Used to Modify the List of Possible Locations

The UI for modifying the list of possible locations consists of:

- A context-sensitive menu and Ribbon buttons that allow you to add, edit, remove, or change the order of items in the list of locations as described in Review Context-sensitive Menu and Ribbon Buttons for Modifying the List of Locations
- A dialog box that is initiated when you select to add or edit an item in the list of locations as described in Review the Dialog Box for Adding or Editing Locations

Review Context-sensitive Menu and Ribbon Buttons for Modifying the List of Locations

When you right-click in the list box that contains the list of locations, a context-sensitive menu is displayed. The Ribbon has corresponding buttons that allow you to perform the same tasks. The **Views:CollectionsTControl** control element in the LocationPageEditor.xaml file defines the methods called based on the action taken and properties that you set as follows:

- **SelectedItem**. This data-bound property is activated when the user selects an item from the list. This property is tied to the **CurrentLocation** property in the view model, which is located in the LocationPageEditorViewModel.cs file and used by

the CollectionTControl control to pass the item selected when you edit or remove an existing item.

- **AddItemAction**. This action is performed when the user selects the **Add Item** option from the context-sensitive menu or the corresponding buttons on the Ribbon. There is a data binding to a property in the view model that returns the **AddLocationAction** object. This object is the **AddLocationCallback** method, located in the LocationPageEditorViewModel.cs file, and displays the dialog box in the AddEditLocationView.xaml file.

- **EditItemAction**. This action is performed when the user selects the **Edit Item** option from the context-sensitive menu. There is a data binding to a property in the view model that returns the **EditLocationAction** object. This object is the **EditLocationCallback** method, located in the LocationPageEditorViewModel.cs file, and displays the dialog box in the AddEditLocationView.xaml file.

- **RemoveAction**. This action is performed when the user selects the **Remove Item** option from the context-sensitive menu. There is a data binding to a property in the view model that returns the **RemoveAction** object. This object is the **EditLocationCallback** method, located in the LocationPageEditorViewModel.cs file, and shows a message that confirms the deletion of the location.

Review the Dialog Box for Adding or Editing Locations

If you add a new location to the list of locations or edit an existing location, a message is displayed that is in the AddEditLocationView.xaml file. The message is displayed using the ShowDialogWindow window method in the LocationPageEditorViewModel.cs file.

The UI in the AddEditLocationView.xaml file consists of:

- A dialog frame named **DialogFrame**, which includes the following elements:
  - A title, which you configure using the **DialogTitle** attribute of the dialog frame
  - An **OK** button, which sets the return status as for the **Approved** property to **True** (The return status is checked in the **AddLocationCallback** method in the LocationPageEditorViewModel.cs file to determine whether the user selected **OK**.)
  - A **Cancel** button, which sets the return status as for the **Approved** property to **False** (The return status is checked in the **AddLocationCallback** method in the LocationPageEditorViewModel.cs file to determine whether the user selected **Cancel**.)

- A WPF element that contains:
  - A label, which you configure using the **Content** attribute
  - A text box, which is bound to the Data element with the name **Location** in the UDI configuration file (the Config.xml file in the example)

Review the Code Used to Manage Configuration Information

The configuration information for your custom wizard page is stored in the UDI Wizard configuration file, which is the:

- Config.xml file in the example provided with the UDI SDK (This file contains only the configuration settings for the example.)
- UDIWizard_Config.xml file provided with MDT, stored in the *installation_folder*\Templates\Distribution\Scripts folder (where *installation_folder* is the folder in which you installed MDT); this file contains the configuration settings for all the built-in wizard pages and stages

  In the SampleEditor example, the **Locations** routine helps manage the configuration information and is located in the LocationPageEditorViewModel.cs file. The **Locations** routine returns a list of the locations from the UDI Wizard configuration file. Specifically, the list returned contains an item for each DataItem element in the UDI Wizard configuration file.

Creating Custom UDI Wizard Pages

The high-level process for creating custom UDI wizard pages is as follows:

1. Make a copy of the SamplePage solution as a starting point.
2. Place the desired controls (fields) on the form.
3. Write code to perform the appropriate tasks when the wizard page loads (overrides for the **OnWindowCreated** method), including the following steps:
   a. Initialize the form.
   b. Read memory variables, task sequence variables, environment variables, or XML file information (such as **Setter** properties).
4. Write any code to perform the appropriate tasks when the page is shown (overrides for the **OnWindowShown** method), including the following steps:

a. Enable or disable controls based on information read when the page loaded in step 3.

b. Update the controls based on information read when then page loaded in step 3, such as the population of controls based on the information read.

5. Write any code to perform the appropriate tasks while the user interacts with the wizard page.

6. Write any code to perform the appropriate tasks when the user selects **Next** in the UDI Wizard (overrides for the **OnNextSelected** method), including the following steps:

   a. Update any memory variables, task sequence variables, environment variables, or XML file information.

   b. Update summary page information (if not performed by the fields on the page).

7. Build the solution.

   Ensure that the version of the DLL you create is the same processor platform as the installation of MDT—specifically, the processor platform for Windows Preinstallation Environment (Windows PE). The UDI Wizard can run in:

   - **The existing operating system on the target computer**. You can run 32-bit versions of your wizard page on 32-bit or 64-bit Windows operating systems. However, you can only run 64-bit versions of your wizard page on 64-bit Windows operating systems.

   - **Windows PE on the target computer**. Windows PE does not support running 32-bit applications on a 64-bit version of Windows PE. So, you need to have built a version for your wizard page for each processor architecture of Windows PE that you plan to use.

8. Copy the DLL for your custom wizard page to *installation_folder*\Templates\Distribution\Tools\ platform folder (where *installation_folder* is the folder in which you installed MDT and *platform* is **x86** for the 32-bit version or **x64** is for the 64-bit version).

9. Complete the steps for creating custom page editor.

Creating Custom Wizard Page Editors

The high-level process for creating custom UDI wizard page editors is as follows:

1. Make a copy of the SampleEditor solution as a starting point.
2. Create the primary page editor UI in an .xaml file.
3. Add instances of the FieldElementControl control as required by the wizard page to be configured (if required).
4. Add instances of the SetterControl control as required by the wizard page to be configured (if required).
5. Add instances of the CollectionTControl control as required by the wizard page to be configured (if required).
6. Add the IDataService interface.
7. Write the appropriate code to update the UDI Wizard configuration file based on the configuration settings to be configured using your custom wizard page editor.
8. Create child dialog boxes in a .xaml file, and call them from the primary page editor using the IMessageBoxService interface as required by the wizard page to be configured.
9. Add the appropriate interfaces to the UDI Wizard Designer Ribbon based on the requirements of the wizard page to be configured.
10. Build the solution.

    > **Note**
    >
    > Ensure that the version of the DLL you create is the same processor platform as the installation of MDT. For example, if you install the 64-bit version of MDT, then build a 64-bit version of your custom page editor.

11. Create a UDI Wizard Designer configuration file to load the necessary DLLs and map the wizard page editor with the corresponding wizard page (the SamplePage.dll.config file in the example).

    For more information about the elements required to perform the mapping between the wizard page and the wizard page editor, see the DesignerMappings element, child elements, and corresponding attributes.

12. Copy the UDI Wizard Designer configuration file that you created in the previous step to the *installation_folder*\Bin\Config folder (where *installation_folder* is the folder in which you installed MDT version).

13. Copy the DLL for your custom wizard page editor to the *installation_folder*\Bin folder (where *installation_folder* is the folder in which you installed MDT).

Creating Custom UDI Tasks

UDI tasks are DLLs written in C++ that implement the ITask interface. You register the DLL with the UDI Wizard Designer task library by creating a UDI Wizard Designer configuration file (.config file) and placing it in the *installation_folder*\Bin\Config folder (where *installation_folder* is the folder in which you installed MDT).

> ⓘ **Note**
>
> You can create a DLL that contains wizard pages, tasks, and validators within the same .dll file. You can also create a single UDI Wizard Designer configuration file (.config) that contains the configuration settings for the wizard pages, tasks, and validators in the DLL.

To create custom UDI tasks

1. Write code that implements the ITask Interface and the following methods:
   - Init. This method is called to initialize your task.
   - Execute. This method is called to run your task.
2. Write code that registers the custom task class factory with the factory registry.
3. Build the solution for your custom task.

   > ⓘ **Note**
   >
   > Ensure that the version of the DLL you create is the same processor platform as the installation of MDT. For example, if you install the 64-bit version of MDT, then build a 64-bit version of your custom UDI task.

4. Create a Task element under the TaskLibrary element in the UDI Wizard Designer configuration file similar to the following excerpt:

   XML

   ```
   <Task DLL="OSDRefreshWizard.dll" Description="Discovers supported
   applications for install." Type="Microsoft.OSDRefresh.AppDiscoveryTask"
   Name="Application Discovery">
   ```

```
    <TaskItem Type="Setter" Name="Status Bitmap">
       <Param Name="BitmapFilename"/>
    </TaskItem>
    <TaskItem Type="Setter" Name="Log File">
       <Param Name="log"/>
    </TaskItem>
    <TaskItem Type="Setter" Name="Write Configuration File">
       <Param Name="writecfg"/>
    </TaskItem>
    <TaskItem Type="Setter" Name="Read Configuration File">
       <Param Name="readcfg"/>
    </TaskItem>
</Task>
```

> **Note**
>
> All Task elements should include the **BitmapFilename** parameter. Specify all other parameters as the task requires. For example, in the previous excerpt, the **log** parameter is used to specify a parameter for the location of a log file.

5. Copy the UDI Wizard Designer configuration file created in the previous step to the *installation_folder*\Bin\Config folder (where *installation_folder* is the folder in which you installed MDT).

6. Copy the DLL for your custom task to the *installation_folder*\Templates\Distribution\Tools\ platform folder (where *installation_folder* is the folder in which you installed MDT and *platform* is **x86** for the 32-bit version or **x64** is for the 64-bit version).

Creating Custom UDI Validators

UDI validators are DLLs written in C++ that implement the **IValidator** interface. You register the DLL with the UDI Wizard Designer validator library by creating a UDI Wizard Designer configuration file (.config file) and placing it in the *installation_folder*\Bin\Config folder (where *installation_folder* is the folder in which you installed MDT).

To create custom UDI validators

1. Write code that creates a subclass of the **BaseValidator** class and implements the following methods:

   - **Init(IControl *pControl, IWizardPageContainer *pContainer, IStringProperties *pProperties)**. The form controller calls the **Init** member to

initialize the validator. This method must call the **Init** method for the **BaseValidator** class. It typically reads any properties set for the validator from the UDI Wizard configuration file. For example, the **InvalidCharactersValidator** validator retrieves the value of the **InvalidChars** property using this method.

- **IsValid**. The form controller calls this method to see whether the control contains valid text. The following is an example of the **IsValid** method for a validator that validates that the field is not empty:

```
C++

BOOL IsValid(LPBSTR pMessage)
{
    __super::IsValid(pMessage);

    _bstr_t text;
    m_pText->GetText(text.GetAddress());
    return (text.length() > 0);
}
```

- **Init(IControl *pControl, LPCTSTR message)**. The form controller calls this member for each keystroke and other events so that the validator can validate the contents of the control and updated messages at the bottom of the wizard page (or clear them).

  Typically, these are the only methods that you need to override. However, depending on the validator, you may need to override other methods in the subclass of the **BaseValidator** class you create. For more information about these other methods, see the **BaseValidator** class.

2. Write code that registers the custom task class with the registry factory.

3. Build the solution for your custom task.

> **Note**
>
> Ensure that the version of the DLL you create is the same processor platform as the installation of MDT. For example, if you install the 64-bit version of MDT, then build a 64-bit version of your custom UDI task.

4. Create a Validator element under the **ValidatorLibrary** element in the UDI Wizard Designer configuration file similar to the following excerpt:

XML

```
<Validator
<Validator DLL="" Description="Must follow a pre-defined pattern"
Type="Microsoft.Wizard.Validation.RegEx" Name="NamedPattern">
   <Param Description="Enter the message you want displayed when the
text in this field doesn't match the pattern:" Name="Message"
DisplayName="Message"/>
   <Param Description="The name of a pre-defined regular expression
pattern. Must be Username, ComputerName, or Workgroup"
Name="NamedPattern" DisplayName="Named Pattern"/>
</Validator>
```

> ⚠ **Warning**
>
> All Validator elements should include the **Message** parameter. Specify all other parameters as required by the validator. For example, in the previous excerpt, the **NamedPattern** parameter is used to specify a parameter for the name of a predefined regular expression pattern.

5. Copy the UDI Wizard Designer configuration file created in the previous step to the *installation_folder*\Bin\Config folder (where *installation_folder* is the folder in which you installed MDT).

6. Copy the DLL for your custom task to the *installation_folder*\Templates\Distribution\Tools\ platform folder (where *installation_folder* is the folder in which you installed MDT and *platform* is **x86** for the 32-bit version or **x64** is for the 64-bit version).

UDI Wizard Reference

Wizard Page Components

You can use any of several prebuilt components to build your custom pages.

Creating Component Instances

The UDI Wizard uses class factories to create new instances of objects for you. These factories are registered with a factory registry, using a string as the key to the factory. For example, the **WmiRepository** component is identified by the string "Microsoft.Wizard.WmiRepository," which is available in the IWmiRepository header file as **ID_WmiRepository**.

Assuming that you have written your page as a subclass of **WizardPageImpl**, you can create a new instance of a **WmiRepoistory** like this:

```
C++

PWmiRepository pWmi;
CreateInstance(Container(), ID_WmiRepository, &pWmi);
```

The **CreateInstance** function is a type-safe template function for creating new instances of components. **PWmiRepository** is a smart pointer, so it handles reference counting for you.

Creatable Components

There is a set of components that you can register with the registry. The first set of components is always registered, because the main UDI Wizard executable file provides it. The other two sets of components are provided in "optional" DLLs. For these components to be available, the DLL must be listed in the **DLLs** section of the .config XML file. Your code does not need to know which executable contains a specific component.

The list of component IDs for components (the component name is the same as the ID but without the initial *ID_*) registered with the factory registry (defined in OSDSetupWizard) is shown in Table 3.

Table 3. Component IDs

Expand table

ID	Description
ID_ACPowerTask	(ITask, IWizardComponent) A preflight task that ensures that your computer is not running on battery alone
ID_AppDiscoveryTask	(ITask, IWizardComponent) A specialized task for discovering which software items you have installed on your computer
ID_BackgroundTask	(**IBackgroundTask**, **IWizardComponent**) Can be used to run a task on another thread
ID_CopyFilesTask	(**ITask**, **IWizardComponent**) A task to copy one or more files
ID_FormController	(**IFormController**) You will most like not need to create an instance yourself, as your page receives its own instance

ID	Description
ID_InvalidCharactersValidator	(**IValidator**) Ensures that no text field contains characters from a list provided to the validator
ID_Logger	(**ILogger**) You will most like not need to create an instance yourself, as your page receives a pointer to the shared instance
ID_NonEmptyValidator	(**IValidator**) A validator that ensures that no field is empty
ID_PasswordValidator	(**IValidator**) A validator that ensures that no two text fields have the same content
ID_Regex	(**IRegEx**) Evaluates regular expressions, looking for matches
ID_RegExValidator	(**IValidator**) A validator that validates against a regular expression or a known pattern
ID_SimpleStringProperties	(**IStringProperties**, **ISimpleStringProperties**) Provides a simple way to send properties to tasks without using XML
ID_ShellExecuteTask	(**ITask**, **IWizardComponent**) Execute an external program
ID_SummaryBag	(**ISummaryBag**) Available indirectly from your page via the Form method
ID_TaskManager	(**ITaskManager**, **IBackgroundCallback**, **IWizardComponent**) Manages running a set of tasks and the UI
ID_WmiRepository	(**IWmiRepository**, **IWizardComponent**) Allows you to run Windows Management Instrumentation (WMI) queries
ID_IXmlDocument	(**IXmlDocument**) Provides a façade for reading and writing XML documents

The defined OSDRefreshWizard.dll, shared pages, and other control components are shown in Table 4 and Table 5.

Table 4. Directory Controls

Expand table

ID	Description
ID_Directory	(**IDirectory**) A façade for obtaining directory information from the file system

Table 5. Defined SharedPages.dll

Expand table

ID	Description
ID_ADHelper	(**IADHelper**) Provides a façade for a limited set of features in Active Directory® Domain Services (AD DS)
ID_CpuInfo	(**ICpuInfo**) Determines whether your CPU is 32 or 64 bit
ID_DomainJoinValidator	(**IDomainJoinValidator**) Has some methods for checking whether a set of credentials is allowed to join a domain
ID_DriveList	(**IDriveList**, **IBindableList**, **IWizardComponent**) Uses WMI to obtain a list of drives on your computer
ID_WiredNetworkTask	(**ITask**) A tasks that checks whether you are connected to the network with a hard-wired (instead of wireless) network adapter

Control Components

You interact with the controls on your page through the **GetControlWrapper** template function, which provides access to one of the types of components listed in Table 6.

Table 6. Components

Expand table

Dialog control types	Description
CONTROL_CHECK_BOX	(**ICheckBox**) A façade for working with check box controls
CONTROL_COMBO_BOX	(**IComboBox**) A façade for combo box controls
CONTROL_GENERIC	(**IControl**) Allows you to work with most types of controls to control enable and visible state
CONTROL_LIST_VIEW	(**IListView**) A façade providing access to the features of a list view control
CONTROL_PROGRESS_BAR	(**IProgressBar**) A façade for working with the position of a progress bar control
CONTROL_RADIO_BUTTON	(**IRadioButton**) A façade for working with radio button controls
CONTROL_STATIC_TEXT	(**IStaticText**) A façade that provides read/write permission to the text of a control, such as a label or text box
CONTROL_TREE_VIEW	(**ItreeView**) A façade for working with a tree view control

Image List Component

This component is a façade for an **ImageList** control on your page. You create an image list via the **IListView** or **ITreeView** interface.

FormController Component

The wizard creates this component for you and passes it to your page. You access it from your page using the **Form** method, which the **WizardPageImpl** base class implements.

InvalidCharacterValidator Component

This is a type of validator that you can include on a page. The ID is **ID_InvalidCharactersValidator** (defined in IValidator.h), which has a text value of "Microsoft.Wizard.Validation.InvalidChars."

This validator looks for a single property (a **Setter** element in the .config file) called **InvalidChars**, which is a list of characters that are not allowed. It checks the characters in a text box; if the text contains any characters from this list, the component reports failure.

NonEmptyValidator Component

This is a type of validator that you can include on a page. The ID is **ID_NonEmptyValidator** (defined in IValidator.h), which has a text value of "Microsoft.Wizard.Validation.NonEmpty."

This validator reports failure if the text box (or any other control that supports **IStaticText**) has an empty string value.

PasswordValidator Component

This is a type of validator that you can include on a page. The ID is **ID_PasswordValidator** (defined in IValidator.h), which has a text value of "Microsoft.Wizard.Validation.Password."

This validator works with two different text controls (controls that support **IStaticText**) and reports failure if they do not contain the same values. In other words, it fails if the **Password** and **Confirm Password** text boxes do not match.

Because this validator requires two controls, it needs more setup than other validators. The setup might look something like this:

```
C++

Form()->AddToGroup(IDC_EDIT_PASSWORD, IDC_EDIT_PASSWORD2);
PValidator pValidator;
Form()->AddValidator(IDC_EDIT_PASSWORD, ID_PasswordValidator, pMessage,
&pValidator);
PStaticText pPassword2;
GetControlWrapper(View(), IDC_EDIT_PASSWORD2, CONTROL_STATIC_TEXT,
&pPassword2);
pValidator->SetProperty(0, pPassword2);
```

First, you define the **Confirm Password** control as a "child" of the **Password** control. That way, if the form controller disables the **Password** control, it will also disable the **Confirm Password** control. Next, add a password validator to the form. Finally, provide the password validator with the interface to the **Confirm Password** control.

Because of the requirement for two controls, you must use code to set up this validator rather than the .config XML file.

RegExValidator Component

This is a type of validator that you can include on a page. The ID is **ID_RegExValidator** (defined in IValidator.h), which has a text value of "Microsoft.Wizard.Validation.RegEx."

This validator compares the contents of a text control (one that supports **IStaticText**) to a regular expression and fails if the text does not match the regular expression.

Alternatively, you can use this validator with a predefined named pattern. To use a regular expression, the XML must contain a setter property called **Pattern**. If you want to use a named pattern instead, use a setter called **NamedPattern** set to one of the values in Table 7.

Table 7. Named Pattern Setters

Expand table

Pattern	Description
Username	Verifies that the text is either of the form domain\user or user@domain
ComputerName	The name must be between 1 and 15 characters long and cannot include a set of characters (such as : and ?)

Pattern	Description
Workgroup	The name must be between 1 and 15 characters long and cannot contain a set of characters (such as =, +, and ?)

FactoryRegistry Component

This component keeps track of all class factories and services. It implements the **IFactoryRegistry** interface and is available indirectly through your page's **Container** method. In addition, the registry loads extension DLLs. After it loads a DLL, the registry looks for an exported function called **RegisterFactories**. You must implement this function and in it register the class factories for your pages, tasks, and validators (and any other class factories you want to register). Here is an example from the sample project:

```
C++

extern "C" __declspec(dllexport) void RegisterFactories(IFactoryRegistry
*factories)
{
Register<LocationPageFactory>(ID_LocationPage, factories);
}
```

Logger Component

This component is available to your page via the **Logger** method (implemented by **WizardPageImpl**). You use this method to write entries to the log file. The contents of the log file are useful for diagnosing issues users might have running the UDI Wizard.

PropertyBag Component

The *property bag* is a container for memory variables. It is available from your page using **Container()->Properties()**. Memory variables are useful for passing temporary data among different pages.

TSVariableBag and TSRepository Components

The **TSVariableBag** component allows you to read and write task sequence variables. It keeps the values in memory until the user selects **Finish** (by default). You can access the **TSVariable** bag via the page's **TSVariables** method (implemented by the

WizardPageImpl base class). These components log all reads and writes of task sequence variables.

WmiRepository Component

This component provides a façade for working with WMI queries. You can call the **CreateInstance** helper function with **ID_WmiRepository** to obtain an instance of this component, which supports the **IWmiRepository** interface. This component returns result records via the **IWmiIterator** interface.

Wizard Page Helper Classes

You can create custom UDI wizard pages using built-in helper classes provided with the UDI SDK. Table 8 lists the helper classes that you can use to create custom wizard pages.

Table 8. Helper Classes

Expand table

Helper class	Description
ClassFactoryImpl Class	This is a useful base class for creating a class factory that you can then register with the factory registry.
Interface Template Class	Use this template class when you want to build a component that implements more than one interface.
Path Helper Class	This class provides common file/directory operations.
Pointer Template Class	This class provides reference counting for lifetime management in COM components. It is important to release interfaces when you are done with them. This template class handles the lifetime automatically.
PUnknown Class	This class is a smart pointer specifically for the IUnknown interface. For all other interfaces, use the Pointer template class.
StringUtil Helper Class	This class provides helper methods that make it easier to work with strings.
SubInterface Template Class	This base class makes it easier to implement a component that supports an interface that itself inherits from another interface.
UnknownImpl Template Class	This class handles most of the details of creating a COM component.
WizardComponent	This base class is used for creating components that need access to the

Helper class	Description
Template Class	wizard services, such as component creation and logging.
WizardPageImpl Template Class	This base class should be used as the base class for all custom wizard pages

ClassFactoryImpl Class

This is a useful base class for creating a class factory that you can then register with the factory registry.

The following is an excerpt from the LocationPage.h file in the sample project to define the **ClassFactoryImpl** class.

C++

```
#pragma once

#include "ClassFactoryImpl.h"

class LocationPageFactory :public ClassFactoryImpl
{
protected:
    IUnknown *CreateNewInstance();
};
```

The following is an excerpt from the LocationPage.cpp file in the sample wizard page used to define the class factory for the page.

C++

```
IUnknown *LocationPageFactory::CreateNewInstance()
{
    return static_cast<IWizardPage *>(new LocationPage);
}
```

Interface Template Class

Use this template class when you want to build a component that implements more than one interface—for example:

C++

```
classLocationPage :public Interface<IFieldCallback,
```

```
WizardPageImpl<IDD_LOCATION_PAGE>>
```

This code creates a base class chain that supports both **IFieldCalback** and the interfaces that **WizardPageImpl** supports (which happens to be **IWizardPage**).

Path Helper Class

This class provides common file/directory operations:

```
C++

static inline std::wstring GetModulePath(HINSTANCE hModule)
```

It also returns the full path to the .exe or .dll file with the instance handle that you provide to this method:

```
C++

static inline std::wstring GetModuleFilename(HINSTANCE hModule)
```

The class returns the full path and file name of the .exe and .dll file with the instance handle that you provide to this method:

```
C++

static inline std::wstring GetDirectoryName(LPCWSTR fullName)
```

. . . or just the path while stripping the file name:

```
C++

static inline std::wstring GetFileName(LPCWSTR fullName)
```

Given a path with a file name, the path helper class returns the file name only:

```
C++

static inline std::wstring Combine(LPCWSTR path, LPCWSTR name)
```

Finally, the class returns a new string that is the combined path and file name (or another path).

Pointer Template Class

This class is defined in Pointer.h. Because COM components use reference counting for lifetime management, it is important that you always release interfaces when you are done with them. Microsoft provides a template class that handles the lifetime automatically. For example, if you want a smart pointer for an XML interface, you could write something like this:

```
C++

Pointer<IXMLDOMNode> pNewChild
pXmlDom->CreateNode(NODE_ELEMENT, L"MyElement", L"", &pNewChild);
```

The first line defines the smart pointer. The second line shows retrieving a smart pointer via another call. The **&** operator always releases an existing interface if it contains one and returns the address for the internal pointer. Once you have retrieved a pointer like this, the **Pointer** instance calls **Release** for you when the variable goes out of scope. Microsoft recommends that you use smart pointers instead of calling **AddRef** and **Release** manually.

In addition, the **Pointer** smart pointer class calls **QueryInterface** to retrieve other interfaces for you. For example, when the factory registry creates a new instance of a component, it has code like this:

```
C++

PWizardComponent pComp = pUnknown;
if (pComp != nullptr)
    pComp->SetContainer(m_pContainer);
```

The first line calls **QueryInterface** behind the scenes to request the **IWizardComponent** interface. The resulting smart pointer will equal **nullptr** if the component does not support that interface.

PUnknown Class

This class is a smart pointer specifically for the **IUnknown** interface. For all other interfaces, use the **Pointer** template class.

StringUtil Helper Class

This class is defined in Utilities.h and provides helper methods that make it easier to work with strings:

```
C++
```

```
static inline int CompareIgnore(LPCWSTR first, LPCWSTR second)
```

This method compares two strings while ignoring case (see Table 9).

Table 9. StringUtil Helper Class

Expand table

Returns	Description
0	Strings match, ignoring case
<0	First < second
>0	First > second

Here is an example:

```
C++
```

```
static inline std::wstring Format(LPCWSTR input, int index, LPCWSTR value)
static inline std::wstring Format(LPCWSTR input, int index, DWORD value)
```

These methods are a bit like the Microsoft .NET **Format** methods in the sense that parameters are in the form of **{0}**. However, they do not perform any formatting of the input—just substitution:

```
C++
```

```
static inline std::wstring Printf(std::wstring format, I val)
static inline std::wstring Printf(std::wstring format, I val1, J val2)
static inline std::wstring Printf(std::wstring format, I val1, J val2, K
val3)
static inline std::wstring Printf(std::wstring format, I val1, J val2, K
val3, L val4)
```

These are wrappers around the **StringCchPrintf** that return a **wstring** so you do not have to allocate memory for strings or buffers yourself.

SubInterface Template Class

This base class makes it easier to implement a component that supports an interface that itself inherits from another interface. For example, the **ICheckBox** interface inherits from **IControl**. Here is how this class is used to define the **CheckBoxWrapper**:

```
C++

classCheckBoxWrapper :public SubInterface<IControl, UnknownImpl<ICheckBox> >
```

The base interface is the first parameter, while the derived interface is the second parameter.

UnknownImpl Template Class

This class is defined in UnknownImpl.h and handles most of the details of creating a COM component. Here is an example of how you would use this base class:

```
C++

classDirectory :public UnknownImpl<IDirectory>
```

This code defines a class that supports the **IDirectory** interface.

WizardComponent Template Class

This class is defined in IWizardComponent.h and is a useful base class for creating components that need access to the wizard services, such as component creation and logging.

As an example, here is how the **CopyFilesTask** component is defined:

```
C++

classCopyFilesTask :public WizardComponent<ITask>
{
    ...
```

The parameter for this template class is the "main" interface you want to use for your component, which in the case of tasks is **ITask**. Using **WizardComponent** means that your component supports both the interface your provide (**ITask** in this example) and **IWizardComponent**.

Whenever you use the class factory registry to create a new component, the registry calls the component's **IWizardComponent->SetContainer** method to provide your

component access to the wizard services.

WizardPageImpl Template Class

Use this class as the base class for your custom pages—for example:

```
C++

class LocationPage :public WizardPageImpl<IDD_LOCATION_PAGE>
```

The parameter is the resource ID for your dialog box template.

Wizard Page Interfaces

The UDI Wizard uses interfaces to access the different controls on your page. Within you page, you use the **GetControlWrapper** function to retrieve a control wrapper. Here is an example:

```
C++

PStaticText pFormat;
GetControlWrapper(View(), IDC_CHECK_PARTITION, CONTROL_STATIC_TEXT,
&pFormat);
```

Here, **PStaticText** is a smart pointer to the **IStaticText** interface. Smart pointers automatically call the COM **Release()** method when they go out of scope or you pass the address of a variable (like **&pFormat**) to a method.

IADHelper Interface

```
C++

__interfaceIADHelper : IUnknown
{
    HRESULT Init(ILogger *pLogger);
    HRESULT ValidLogon(LPCTSTR userName, LPCTSTR password, LPCTSTR domain);
    HRESULT HasAccess(LPCTSTR username, LPCTSTR password, LPCTSTR domain,
LPCTSTR computerName, LPCTSTR accountDomain);
};
```

HRESULT Init(ILogger *pLogger)

Initialize this component, passing it to the logger so that it can log information.

HRESULTValidLogon(LPCTSTR userName, LPCTSTR password, LPCTSTR domain)

This method verifies whether a set of credentials is valid, as shown in Table 10.

Table 10. HResultValidLogon

Expand table

HResult	Description
S_OK	Credentials are valid
S_FALSE	Credentials are not valid
E_FAIL	Could not locate the domain controller; check logs for details

HRESULT HasAccess(LPCTSTR username, LPCTSTR password, LPCTSTR domain, LPCTSTR computerName, LPCTSTR accountDomain)

This method verifies whether a set of credentials has read/write access to the computer object in AD DS, as shown in Table 11.

Table 11. HResult HasAccess

Expand table

HRESULT	Description
S_OK	The user has access
E_FAIL	The user does not have access. Check the log file for additional information.

IBackgroundTask Interface

C++

```
__interface IBackgroundTask : IUnknown
{
    HRESULT Init(ITask *pTask, int id, IBackgroundCallback *pCallback);
    void Start(void);
```

```
    BOOL Running(void);
    HRESULT Wait(DWORD waitMilliseconds);
    HRESULT Terminate(DWORD exitCode);
    HRESULT GetExitCode(LPDWORD pCode, HRESULT *pHresult);
    HRESULT Close(void);
};
```

Overview

The **Progress** page uses this class to run tasks on a separate thread. You can also use this class whenever you want to perform operations on a separate thread. *Tasks* are any class that supports the **ITask** interface.

This interface is implemented by the **ID_BackgroundTask** ("Microsoft.Wizard.BackgroundTask") component, defined in the IBackgroundTask.h interface.

HRESULT Init(ITask *pTask, int id, IBackgroundCallback *pCallback)

This interface initializes the component, as shown in Table 12.

Table 12. HRESULT Init

Expand table

Parameter	Description
pTask	Pointer to the class that contains the code you want to run on another thread
Id	A number you can use in the callback's **Finished** method to tell which task finished running; useful if you start several tasks with the same callback method
pCallback	A class that implements the **Finished** method, which is called whenever a task finishes running; the call to the **Finished** method will be on the background thread, not the UI thread

void Start(void)

This method starts the task on a background thread and returns the elements shown in Table 13.

Table 13. Return Background Thread

Expand table

Returns	Description
E_INVALIDARG	The task is already running, so you cannot start it right now.
E_FAIL	There was a problem starting the thread.
S_OK	The thread was started.

BOOL Running()

This method returns TRUE if the background task is currently running and FALSE if it is not running.

HRESULT Wait(DWORD waitMilliseconds)

This method waits until either the thread stops running or the number of milliseconds has elapsed.

HRESULT Terminate(DWORD exitCode)

This method kills the thread that is running (see Table 14 and Table 15). This process may take a short amount of time to finish after this method returns.

Table 14. HRESULT Terminate Exit Code

Expand table

Parameter	Description
exitCode	The exit code that will be sent to the Finished callback method, which will also be available from the **GetExitCode** method.

Table 15. Termination Codes

Expand table

Returns	Description
E_FAIL	The call to terminate failed.
S_OK	The request to terminate the thread succeeded.

HRESULT GetExitCode(LPDWORD pCode, HRESULT *pHresult)

Use this method to get the results of running the task on the background thread (see Table 16).

Table 16. Result Codes

Expand table

Parameter	Description
pCode	Pointer to a **DWORD** that will be set on return or **nullptr** if you do not need the return value. On exit, this parameter is set to **STILL_ACTIVE** if the thread is running, the code returned by the task's **Execute** method, or the value passed to the **Terminate** method if you called that method.
pHresult	Pointer to an **HRESULT** that will be set on return or **nullptr** if you do not need the **HRESULT** value.

HRESULT Close(void)

This method releases the background thread. It returns **E_INVALIDARG** if the thread is currently running and **S_OK** otherwise.

ICheckBox Interface

C++

```
__interface ICheckBox : IControl
{
    void Check(BOOL check);
    BOOL IsButtonChecked();
};
```

void Check(BOOL check)

Set the checked state of the check box. When the method is TRUE, the check box is selected; when the method is FALSE, the check box is cleared.

BOOL IsButtonChecked()

This method reports the current check state of a check box.

IComboBox Interface

```
C++
```

```
__interface IComboBox : IControl
{
    HRESULT Bind([in] IBindableList *pList);
    HRESULT Select(int index);
    int Selected(void);
    void Add([in] LPCTSTR caption);
    HRESULT GetText([out, retval] LPBSTR pText);
    void Clear();
};
```

Overview

This interface is implemented by the **CheckBoxWrapper** component. You retrieve an instance of this component using the **GetControlWrapper** helper function with the type **CONTROL_COMBO_BOX**.

HRESULT Bind([in] IBindableList *pList)

Use this method when you have a data source that implements the **IBindableList** interface. The list box initializes the contents with the captions from this list.

HRESULT Select(int index)

Select the item in the combo box at the index.

int Selected(void)

This method returns the index of the selected item or **-1** if nothing is selected.

void Add([in] LPCTSTR caption)

Manually add an item to the combo box.

HRESULT GetText([out, retval] LPBSTR pText)

Retrieve the string of the currently selected item in the combo box.

void Clear()

Remove all the items from the combo box.

IControl Interface

```
C++

__interface IControl : IUnknown
{
    HRESULT SetEnable(BOOL enable);
    BOOL IsEnabled(void);
    HRESULT SetVisible(BOOL visible);
};
```

Overview

This interface is implemented by the **ControlWrapper** component. You retrieve an instance of this component using the **GetControlWrapper** helper function with the type **CONTROL_GENERIC**.

HRESULT SetEnable(BOOL enable)

Enable or disable the control.

BOOL IsEnabled(void)

Returns TRUE if the control is enabled, FALSE if it is not.

HRESULT SetVisible(BOOL visible)

Show or hide the control.

ICpuInfo Interface

```
C++

__interface ICpuInfo : IUnknown
{
    BOOL Is64Bit(void);
};
```

Overview

You obtain this interface by creating a new **ID_CpuInfo** component. The single method reports whether the CPU is 32 or 64 bit. Note that if you have a 32-bit operating system on a 64-bit computer, this method returns TRUE, because it is only reporting the width of the CPU (not the operating system).

IDirectory Interface

```
C++

__interface IDirectory : IUnknown
{
    BOOL FileExists(LPCWSTR name);
    BOOL FindFirst([in] LPCWSTR name);
    HRESULT FoundName([out, retval] LPBSTR name);
    DWORD FoundAttributes(void);
    BOOL FindNext(void);
    void FinishFind(void);
};
```

Overview

The **Directory** component, which you create using **ID_Directory**, provides a façade for working with directories in the file system.

BOOL FileExists(LPCWSTR name)

This method returns TRUE if a file with the name you provide exists.

BOOL FindFirst([in] LPCWSTR name)

This method finds a first match for the name you provide. It supports wildcard characters and returns both file and directory names. The method returns TRUE if a match was found, FALSE otherwise.

HRESULT FoundName([out, retval] LPBSTR name)

This method retrieves the name of the file found with a call to **FindFirst** or **FindNext**.

DWORD FoundAttributes(void)

This method returns the attribute for the most recent found file or directory. You can use code as follows to test whether it is a directory:

```
C++

pDirectory->FoundAttributes() & FILE_ATTRIBUTE_DIRECTORY
```

BOOL FindNext(void)

Find the next. This method returns TRUE if another match was found, FALSE otherwise.

void FinishFind(void)

This method releases resources used for the Find operation.

IDomainJoinValidator Interface

```
C++

__interface IDomainJoinValidator : IUnknown
{
    HRESULT Init(ILogger *pLogger, IWizardPageContainer *pContainer,
IStaticText *pUsername, IStaticText *pPassword, IStaticText *pComputerName);
    HRESULT IsUsernameValid(LPCWSTR domainName);
    BOOL CanModifyComputerAdEntry(LPCWSTR domainName);
};
```

Overview

You obtain an instance of this interface using the **ID_DomainJoinValidator** value to the **CreateInstance** template function.

HRESULT Init(ILogger *pLogger, IWizardPageContainer *pContainer, IStaticText *pUsername, IStaticText *pPassword, IStaticText *pComputerName)

Initialize the instance, as shown in Table 17.

Table 17. HRESULT Init - Instance Initialization

Expand table

Parameter	Description
pLogger	The logger instance, which is available to your page via the page's **Logger** method
pContainer	Passes the results from your page's **Container** method
pUsername	The text box that contains the user name to be validated
pPassword	The text box that contains the password to be validated
PComputerName	The text box that contains the name of the computer that will eventually be joined to the domain

HRESULT IsUsernameValid(LPCWSTR domainName)

This method uses the **IADHelper->ValidLogon** method to do the work. See that method for details.

BOOL CanModifyComputerAdEntry(LPCWSTR domainName)

Verify whether the user has rights to modify the computer entry. Most of the work is done by **IADHelper->HasAccess**. If this method returns FALSE, check the log file for details.

IDriveList Interface

C++

```
__interface IDriveList : IUnknown
{
    HRESULT Init(IWmiRepository *pWmi);
    HRESULT SetWhereClause(LPCTSTR whereClause);
    HRESULT SetMinimumDriveSize(__int64 size);
    HRESULT Update(void);
    HRESULT AddProperty(ENUM_DISK_QUERY_SECTION section, LPCTSTR propName,
LPCTSTR propNameReturned);

    size_t Count(void);
    HRESULT GetProperty(size_t index, LPCTSTR propName,  LPVARIANT value);
    HRESULT GetCaption(size_t index,  LPBSTR pCaption);
}
```

HRESULT Init(IWmiRepository *pWmi)

Call this method before you call any other components. You will need to create a new **WmiRepository** before you call this method.

HRESULT SetWhereClause(LPCTSTR whereClause)

This method allows you to add text that will appear as a "where" clause in the query. For example, the following line returns only USB drives:

```
C++

pDrives->SetWhereClause(L"WHERE InterfaceType='USB'");
```

HRESULT SetMinimumDriveSize(__int64 size)

Set the minimize drive size, in bytes, for drives that will be returned from the query.

HRESULT Update(void)

Execute the query. The drive list available after calling this method is sorted by drive letter.

HRESULT AddProperty(ENUM_DISK_QUERY_SECTION section, LPCTSTR propName, LPCTSTR propNameReturned)

This method adds the names of additional properties that you want to make available in the query results. Call this method before calling **Update**. Table 18 shows three of the useful properties.

Table 18. HRESULT AddProperty: Useful Properties

Expand table

Section	Property	Description
DISKQUERY_LOGICALDISK	**Size**	The size, in bytes, represented as a string
DISKQUERY_DISKPARTITION	**DiskIndex**	The disk number as an integer, starting with 0
DISKQUERY_LOGICALDISK	**VolumeName**	The volume label

size_t Count(void)

The number of records the query returns. Call **Update** before you call this method.

HRESULT GetProperty(size_t index, LPCTSTR propName, LPVARIANT value)

This method retrieves the value of a property from the query results, as shown in Table 19.

Table 19. HRESULT GetProperty

Expand table

Parameter	Description
Index	Zero-based index to the result record
propName	Name of the property, such as "Size"
Value	On return, this parameter contains a variant value of the property

HRESULT GetCaption(size_t index, LPBSTR pCaption)

This method retrieves the caption for a record which is the same as the **Caption** property.

IImageList Interface

C++

```
__interface IImageList
{
    HRESULT CreateImageList(int width, int height, UINT flags);
    HImageList GetImageList(void);
    int AddImage(HInstance hInstance, int resourceId);
};
```

Overview

This interface is implemented by the **ImageList** component. You retrieve an instance of this component from the **IListView** interface.

HRESULT CreateImageList(int width, int height, UINT flags)

Create a new image list, which this component manages. Call this method only once.

HImageList GetImageList(void)

This method returns the handle for the image list in case you need to perform other operations on the image list.

int AddImage(HInstance hInstance, int resourceId)

Add a new image to the image list from a resource, as shown in Table 20.

Table 20. HRESULT IImageList Interface

Expand table

Parameter	Description
hInstance	Instance handle of the module that contains the bitmap resource
resourceId	ID of the resource to load into the image list

IListView Interface

C++

```
__interface IListView : IControl
{
    int AddItem([in] LPCTSTR text);
    int AddColumn(int width, [in] LPCTSTR text);
    HRESULT SetSubItem(int index, int column, [in] LPCTSTR text);
    int GetWidth(void);
    void SetExtendedStyle(DWORD style);
    int GetSelectedItem(void);
    HRESULT SelectItem(int index);
    BOOL IsItemChecked(int index);
    int GetItemCount(void);
    HRESULT CreateImageList(int width, int height, UINT flags);
    int AddImage(HINSTANCE hInstance, int resourceId);
    HRESULT SetImage(int index, int imageIndex);
    HRESULT Clear(void);
};
```

Overview

This interface is implemented by the **ControlWrapper** component. You retrieve an instance of this component using the **GetControlWrapper** helper function with the type **CONTROL_LIST_VIEW**.

int AddItem([in] LPCTSTR text)

Add a new row to the list box. The method returns the index of the item just added.

int AddColumn(int width, [in] LPCTSTR text)

Add a new column to the list view.

HRESULT SetSubItem(int index, int column, [in] LPCTSTR text)

Set the text in a column other than the first column of the list box, as shown in Table 21.

Table 21. HRESULT SetSubItem

Expand table

Parameter	Description
index	The index of the list item you want to modify
column	The index of the column you want to update; the first column is set with **AddItem**, columns two and following are set with this method
text	The string to show in the column

int GetWidth(void)

This method returns the width of the entire text box.

void SetExtendedStyle(DWORD style)

This method allows you to set extended styles on the list box—for example:

```
C++

m_pList->SetExtendedStyle(LVS_EX_FULLROWSELECT);
```

int GetSelectedItem(void)

This method returns the index of the list view item currently selected.

HRESULT SelectItem(int index)

Set the selected item in the list to this index.

BOOL IsItemChecked(int index)

This method returns TRUE if an item in the list is selected. This method requires that you call **SetExtendedStyle** to set the check box style.

int GetItemCount(void)

This method returns the number of items in the list view.

HRESULT CreateImageList(int width, int height, UINT flags)

Create a new image list, and attach it to the list view.

int AddImage(HINSTANCE hInstance, int resourceId)

Add an image to the list view's image list. You need to call **CreateImageList**, first.

HRESULT SetImage(int index, int imageIndex)

Set the image that will be shown on the left side for a specific list view item.

HRESULT Clear(void)

Remove all items from the list view.

IProgressBar Interface

C++

```
__interface IProgressBar : IControl
{
    HRESULT SetPercentage(int position);
    int GetPercentage(void);
};
```

Overview

This interface is implemented by the **ProgressBarWrapper** component. You retrieve an instance of this component using the **GetControlWrapper** helper function with the type **CONTROL_PROGRESS_BAR**.

HRESULT SetPercentage(int position)

Set the position of the progress bar using a number between 0 and 100. By default, new Win32® progress bars have a maximum range of 100.

int GetPercentage(void)

This method returns the current position of the progress bar.

IRadioButton Interface

```
C++
```

```
__interface IRadioButton : IControl
{
public:
    void SetGroup(int firstId, int lastId);
    void CheckRadio(int id);
    BOOL IsButtonChecked(int id);
    void EnableRadio(int id, BOOL enable);
};
```

Overview

This interface is implemented by the **RadioButtonWrapper** component. You retrieve an instance of this component using the **GetControlWrapper** helper function with the type **CONTROL_RADIO_BUTTON**.

void SetGroup(int firstId, int lastId)

Provide the wrapper with the range of radio buttons that should be treated as a group. Call this method before you call **CheckRadio**.

void CheckRadio(int id)

Set the specific radio button to be the single button in the group of radio buttons selected. Call **SetGroup** before calling this method.

BOOL IsButtonChecked(int id)

This method returns TRUE if the radio button is currently selected, FALSE otherwise.

void EnableRadio(int id, BOOL enable)

This method enables or disables a radio button.

IStaticText Interface

```
C++
```

```
__interface IStaticText : IControl
{
    HRESULT SetText([in] LPCTSTR pText);
    HRESULT GetText([out, retval] LPBSTR pText);
};
```

Overview

This interface is implemented by the **StaticTextWrapper** component. You retrieve an instance of this component using the **GetControlWrapper** helper function with the type **CONTROL_STATIC_TEXT**.

HRESULT SetText([in] LPCTSTR pText)

Set the text for the control.

HRESULT GetText([out, retval] LPBSTR pText)

This method returns the current value of the text for the control.

ITask Interface

```
C++
```

```
__interface IControl : IUnknown
{
    HRESULT Init(IStringProperties *pProperties, ISettingsProperties
```

```
*pTaskSettings);
    HRESULT Execute(LPDWORD pReturnCode);
};
```

Implement this interface if you want your component to be available as a task in the preflight page or if you want to use the **BackgroundTask** component to perform work on a background thread.

Here are components that implement the **ITask** interface:

- ID_ShellExecuteTask, L"Microsoft.Wizard.ShellExecuteTask"
- ID_CopyFilesTask, L"Microsoft.Wizard.CopyFilesTask"
- ID_ACPowerTask, L"Microsoft.OSDRefresh.ACPowerTask"
- ID_WiredNetworkTask, L"Microsoft.SharedPages.WiredNetworkTask"

Init

C++

```
HRESULT Init(IStringProperties *pProperties, ISettingsProperties
*pTaskSettings)
```

If you are writing a task for the preflight page, call this method to initialize your task. The .config file contain XML that might look something like this:

XML

```
<Task DisplayName="Check Windows Scripting Host"
Type="Microsoft.Wizard.ShellExecuteTask">
  <Setter Property="filename">%windir%\system32\cscript.exe</Setter>
  <Setter Property="parameters">Preflight\OSDCheckWSH.vbs</Setter>
  <Setter Property="BitmapFilename">images\WinScriptHost.bmp</Setter>
  <ExitCodes>
    <ExitCode State="Success" Type="0" Value="0" Text="" />
    <ExitCode State="Error" Type="-1" Value="*" Text="Windows Scripting Host
not installed." />
  </ExitCodes>
</Task>
```

The **pProperties** parameter provides access to the three setter values, whereas the **pTaskSettings** parameter provides access to the **Task** element and children. Most tasks only need to read data from the **pProperties** parameter.

Execute

```
C++

HRESULT Execute(LPDWORD pReturnCode)
```

Here is where you write the code that performs the task. This method should return **S_OK** if there were no errors, and it can return another **HRESULT** if an error occurred while the task was running. Values other than **S_OK** that this method returns are matched up to <Error> elements in the <ExitCodes> section if you are using the preflight page.

The **pReturnCode** parameter must be updated with a number that reports the state of the task. These values are matched by the preflights page to <ExitCode> elements.

ITreeView Interface

```
C++

__interface ITreeView : IControl
{
    void EnableCheckboxes(void);
    HRESULT CreateImageList(int width, int height, UINT flags);
    int AddImage(HINSTANCE hInstance, int resourceId);

    HTREEITEM AddItem(LPCTSTR text, HTREEITEM hParent = NULL);
    void SetImage(HTREEITEM item, int image, int expandImage);

    void Clear(void);
    BOOL SetFirstVisible(HTREEITEM item);
    BOOL SelectItem(HTREEITEM item);
    void CheckItem(HTREEITEM item, UINT checkState);
    HTREEITEM SelectedItem(void);
    int SetItemHeight(SHORT height);
    HRESULT EnableItem(HTREEITEM item, BOOL enable);
    void Expand(HTREEITEM hItem, BOOL expand);

    HTREEITEM GetChild(HTREEITEM hParent);
    HTREEITEM GetParent(HTREEITEM hNode);
    HTREEITEM GetNextItem(HTREEITEM hPrevious);

    UINT IsChecked(HTREEITEM item);
    BOOL IsEnabled(HTREEITEM item);

    INT_PTR CommonControlEvent(WORD controlId, void* pInfo, BOOL *pCancel);
    HRESULT SetEventHandler(ITreeViewEvent *pEventHandler);
```

```
    void SetSelectedBackColor(COLORREF color);
};
```

Overview

This interface is implemented by the **TreeViewWrapper** component. You retrieve an instance of this component using the **GetControlWrapper** helper function with the type **CONTROL_TREE_VIEW**.

void EnableCheckboxes(void)

This method turns on check boxes in the tree view control by setting the **TVS_CHECKBOXES** style.

HRESULT CreateImageList(int width, int height, UINT flags)

Add a new image list to the tree view control. The **flags** parameter is passed in the call to the **ImageList_Create** Win32 function.

int AddImage(HINSTANCE hInstance, int resourceId)

Add an image to the image list from a resource (**resourceId**) in the module with the instance handle **hInstance**.

HTREEITEM AddItem(LPCTSTR text, HTREEITEM hParent = NULL)

Add a node to the tree view. The new node will be added at the top level if **hParent** is NULL. Otherwise, provide the handle to the parent item where you want the new item added. This method returns the handle to the new item.

void SetImage(HTREEITEM item, int image, int expandImage)

Set the image to use for a tree view item. You can set both the normal and the expanded image.

void Clear(void)

Remove all items from the tree view.

BOOL SetFirstVisible(HTREEITEM item)

Ensure that the tree view item is visible. The tree view will scroll if required to make this item visible.

BOOL SelectItem(HTREEITEM item)

Set the currently selected item to the item that you provide. You can call **SetFirstVisible** after this to ensure that the newly selected item is visible.

void CheckItem(HTREEITEM item, UINT checkState)

The method basically sets the image that will be shown for the check box in the tree view. These images are in a separate **ImageList** control that the tree view manages. By default, this image list has three images in it, shown in Table 22.

Table 22.void CheckItem Image List Default

Expand table

checkState	Description
0	Blank
1	Cleared
2	Selected

HTREEITEM SelectedItem(void)

This method returns the handle of the tree view item currently selected.

int SetItemHeight(SHORT height)

This method sets the height of all items in the tree view control in pixels. It returns the previous height in pixels.

HRESULT EnableItem(HTREEITEM item, BOOL enable)

This method enables or disables a single item in the tree. Disabling an item with children will not disable the children.

void Expand(HTREEITEM hItem, BOOL expand)

This method expands or collapses a node in the tree.

HTREEITEM GetChild(HTREEITEM hParent)

This method returns the first child of a tree view item or NULL if there are no children.

HTREEITEM GetParent(HTREEITEM hNode)

This method returns the handle of the parent for a node in the tree view or NULL if the node is at the top level.

HTREEITEM GetNextItem(HTREEITEM hPrevious)

You can call this method with a handle that **GetChild** returns to iterate through all the children of a node. This method returns the next sibling in the tree that shares the same parent.

UINT IsChecked(HTREEITEM item)

This method returns **0** if the tree view node is not selected and **1** if it is.

BOOL IsEnabled(HTREEITEM item)

This method returns TRUE if the tree view node is enabled, FALSE otherwise.

INT_PTR CommonControlEvent(WORD controlId, void* pInfo, BOOL *pCancel)

This method is for internal use only.

HRESULT SetEventHandler(ITreeViewEvent *pEventHandler)

Call this method if you want to receive notification when the selected item changes or the user changes the check state of a tree view item. You must implement the **ITreeViewEvent** in your component to receive these callbacks.

void SetSelectedBackColor(COLORREF color)

Set the background color used for the selected item.

IWmiIteration Interface

```
C++
```

```
__interface IWmiIterator : IUnknown
{
    HRESULT Next(void);
    HRESULT GetProperty(LPCTSTR propertyName, [out] LPVARIANT pValue);
};
```

Overview

You typically use this interface, along with **IWmiRepository**, when working with WMI calls. The **IWmiIteration** interface allows you to iterate through the values that a query returns.

HRESULT Next(void)

Move to the next item in the query results, as shown in Table 23.

Table 23. HRESULT Next(void) Query Returns

Expand table

HRRESULT	Description
S_OK	Moved to the next result; you can use **GetProperty** to retrieve properties of that result.
S_FALSE	There are no more items in the list.
E_NOT_SET	There are no query results

HRESULT GetProperty(LPCTSTR propertyName, [out] LPVARIANT pValue)

This method retrieves the value of a property from the current result record, as shown in Table 24 and Table 25.

Table 24. HRESULT GetProperty

Expand table

Parameter	Description
propertyName	Name of the property you want to retrieve
pValue	Points to a VARIANT structure that on return contains the property value

Table 25. HRESULT GetProperty Result

Expand table

HRESULT	Description
S_OK	Property value was retrieved.
WBEM_E_NOT_FOUND	There is no property with the name.
E_NOT_VALID_STATE	There is no current record.

> **Note**
>
> The **GetProperty** method can return other WMI error codes other than those listed in Table 25. The values listed are the common results that are returned.

IWmiRepository Interface

C++

```
__interface IWmiRepository : IUnknown
{
    HRESULT SetNamespace(LPCWSTR namespaceName);
    HRESULT ExecQuery(LPCWSTR query, [out] IWmiIterator **ppIterator);
};
```

Overview

This interface is implemented by the **WmiRepository** component (**ID_WmiRepository**).

HRESULT SetNamespace(LPCWSTR namespaceName)

This method sets the WMI namespace that will be used for the query. Call this method before you call **ExecQuery**. If you do not call this method, the namespace will be root\cimv2. This method always returns **S_OK**.

HRESULT ExecQuery(LPCWSTR query, [out] IWmiIterator **ppIterator)

Execute a query against the WMI namespace set with a call to **SetNamespace**, as shown in Table 26 and Table 27.

Table 26. HRESULT ExecQuery

Expand table

Parameter	Description
Query	The string for the WMI query you want to execute
ppIterator	Pass a pointer to an interface pointer, which on return will be filled in with an interface, giving you access to the query results

Table 27. HRESULT Query Result

Expand table

HRESULT	Description
S_OK	Query succeeded
Other	If the query did not succeed, returns a WMI **HRESULT**

IFormController Interface

C++

```
__interface IFormController : IUnknown
{
    Init(IWizardPageView *pView, IWizardPageContainer *pContainer);
    SetPageInfo(ISettingsProperties *pPageInfo);

    Validate(void);

    AddToGroup(int groupControlId, int controlId);
    UpdateCheckGroup(int groupControlId);
    AddValidator(int controlId, IValidator *pValidator, IControl *pCOntrol =
0);

    AddValidator(int controlId, LPCWSTR validatorId, LPCWSTR message,
IValidator **ppValidator = nullptr);
    DisableValidation(int controlId, BOOL disable);
```

```
    AddField(LPCWSTR fieldName, int controlId, BOOL suppressLog,
DialogControlTypes type);
    AddRadioGroup(LPCWSTR groupName, int radioControlId);
    EnableRadioGroup(LPCWSTR groupName, BOOL enable);
    InitFields(IFieldCallback *pFieldCallback = nullptr);
    SaveFields(IFieldCallback *pFieldCallback = nullptr);
    BOOL IsFieldDisabled(int controlId);

    InitSection(LPCWSTR key, LPCWSTR sectionCaption);
    AddSummaryItem(LPCWSTR first, LPCWSTR second);
    SuppressLogValue(LPCWSTR tsVariableName);
    SaveText(int controlId, LPCWSTR tsVariableName, LPCWSTR summaryCaption);
    LoadText(int controlId, LPCWSTR tsVariableName);

    void ControlEvent(WORD eventId, WORD controlId);
    BOOL IsValid(void);
 };
```

Overview

Each page in the UDI Wizard has its own form controller that implements this interface. You use this controller to connect the field data in the .config XML file to the controls on your page. The form controller then handles many of the details for you.

Setting up the Form

Generally, set up the form controller in your page's **OnWindowCreated** method. Doing so usually involves calling the methods shown in Table 28.

Table 28. OnWindowCreated Method

Expand table

Method	Description
Init	Initializes the form controller
AddField	Provides a connection between a field in the .config XML file that is a string name and a control in your page's dialog box that is an ID
AddRadioGroup	Used to connect a radio button to both a group and a control in your dialog box
AddToGroup	Allows you "child" controls that are enabled or disabled along with their parent or based on which radio button is selected
InitFields	Call after you have called all the **Add** methods to set up the form

Method	Description
Validate	Performs the initial validation

Processing Form Events

Add the following call to your **OnControlEvent** method:

```
C++

Form()->ControlEvent(eventId, controlId);
```

This call passes events on to the form controller so it can process form-related events.

Save Form Data

In the **OnNextSelected** method, call the form methods shown in Table 29.

Table 29. OnNextSelected Method

Expand table

Method	Description
InitSection	Provides the name of the section that will be shown on the **Summary** page for this page
SaveFields	Save field values to task sequence variables and to the **Summary** page

Init

```
C++

HRESULT Init(IWizardPageView *pView, IWizardPageContainer *pContainer)
```

You usually call this method near the start of your page's **OnWindowCreated** method. The command should look something like this:

```
C++

Form()->Init(View(), Container());
```

SetPageInfo

```
C++

HRESULT SetPageInfo(ISettingsProperties *pPageInfo)
```

This method is called internally, and you should not call it yourself. It provides the page's XML to the form controller.

Validate

```
C++

HRESULT Validate(void)
```

This method executes all the validators attached to controls. If a validator does not pass, the form controller displays a warning message and disables the **Next** button, then stops processing validators. Typically, you only need to call this method at the end of your **OnWindowCreated** method; it always returns **S_OK**.

AddToGroup

```
C++

AddToGroup(int groupControlId, int controlId)
```

This method adds a control as a "child" of a check box or radio button, as shown in Table 30. All such child controls will be disabled when the parent control is not selected. The method always returns **S_OK**.

Table 30. AddToGroup

Expand table

Parameter	Description
groupControlId	The ID of the check box or radio button that will control the enable state of the child control
ControlId	The ID of the control that you want to add as a child

UpdateCheckGroup

```
C++

HRESULT UpdateCheckGroup(int groupControlId)
```

This method updates the enable or disable status of a group's child controls based on the status of the parent control. Generally, you do not need to call this method yourself, because the form controller calls it for you.

AddValidator

```
C++

HRESULT AddValidator(int controlId, IValidator *pValidator, IControl
*pControl = 0)
```

Call this method only if you have a validator you want to create in code instead of with the XML. This method always returns **S_OK**.

AddValidator

```
C++

HRESULT AddValidator(int controlId, LPCWSTR validatorId, LPCWSTR message,
IValidator **ppValidator = nullptr)
```

Call this method only if you have a validator you want to create in code instead of with the XML.

DisableValidation

```
C++

HRESULT DisableValidation(int controlId, BOOL disable)
```

Call this method to either explicitly disable validator for a control or restore normal validation, as shown in Table 31. This method is useful, for example, when you have enable/disable rules for controls that are not covered with form validation and you need to disable validation for a control. In other words, you would not normally call this method. This method always returns **S_OK**.

Table 31. HRESULT DisableValidation

Expand table

Parameter	Description
controlId	The control for which you want to enable or disable validation
Disable	Set to TRUE to disable validation and to FALSE to restore normal validation

AddField

```
C++

HRESULT AddField(LPCWSTR fieldName, int controlId, BOOL suppressLog,
DialogControlTypes type)
```

Add a control mapping between the name in a **Field** element of the .config XML file and the control ID in your page's dialog box, as shown in Table 32. You must call this method before the call to **InitFields**, because **InitFields** uses this information. This method always returns **S_OK**.

Table 32. HRESULT AddField

Expand table

Parameter	Description
Fieldname	Name of the field as it appears in your page's XML
controlId	The ID of the control in your page's dialog box template
suppressLog	Set to TRUE if you do not want the values from this field written to the log file; always set this parameter to TRUE for password or PIN fields
Type	The type of control, which is one of the following: - **CONTROL_STATIC_TEXT** - **CONTROL_COMBO_BOX** - **CONTROL_LIST_VIEW** - **CONTROL_PROGRESS_BAR** - **CONTROL_GENERIC** - **CONTROL_RADIO_BUTTON** - **CONTROL_CHECK_BOX** - **CONTROL_TREE_VIEW**

AddRadioGroup

```
C++

HRESULT AddRadioGroup(LPCWSTR groupName, int radioControlId)
```

This method adds a control to a named radio button group, as shown in Table 33. You must call this before the **InitFields** method, because that method uses attributes on the **RadioGroup** element to control settings for all the radio button controls in the group. Radio groups can be locked, for example, so that all the radio buttons are disabled, but child controls are enabled or disabled based only on which radio button is selected. This method always returns **S_OK**.

Table 33. HRESULT AddRadioGroup

Expand table

Parameter	Description
groupName	A string that defines a group of radio buttons on this page
radioControlId	The ID of a single radio button to add to this group

EnableRadioGroup

```
C++

HRESULT EnableRadioGroup(LPCWSTR groupName, BOOL enable)
```

This method allows you to enable or disable an entire radio button group. Disabling a radio group disables all the radio button controls in the group as well as any children of those radio buttons that have been added with **AddToGroup**. See Table 34 and Table 35.

Table 34. EnableRadioGroup

Expand table

Parameter	Description
groupName	Name of a radio button group that you defined already with a call to **AddRadioGroup**
Enable	Set to TRUE to enable the radio button group and FALSE to disable the group

Table 35. HRESULT EnableRadioGroup

Expand table

HRESULT	Description
S_OK	Group enabled or disabled
E_INVALIDARG	There is no radio button group with the name you provided

InitFields

```
C++

HRESULT InitFields(IFieldCallback *pFieldCallback = nullptr)
```

Before calling this method, call **AddField** for each field that the XML can control. This method always returns **S_OK**.

The **pFieldCallback** parameter is optional. If you provide it, the form controller calls **SetFieldDefault** for controls that are not either **CONTROL_STATIC_TEXT** or **CONTROL_CHECK_BOX**. This behavior allows you to retrieve a default value from the XML and set it in the control yourself.

SaveFields

```
C++

HRESULT SaveFields(IFieldCallback *pFieldCallback = nullptr)
```

This method saves field values to task sequence variables and to the summary data that will be shown on the **Summary** page. Providing a pointer in **pFieldCallback** allows you to handle saving values for controls that do not support **CONTROL_STATIC_TEXT**.

IsFieldDisabled

```
C++

BOOL IsFieldDisabled(int controlId)
```

This method allows you to determine whether a field has been disabled in the XML.

InitSection

```
C++

HRESULT InitSection(LPCWSTR key, LPCWSTR sectionCaption)
```

This method initializes the summary data that will be shown on the **Summary** page, as shown in Table 36. Call this method in your **OnNextSelected** method before calling **SaveFields**. This method always returns **S_OK**.

Table 36. HRESULT InitSection

Expand table

Parameter	Description
Key	This parameter should be unique to your page. It is used to ensure that each page has its own summary information.
sectionCaption	The header that will be shown on the **Summary** page for this page's summary information. Typically, you use **DisplayName()** as the value for this parameter.

AddSummaryItem

```
C++

HRESULT AddSummaryItem(LPCWSTR first, LPCWSTR second)
```

This method allows you to add summary items to the **Summary** page above and beyond those items set with the XML. See Table 37.

Table 37. HRESULT AddSummaryItem

Expand table

Parameter	Description
First	The caption for the summary item, which is shown on the left side
Second	The value that will be shown on the right side

SuppressLogValue

```
C++

HRESULT SuppressLogValue(LPCWSTR tsVariableName)
```

Call this method for task-sequence variables for which you do not want the values to be written to the log file. Call this method for task sequence variables that store passwords, PINs, or other sensitive values a user might enter.

SaveText

```
C++

HRESULT SaveText(int controlId, LPCWSTR tsVariableName, LPCWSTR
summaryCaption)
```

This method saves the value of a text control to both a task sequence variable and the summary section. Typically, you will not need to call this method yourself, because the form controller does this for all fields. See Table 38.

Table 38. HRESULT SaveText

Expand table

Parameter	Description
controlId	The ID of the text box that contains the value you want to save (or any other control that can return text)
tsVariableName	Name of the task sequence variable that you want to modify
summaryCaption	The caption on the **Summary** page for this value

LoadText

```
C++

HRESULT LoadText(int controlId, LPCWSTR tsVariableName)
```

This method reads the value of a task sequence variable and sets the text box to this value.

ControlEvent

```
C++

void ControlEvent(WORD eventId, WORD controlId)
```

Call this method on your **OnControlEvent** method to ensure that the form controller can process control events, which it needs to do to function correctly. The values you pass to this method are the same values passed to the **OnControlEvent** method.

IsValid

```
C++

BOOL IsValid(void)
```

This method returns the status of the most recent validation of the form. If any of the control validators reported an error, this method returns FALSE. In other words, it returns TRUE only if all the controls on the page are valid.

IValidator Interface

```
C++

__interface IValidator : IUnknown
{
    HRESULT Init(IControl *pControl, LPCTSTR message);
    HRESULT Init(IControl *pControl, IWizardPageContainer *pContainer,
IStringProperties *pProperties);
    BOOL, IsValid(LPBSTR pMessage);
    HRESULT SetProperty(int propertyId, LPVARIANT pValue);
    HRESULT SetProperty(int propertyId, IUnknown *pUnknown);
    HRESULT SetProperty)(int propertyId, LPCTSTR pValue);
};
```

Overview

Validators are components that can validate a single control on your page. The easiest way to implement a validator is to make it a subclass of the **BaseValidator** class, which is defined in the BaseValidator.h header file.

HRESULT Init(IControl *pControl, LPCTSTR message)

If you create a validator in code, you can call this method to initialize the validator. See Table 39.

Table 39. HRESULT Init

Expand table

Parameter	Description
pControl	The control that your validator must validate
Message	The message to display on the page if the control is not valid

HRESULT Init(IControl *pControl, IWizardPageContainer *pContainer, IStringProperties *pProperties)

The form controller calls this method to initialize validators that it creates based on the page's XML. See Table 40.

Table 40. HRESULT Init Method

Expand table

Parameter	Description
pControl	The control that your validator must validate
pContainer	In case your validator needs access to the logger or needs to create other components
pProperties	Provides access to the properties (setter elements) for your validator

BOOL, IsValid(LPBSTR pMessage)

This method returns TRUE if the control is valid or FALSE if the control is invalid. On return, **pMessage** should be filled in with a new **BSTR** that contains the message to display when the control is not valid.

HRESULT SetProperty(int propertyId, LPVARIANT pValue)

You can implement this method if you need extra values that are not provided in the XML.

HRESULT SetProperty(int propertyId, IUnknown *pUnknown)

You can implement this method if you need extra values that are not provided in the XML.

HRESULT SetProperty)(int propertyId, LPCTSTR pValue)

You can implement this method if you need extra values that are not provided in the XML.

IRegEx Interface

C++

```
__interface IRegEx : IUnknown
{
    BOOL MatchesRegex(LPCTSTR input, LPCTSTR regex);
    HRESULT GetMatch(size_t index, LPBSTR pValue);
};
```

This method is implemented by the **ID_Regex** component (IRegex.h) and provides support for regular expression processing.

BOOL MatchesRegex(LPCTSTR input, LPCTSTR regex)

This method runs the regular expression against the input text. It uses the C++ standard library's **regex_match** function to do the actual work. The method returns TRUE if there were matches, FALSE otherwise.

HRESULT GetMatch(size_t index, LPBSTR pValue)

This method allows you to retrieve the matches from the most recent **MatchesRegex** call. Note that there is no error processing in this method, and it either returns **S_OK** or throws an exception.

ISummaryInfo Interface

C++

```
__interface ISummaryInfo : IUnknown
{
    size_t Count(void);
    HRESULT Clear(void);
    HRESULT AddInfo(LPCTSTR pFirst, LPCTSTR pSecond);
    HRESULT GetInfo(size_t index, LPBSTR pFirst, LPBSTR pSecond);
```

```
    HRESULT GetCaption(LPBSTR pCaption);
    HRESULT SetCaption(LPCTSTR caption);
};
```

You should not need to use this interface directly. Instead, use **IFormController**.

ISummaryBag

```
C++

__interface ISummaryBag : IUnknown
{
    size_t Count(void);
    HRESULT GetInfoByIndex(size_t index, [out] ISummaryInfo **ppSummary);
    HRESULT GetInfoByKey(LPCTSTR key, [out] ISummaryInfo **ppSummary);
};
```

You should not need to use this interface directly. Instead, use **IFormController**.

ITSVariableBag Interface

```
C++

__interface ITSVariableBag : IUnknown
{
    void GetValue([in] LPCTSTR variableName, [out] LPBSTR pValue);
    void SetValue([in] LPCTSTR variableName, [in] LPCTSTR pValue);
    void Clear(void);
    HRESULT Remove([in] LPCTSTR variableName);
    HRESULT SuppressLogValue([in] LPCTSTR variableName);
    void Save(void);
};
```

This interface provides access to task sequence variables. You can access this interface using your page's **TSVariables()** method.

void GetValue([in] LPCTSTR variableName, [out] LPBSTR pValue)

This method reads the value of a task sequence variable.

> ⓘ **Note**
>
> Values are cached after the first read.

void SetValue([in] LPCTSTR variableName, [in] LPCTSTR pValue)

This method sets the value of a task sequence variable. This value is saved in memory. Task sequence values are written once you select **Finish** in the UDI Wizard.

void Clear(void)

This method removes all task sequence values that have been saved in memory.

HRESULT Remove([in] LPCTSTR variableName)

This method removes a specific task sequence value from memory. The next time you call **GetValue** with the same task sequence name, the method attempts to retrieve it from the task sequence.

HRESULT SuppressLogValue([in] LPCTSTR variableName)

Whenever task sequence variables are written, such as when you select **Finish** in the UDI Wizard, the names and values are written to the log file. Call this method to suppress logging of sensitive values, such as passwords or PINs, for a specific task sequence variable.

void Save(void)

This method saves all task sequence values that have been set with calls to **SetValue**.

ITSVariableRepository Interface

```
C++
```

```
__interface ITSVariableRepository : IUnknown
{
    void GetValue([in] LPCTSTR variableName, BOOL logValue, [out] LPBSTR
pValue);
    void SetValue([in] LPCTSTR variableName, BOOL logValue, [in] LPCTSTR
value);
};
```

This interface is for internal use by **TSVariableBag** for reading and writing task sequence variables.

IWizardFinish Interface

```
C++

__interface IWizardFinish : IUnknown
{
    HRESULT Canceled(void);
    HRESULT Finished(void);
};
```

This interface is useful in advanced scenarios where you want to perform additional processing when you select **Finish** or **Cancel** in the UDI Wizard. The UDI Wizard contains a **Finish** task that saves task sequence variables when you select **Finish**. If you cancel the wizard, the task only sets the **OSDSetupWizCancelled** task sequence variable to TRUE and does not save changes to any other task sequence variables.

If you create your own finish component, you need to register it with code like this:

```
C++

Register<MyFinishTaskFactory>(ID_MyFinishTask, pRegistry);

PWizardFinish pFinish;
CreateInstance(pRegistry, ID_MyFinishTask, &pFinish);

PWizardFinishService pService;
GetService<IWizardFinishService>(pRegistry, &pService);

pService->Register(pFinish);
```

IBindableList Interface

```
C++

__interface IBindableList : IUnknown
{
    size_t Count(void);
    HRESULT GetCaption(size_t index, LPBSTR pCaption);
};
```

Implement this interface if you have a data source component that you want to bind to a combo box by calling its **Bind** method.

size_t Count(void)

This method returns the number of items in the list.

HRESULT GetCaption(size_t index, LPBSTR pCaption)

This method returns the caption of the item at a specific index.

IDataNodes Interface

C++

```
__interface IDataNodes : IUnknown
{
    size_t Count();
    HRESULT SetCaptionProperty(LPCTSTR captionProperty);
    HRESULT GetProperty(size_t index, LPCTSTR propertyName, [out] LPBSTR
propertyValue);
    HRESULT GetNode(size_t index, [out] ISettingsProperties **ppNode);
};
```

This interface provides access to hierarchical data that can be saved in a page. You obtain this interface via methods on the **ISettingsProperties** interface, which is available to your page through the **Settings** method.

Data in a page's XML can look something like this

XML

```
      <Data Name="Network">
        <DataItem>
          <Setter Property="DisplayName">Public</Setter>
          <Setter Property="Share">\\servername\Share</Setter>
        </DataItem>
        <DataItem>
          <Setter Property="DisplayName">Dev Team</Setter>
          <Setter Property="Share">\\servername\DevShare</Setter>
        </DataItem>
      </Data>
```

Calling **Settings()->GetDataNode(L"Network", &pData)** gives you an **IDataNodes** instance with two data items (each of which in turn has two properties).

size_t Count()

This method returns the number of **DataItem** elements.

HRESULT SetCaptionProperty(LPCTSTR captionProperty)

The component that supports this interface also supports **IBindableList**, which makes it easy to populate a combo box with data from the page's XML. This method controls which property (setter) in each **DataItem** element will be used for this binding. For example, you could call this method with **DisplayName**, and it would use that setter property for data binding. The combo box would then contain **Public** and **Dev Team** as items.

HRESULT GetProperty(size_t index, LPCTSTR propertyName, [out] LPBSTR propertyValue)

This method gets a property from one of the **DataItem** elements. See Table 41 and Table 42.

Table 41. DataItem GetProperty

Expand table

Parameter	Description
Index	The index value (starting with 0) of the **DataItem** for which you want to retrieve a property value
propertyName	Name of the setter property for which you want to retrieve a value
propertyValue	On return, contains the string value of a property

Table 42. HRESULT GetProperty

Expand table

HRESULT	Description
S_OK	The property was retrieved.
E_INVALIDARG	The index is past the end of the array.

HRESULT GetNode(size_t index, [out] ISettingsProperties **ppNode)

This method is similar to **GetProperty**, but instead of returning one value from a **DataItem**, it returns the entire **DataItem** wrapped in an **ISettingsProperties** interface. See Table 43 and Table 44.

Table 43. HRESULT GetNode

Expand table

Parameter	Description
Index	The index value (starting with 0) of the **DataItem** for which you want to retrieve a property value
ppNode	On exit, the **ISettingsProperties** interface that wraps the **DataItem** node

Table 44. HRESULT GetNode Results

Expand table

HRESULT	Description
S_OK	The node was retrieved.
E_INVALIDARG	The index is past the end of the array.

IFactoryRegistry Interface

C++

```
__interface IFactoryRegistry : IUnknown
{
    void Register(LPCTSTR type,  IClassFactory *pFactory);
    HRESULT LoadAndRegister(LPCTSTR dllName, ILogger *pLogger);
    BOOL Contains(LPCTSTR type);
    HRESULT GetFactory(LPCTSTR type,  IClassFactory **ppFactory);
    HRESULT CreateInstance(LPCTSTR type,  IUnknown **ppInstance);
    HRESULT SetContainer(IWizardPageContainer *pContainer);
    HRESULT RegisterService(REFGUID iid, IUnknown *pService);
    HRESULT GetService(REFGUID iid,  IUnknown **ppService);
};
```

Overview

When you create a new custom page, at a minimum you need to create a *page factory*—a class that implements **IClassFactory**. (You can use **ClassFactoryImpl** as a base class for your factory.)

void Register(LPCTSTR type, IClassFactory *pFactory)

This method registers a class factory with the registry. See Table 45.

Table 45. IClassFactory void Register

Expand table

Parameter	Description
Type	A string that identifies the factory you are registering; generally, this parameter should have your company name in the string to ensure that it is unique
pFactory	A pointer to your class factory instance

HRESULT LoadAndRegister(LPCTSTR dllName, ILogger *pLogger)

This method is for internal use only.

BOOL Contains(LPCTSTR type)

This method is generally for internal use. It checks to see whether a class factory has been registered for a type.

HRESULT GetFactory(LPCTSTR type, IClassFactory **ppFactory)

This method allows you to retrieve the class factory. Typically, you would call **CreateInstance**. However, if you are going to create a large number of the same component, it is more efficient to retrieve the factory, and then ask it to create the instances for you.

HRESULT CreateInstance(LPCTSTR type, IUnknown **ppInstance)

This method creates a new instance of a component, given its type. Use the **CreateInstance** template method instead, which allows type-safe object creation.

HRESULT SetContainer(IWizardPageContainer *pContainer)

This method is for internal use only.

HRESULT RegisterService(REFGUID iid, IUnknown *pService)

Services are single instances of a component that can be used in multiple places. You can use this method to register a service on one page, and then retrieve that same

instance from another page.

HRESULT GetService(REFGUID iid, IUnknown **ppService)

This method retrieves a service that was previously registered with a call to RegisterService.

HRESULT SetLanguage(LANGID languageId)

This method sets the language of the UDI Wizard to the language identifier you provided in the **languageId** parameter.

LANGID GetLanguage()

This method returns the value of the language identifier you provided with the **/locale** command-line parameter for the UDI Wizard. The method returns one of the following values:

- Value of the language identifier provided with the **/locale** command-line parameter
- 0, if you did not provide the **/locale** command-line parameter

ILogger Interface

C++

```
__interface ILogger : IUnknown
{
    HRESULT Init(LPCWSTR logFilename);
    HRESULT MoveLog(LPCWSTR logFilename);
    HRESULT LogBase(EMessageType messageType, LPCTSTR component, SYSTEMTIME
eventTime, LPCTSTR message);
    HRESULT Log(EMessageType messageType, LPCTSTR component, LPCTSTR
message);
    HRESULT Error(HRESULT error, LPCTSTR component, LPCTSTR message);
    HRESULT Error2(HRESULT error, LPCTSTR component, LPCTSTR message,
LPCTSTR message2);
    HRESULT Normal(LPCTSTR component, LPCTSTR message);
    HRESULT Normal2(LPCTSTR component, LPCTSTR message, LPCTSTR message2);
    HRESULT Verbose(LPCTSTR component, LPCTSTR message);
    HRESULT Verbose2(LPCTSTR component, LPCTSTR message, LPCTSTR message2);
    HRESULT Debug(LPCWSTR component, LPCWSTR message);
    HRESULT EnableDebug(BOOL debug);
    HRESULT Close(void);
```

```
    HRESULT GetLogFilename(LPBSTR pFilename);
};
```

Overview

The UDI Wizard logs information to a log file, which helps troubleshoot issues found in the field. It is a good idea for your pages to log information. You can obtain a pointer to this interface from within your page using the page's **Logger()** method. Lines in the log file contain a "level" number that represents error, normal, verbose, or debug messages.

> ⓘ **Note**
>
> Debug messages are not saved to the log file unless debug support is turned on. You can turn on debug support by adding the following line to the **Style** element in the .config file:

XML

```
<Setter Property="debug">true</Setter>
```

Init

C++

```
HRESULT Init(LPCWSTR logFilename)
```

This method is for internal use only.

MoveLog

C++

```
HRESULT MoveLog(LPCWSTR logFilename)
```

This method is for internal use only.

LogBase

C++

```
HRESULT LogBase(EMessageType messageType, LPCTSTR component, SYSTEMTIME
eventTime, LPCTSTR message)
```

This method is for internal use only.

Log

```
C++

HRESULT Log(EMessageType messageType, LPCTSTR component, LPCTSTR message)
```

This method is for internal use only.

Error

```
C++

HRESULT Error(HRESULT error, LPCTSTR component, LPCTSTR message)
```

Call this method to log information about an error. See Table 46.

Table 46. HRESULT Error

Expand table

Parameter	Description
Error	The error code returned by a call (This code will be displayed in the log entry as a number.)
Component	A string that identifies the source of the error, which is generally your page or the component that you have written
Message	The message that explains what caused the error

Error2

```
C++

HRESULT Error2(HRESULT error, LPCTSTR component, LPCTSTR message, LPCTSTR
message2)
```

This method is like the **Error** method but allows you to provide a two-part message. The final message will have "message," and then "message2" in the output file. This is simply a convenience method.

Normal

```
C++

HRESULT Normal(LPCTSTR component, LPCTSTR message)
```

This method logs a normal message. See the description of the Error method for parameters.

Normal2

```
C++

HRESULT Normal2(LPCTSTR component, LPCTSTR message, LPCTSTR message2)
```

This method logs a normal message. See the description of the Error2 method for parameters.

Verbose

```
C++

HRESULT Verbose(LPCTSTR component, LPCTSTR message)
```

This method logs a verbose message. See the description of the Error method for parameters.

Verbose2

```
C++

HRESULT Verbose2(LPCTSTR component, LPCTSTR message, LPCTSTR message2)
```

This method logs a verbose message. See the description of the Error2 method for parameters.

Debug

```
C++

HRESULT Debug(LPCWSTR component, LPCWSTR message)
```

This method logs a debug message. See the description of the Error method for parameters. Debug messages are not saved to the file unless enabled. See the Overview section for details.

EnableDebug

```
C++

HRESULT EnableDebug(BOOL debug)
```

This method is for internal use only.

Close

```
C++

HRESULT Close(void)
```

This method is for internal use only.

GetLogFilename

```
C++

HRESULT GetLogFilename(LPBSTR pFilename)
```

This method retrieves the name of the log file.

IOrientation Interface

```
C++

__interface IOrientation : IUnknown
{
    void SetController(IWizardDialogController *pController);
    int AddPage(LPCTSTR name);
```

```
    void SelectPage(int index);
};
```

This interface is for internal use only.

ISettings Interface

```
C++

__interface ISettings : IUnknown
{
    int NumDlls();
    int NumPages();

    HRESULT SetStage(LPCWSTR stageName);
    HRESULT GetDllName(long index, __out LPBSTR pDllName);
    HRESULT GetPageInfo(long index, __out ISettingsProperties **ppPageInfo);
    HRESULT GetStyle(__out ISettingsProperties **ppStyleInfo);
};
```

This interface is for internal use only.

ISettingsProperties Interface

```
C++

__interface ISettingsProperties : IUnknown
{
    HRESULT GetAttribute(LPCTSTR attributeName, __out LPBSTR
attributeValue);
    IStringProperties * Properties();
    HRESULT SelectNodes(LPCTSTR xPath, __out IXMLDOMNodeList **ppList);
    HRESULT SelectSingleNode(LPCTSTR xPath, __out IXMLDOMNode **ppNode);
    HRESULT GetDataNode(LPCTSTR name, __out ISettingsProperties **ppNode);
    HRESULT GetDataNodes(__out IDataNodes **ppNodes);
    HRESULT GetChildDataNodes(LPCTSTR childeName, __out IDataNodes
**ppNodes);
};
```

Overview

This interface provides access to page data. To get to the top level of page data, use the page's **Settings()** method.

HRESULT GetAttribute(LPCTSTR attributeName, LPBSTR attributeValue)

This method allows you to retrieve the values of attributes on the main node, which is the **Page** node when you are using the **Settings()** method of the page.

IStringProperties * Properties()

This method provides access to the setter property values under the main node. For a page, these are the top-level properties.

HRESULT SelectNodes(LPCTSTR xPath, IXMLDOMNodeList **ppList)

Call this method if you want to directly get a list of XML nodes using an XPath expression. It is better to use one of the other methods if you can. Use this method only if you cannot get to nodes any other way.

HRESULT SelectSingleNode(LPCTSTR xPath, IXMLDOMNode **ppNode)

Call this method if you want to directly get a single XML node using an XPath expression. It is better to use one of the other methods if you can. Use this method only if you can't get to a node any other way.

HRESULT GetDataNode(LPCTSTR name, ISettingsProperties **ppNode)

Retrieve a **Data** element based on that element's **Name** attribute.

HRESULT GetDataNodes(IDataNodes **ppNodes)

This method retrieves a list of **DataItem** elements under the current node. From the page level, call **GetDataNode** to retrieve an **ISettingsProperty** interface for the data. Then, on that instance, call **GetDataNodes** to retrieve the list of records. For example, given this XML:

XML

```
    <Page ...>
      <Data Name="Network">
        <DataItem>
          <Setter Property="DisplayName">Public</Setter>
```

```
          <Setter Property="Share">\\servername\Share</Setter>
        </DataItem>
        <DataItem>
          <Setter Property="DisplayName">Dev Team</Setter>
          <Setter Property="Share">\\servername\DevShare</Setter>
        </DataItem>
      </Data>
```

C++

```
PSettingsProperties pData;
Settings()->GetDataNode(L"Network", &pData);
PDataNodes pNodes;
pData->GetDataNodes(&pNodes);
```

HRESULT GetChildDataNodes(LPCTSTR childeName, IDataNodes **ppNodes)

This method provides a quick way to get to the set of **DataItem** nodes under a specific **Data** node. Using the XML from the **GetDataNodes** example, the following code does exactly the same thing as the four lines of code in the example under **GetDataNodes** but with error checking:

C++

```
ISimpleStringProperties Interface
```

ISimpleStringProperties Interface

C++

```
__interface ISimpleStringProperties : IStringProperties
{
void Add(LPCTSTR propertyName, LPCTSTR value);
};
```

By itself, this interface may not be useful. However, it is implemented by the **ID_SimpleStringProperties** component, which also implements the **IStringProperties** interface. You can use this component in cases where you need to pass a set of properties to another component, such as a task, but you want to add values programmatically instead of using values from XML. Here is an example of how you would use this interface:

C++

```
PSimpleStringProperties *pProperties;
CreateInstance(Container(), ID_SimpleStringProperties, &pProperties);
pProperties->Add(L"filename", L"%windir%\\system32\\cscript.exe");
pTask->Init(pProperties, nullptr);
IStringProperties
__interface IStringProperties : IUnknown
{
    HRESULT Get(LPCTSTR propertyName, [out] LPBSTR pPropValue);
};
```

This interface provides simple access to a set of setter elements that come from XML. This interface is available for the properties of a page using **Settings()->Properties()**.

HRESULT Get(LPCTSTR propertyName, [out] LPBSTR pPropValue)

This method retrieves a single property value. See Table 47 and Table 48.

Table 47. IHRESULT Get Property Value

Expand table

Parameter	Description
propertyName	Name of the property that you want to read
pPropValue	On exit, contains the property value as a string (This value will be **nullptr** if there is no such property.)

Table 48. IHRESULT Get Property Value Results

Expand table

HRESULT	Description
S_OK	Property value is retrieved.
E_INVALIDARG	There is no property with the name you provided.

ITaskManager Interface

C++

```
__interface ITaskManager : IUnknown
{
    HRESULT Init(IWizardPageView *pPageView, int idListView, int idMessage,
int idRetryButton, ISettingsProperties *pPageInfo, ITaskManagerCallback
*pCallback);
    HRESULT SetFailMessage(LPCWSTR message);

    HRESULT Start(void);

    HRESULT GetTaskMessage(size_t index, LPBSTR message);
    HRESULT GetResultType)(size_t index, LPBSTR type);
    HRESULT GetProperty(size_t index, LPCTSTR propertyName, LPBSTR value);
    int GetSelectedIndex(void);
    HRESULT Wait(DWORD waitMilliseconds);
    size_t FailedCount(void);
    size_t WarningCount(void);
    size_t SucceedCount(void);
    size_t RunningCount(void);

    void OnCommonControlEvent(WORD controlId, LPNMHDR pInfo);
    void OnControlEvent(WORD eventId, WORD controlId);
    void EnableButtons(BOOL enable);
}
```

This interface is implemented by the **TaskManager** component (**ID_TaskManager** in ITaskManager.h), which is the component that runs tasks on the preflight page. You can either use the preflight page directly, which is what you do most of the time, or build your own page, letting this component do most of the work.

HRESULT Init(IWizardPageView *pPageView, int idListView, int idMessage, int idRetryButton, ISettingsProperties *pPageInfo, ITaskManagerCallback *pCallback)

You must call this method before calling any other method. It initializes the **TaskManager** component. See Table 49.

Table 49. HRESULT Init

Expand table

Parameter	Description
pPageView	Provides access to the page that will be running tasks (This page must have a specific set of controls, which are outlined in the next few parameters.)
idListView	The control ID of a **ListView** control that will display the list of tasks and the status of those tasks

Parameter	Description
idMessage	The control ID of a text box that will be used to display a message for the task that you select
idRetryButton	The control ID of a button you can select to run the tasks again
pPageInfo	A wrapper around the page's XML (**TaskManager** loads the set of tasks to run from this XML.)
pCallback	Can be null (If this parameter is not null, **TaskManager** calls the **Started** method when it starts a task and the **Finished** method for each task that finishes running.)

HRESULT SetFailMessage(LPCWSTR message)

This method sets the message that will be displayed if one or more tasks fail.

HRESULT Start(void)

This method starts all the tasks. Each task is started on a separate thread.

HRESULT GetTaskMessage(size_t index, LPBSTR message)

This method is for internal use only. It retrieves the current message for a task based on its index in the list of tasks.

HRESULT GetResultType)(size_t index, LPBSTR type)

This method retrieves the current "type" for a task. Table 50 shows the available types.

Table 50. HRESULT GetResultType

Expand table

Type	Description
0	Represents a task that succeeded
1	Represents a tasks that returned a warning
-1	Represents a failed task

The type is retrieved by looking at the task's exit or error code and finding a match in the task's <ExitCodes> XML element.

HRESULT GetProperty(size_t index, LPCTSTR propertyName, LPBSTR value)

This method is used by the progress and preflight pages to retrieve the **BitmapFilename** setter property so it can display an image next to the message for the task that you highlight. In other words, you can add a custom setter to the task's XML, and then retrieve it with this method.

int GetSelectedIndex(void)

This method retrieves the index of the currently selected task, which is useful if you want to retrieve additional information about the task (see **GetProperty** method) to display for the selected task. The progress and preflight pages use this method to display an image for the selected task.

HRESULT Wait(DWORD waitMilliseconds)

This method mainly helps with unit tests so the test can ensure that tasks finish before the unit test exits. You would not normally call this method. It returns either when all tasks finish running or the wait time has elapsed.

size_t FailedCount(void)

This method returns the number of tasks currently marked as failed.

size_t WarningCount(void)

This method returns the number of tasks currently marked as warning.

size_t SucceedCount(void)

This method returns the number of tasks currently marked as succeeded.

size_t RunningCount(void)

This method returns the number of tasks currently running.

void OnCommonControlEvent(WORD controlId, LPNMHDR pInfo)

Call this method from your page's **OnCommonControlEvent** so the **TaskManager** can process events it needs.

void OnControlEvent(WORD eventId, WORD controlId)

Call this method from your page's **OnControlEvent** so the **TaskManager** can process events it needs.

void EnableButtons(BOOL enable)

This method is for internal use only.

IWizardComponent Interface

```
C++

__interface IWizardComponent : IUnknown
{
    HRESULT SetContainer(IWizardPageContainer *pContainer);
};
```

Overview

Typically, you will not implement this interface directly but instead through the **WizardComponent** template class. If your component implements this interface and you have registered a class factory with the registry, your component receives a pointer to the **IWizardPageContainer** instance when it is created. This helps you, for example, access the Logger or the registry for creating other components that your component might need.

IWizardDialogController Interface

```
C++

__interface IWizardDialogController : IUnknown
{
    void Initialize(ISettings *pSettings);
    void InitPages(void);
    void Start();
    void Next();
    void Finish();
```

```
    void Previous();
    int NumPages();
    void Cancel();

    HRESULT Focus(WizardButtons button);
    HRESULT SetEnable(WizardButtons button, BOOL enable);
    void ShowWarningMessage(LPCTSTR message);
    void HideWarningMessage();

    void ChangePage(size_t newIndex);
    IUnknown *CurrentPage(void);
    HRESULT GetCurrentTitle([out, retval] LPBSTR pDisplayName);
};
```

This interface is for internal use only.

IWizardDialogView Interface

C++

```
__interface IWizardDialogView : IUnknown
{
    HRESULT LoadBannerImage(LPCTSTR bannerFilename);
    HRESULT LoadPage(LPCTSTR pageType, ISettingsProperties *pPageSettings,
IWizardPageView **view);
    HRESULT SetEnable(WizardButtons button, BOOL enable);
    HRESULT Focus(WizardButtons button);
    void EnableFinish(BOOL isFinish);
    void Exit(int exitCode);
    void ShowWarningMessage(LPCTSTR message);
    void HideWarningMessage(void);
    void SetTitle(LPCTSTR title);
    void SetPageTitle(LPCTSTR title);
    int ShowMessageBox(LPCTSTR message, LPCTSTR lpCaption, UINT uType);
    HWND GetHwnd(void);
    void UpdateFocus(void);
};
```

This interface is for internal use only.

IWizardPage Interface

C++

```
__interface IWizardPage : IUnknown
{
    HRESULT SetPageSettings(ISettingsProperties *pPageSettings);
    HINSTANCE GetInstanceHandle(void);
    int GetDialogResourceId(void);
```

```
    void WindowCreated(IWizardPageView *pView, IWizardPageContainer
*pContainer);
    void WindowShown(void);
    void WindowHidden(void);

    HRESULT NextSelected(void);
    void ControlEvent(WORD eventId, WORD controlId);
    void CommonControlEvent(WORD controlId, LPNMHDR pInfo, LPBOOL pCancel);
    void UnhandledEvent(HWND hwnd, UINT message, WPARAM wParam, LPARAM
lParam);
};
```

Overview

This interface is implemented by **WizardPageImpl**, so you will not typically have to implement this it yourself. The wizard calls all of these methods for you when it interacts with your custom pages.

IWizardPageContainer Interface

C++

```
__interface IWizardPageContainer : IUnknown
{
    ILogger * Logger(void);
    IPropertyBag * Properties(void);
    HRESULT CreateInstance(LPCTSTR type, [out] IUnknown **ppInstance);
    HRESULT GetService(REFIID iid, [out] IUnknown **ppInstance);
    HRESULT ReplaceVariables(LPCTSTR source, [out] LPBSTR pDest);
    HRESULT GotoPage(LPCTSTR pageName);
    int ShowMessageBox(LPCTSTR message, LPCTSTR lpCaption, UINT uType);
    BOOL InPreview(void);
    HWND GetHwnd(void);
};
```

Overview

This interface is available to your page via the **Container** method (implemented by **WizardPageImpl**) and gives you access to various services of the wizard.

ILogger * Logger(void)

Use this method to write messages to the log file—for example:

C++

```
Logger()->Verbose(s_component, L"Message for log file");
```

IPropertyBag * Properties(void)

This method provides access to "memory" variables, which are properties that are in memory only while the UDI Wizard is running. These properties are available to other pages either in code or in the XML using the **$memoryVarName$** syntax.

HRESULT CreateInstance(LPCTSTR type, [out] IUnknown **ppInstance)

This method allows you to create a new instance of any component that has been registered. However, it is better to use the template function **CreateInstance**, because it is strongly typed.

HRESULT GetService(REFIID iid, [out] IUnknown **ppInstance)

This method allows you to retrieve a service that has been registered. However, it is better to call the **GetService** template function, which is strongly typed (instead of using **IUnknown**).

HRESULT ReplaceVariables(LPCTSTR source, [out] LPBSTR pDest)

This method handles working with variables inside string values. It supports the formats shown in Table 51 and Table 52.

Table 51. HRESULT ReplaceVariables

Expand table

Format	Description
$Name$	Replaces the value of a memory variable with this name (If there is no memory variable with the name, the "token" will be removed.)
%Name%	Either a task sequence variable or an environment variable. The order is as follows: 1. Use the value of a task sequence variable, if present. 2. Use the value of an environment variable, if present. 3. Otherwise, remove this text from the string.

Table 52. HRESULT Parameter

Expand table

Parameter	Description
Source	The input string, which can contain any combination of $ and % variables or none at all
pDest	On return, contains a new string that has all the tokens replaced according to Table 51

HRESULT GotoPage(LPCTSTR pageName)

This method has not been fully tested. The idea is that you can switch directly to a specific page based on the name of the page as defined in the .config XML file. Calling this method bypasses the **OnNextSelected** on your page. In addition, the behavior of this method is subject to change, so use it at your own risk.

int ShowMessageBox(LPCTSTR message, LPCTSTR lpCaption, UINT uType)

This method displays a message box with the text and caption that you provide. The **uType** parameter is any value that you can supply to the **MessageBox** Win32 function.

BOOL InPreview(void)

This method returns TRUE if you launched the wizard in "preview" mode by supplying the **/preview** switch. In preview mode, the **Next** button is never disabled. This method allows you to bypass code in preview mode, for example, that could cause issues when you do not have valid data on the page.

HWND GetHwnd(void)

This method returns the **HWND** for the main dialog box. Use this method with care. Generally, the UDI Wizard application programming interface is designed so that you never work directly with window handles.

IWizardPageView Interface

```
C++
```

```
__interface IWizardPageView : IUnknown
{
    HRESULT GetControlWrapper(int itemId, DialogControlTypes controlType,
IUnknown **ppControl);
    HWND GetHwnd(void);
    HWND GetControl(int itemId);
    HRESULT Show (void);
    HRESULT Hide(void);
    HRESULT Focus(int itemId);
    IWizardPage * Page(void);
    IFormController * Form(void);

    HRESULT FocusWizardButton(WizardButtons button);
    HRESULT SetEnable(WizardButtons button, BOOL enable);
    void ShowWarningMessage(LPCTSTR message);
    void HideWarningMessage(void);
};
```

This interface is available to the code in your page through the **View** method (implemented by **WizardPageImpl**).

HRESULT GetControlWrapper(int itemId, DialogControlTypes controlType, IUnknown *ppControl)

The UDI Wizard uses *wrappers*, which are really façades for interacting with the controls on your page. Using these façades instead of the actual controls makes it much easier to write tests for your page, because you can provide mock façades from your tests.

Instead of using this method directly, it is better to use the **GetControlWrapper** template method, which is strongly typed—for example:

C++

```
PComboBox m_pLanguagePackCombo;
GetControlWrapper(View(), IDC_MY_COMBO, CONTROL_COMBO_BOX, &m_pCombo);
```

HWND GetHwnd(void)

This method returns the window handle for your page. Generally, you should not need access to this window handle.

HWND GetControl(int itemId)

If you must, you can call this method to get the window handle for a control on your page. (It is better to call the **GetControlWrapper** template function).

HRESULT Show (void)

This method is for internal use only.

HRESULT Hide(void)

This method is for internal use only.

HRESULT Focus(int itemId)

Set the input focus to a specific control.

IWizardPage * Page(void)

This method is for internal use only.

IFormController * Form(void)

This method is for internal use only.

HRESULT FocusWizardButton(WizardButtons button)

Sets the focus to one of the wizard's buttons.**WizardButtons** has two values: **BackButton** and **NextButton**.

HRESULT SetEnable(WizardButtons button, BOOL enable)

Request that one of the wizard buttons be enabled or disabled. The button might not match the state that you request. For example, if you run the UDI Wizard with the **/preview** switch, the buttons will always be enabled. **WizardButtons** has two values: **BackButton** and **NextButton**.

void ShowWarningMessage(LPCTSTR message)

This method displays a warning message at the bottom of the page content area. This message can be any text you want.

void HideWarningMessage(void)

Hide a warning message that you displayed with a call to **ShowWarningMessage**.

IXmlDocument Interface

```
C++
```

```
__interface IXmlDocument : IUnknown
    HRESULT Load(LPCTSTR filename);
    HRESULT LoadXml(LPCTSTR xml);
    HRESULT Save(LPCWSTR filename);
    HRESULT GetParseErrorMessage(LPBSTR pMessage);
    HRESULT SelectNodes(LPCTSTR xpath, IXMLDOMNodeList **ppNodes);
    HRESULT SelectSingleNode(LPCTSTR xpath, IXMLDOMNode **ppNode);
    HRESULT AddSchema(LPCTSTR filename, LPCTSTR ns);
    HRESULT AddAttribute(IXMLDOMNode *pNode, LPCWSTR name, LPCWSTR value);
    HRESULT CreateNode(DOMNodeType type, LPCWSTR name, LPCWSTR ns,
IXMLDOMNode **ppNode);
};
```

Overview

This interface is implemented by the **ID_IXmlDocument** component, which is a façade designed to make it easier to work with XML documents in C++.

HRESULT Load(LPCTSTR filename)

This method loads an XML document from an external file. It returns **S_OK** if the file was loaded without errors or **S_FALSE** if an error occurred. When there is an error, you can get the error message by calling **GetParseErrorMessage**.

HRESULT LoadXml(LPCTSTR xml)

This method loads an XML document from a string instead of an external file. Other than the source for reading the XML, the behavior is the same as the **Load** method.

HRESULT Save(LPCWSTR filename)

This method saves the XML document that is in memory to an external file.

HRESULT GetParseErrorMessage(LPBSTR pMessage)

This method returns a new string with the error message from loading the XML document, if any. It always returns **S_OK**.

HRESULT SelectNodes(LPCTSTR xpath, IXMLDOMNodeList **ppNodes)

This method allows you to use an XPath expression to retrieve a collection of nodes from the document. It always returns **S_OK**.

HRESULT SelectSingleNode(LPCTSTR xpath, IXMLDOMNode **ppNode)

This method allows you to use an XPath expression to retrieve one node from the document. It always returns **S_OK**.

HRESULT AddSchema(LPCTSTR filename, LPCTSTR ns)

This method adds the name of an external schema file that will be used to validate the schema of your XML document when it is loaded. The namespace you provide is the string you can use in XPath queries, although this has not been tested.

HRESULT AddAttribute(IXMLDOMNode *pNode, LPCWSTR name, LPCWSTR value)

This method adds a new attribute to an existing node in the XML document. See Table 53.

Table 53. HRESULT AddAttribute

Expand table

Parameter	Description
pNode	The node to which you want to add an attribute
Name	Name of the new attribute
Value	The value for the new attribute

HRESULT CreateNode(DOMNodeType type, LPCWSTR name, LPCWSTR ns, IXMLDOMNode **ppNode)

Call this method to create a new node:

```
C++
```

```
Pointer<IXMLDOMNode> pNewChild
pXmlDom->CreateNode(NODE_ELEMENT, L"MyElement", L"", &pNewChild);
```

Once you create a new node, you can add it as a child to another node by calling the parent's **appendChild** method.

Helper Functions

CreateInstance Template Function

```
C++

HRESULT CreateInstance(IWizardPageContainer *pContainer, LPCTSTR type, I
**ppObject)
```

This function is defined in IWizardPageContainer.h and provides a type-safe wrapper over the **IWizardPageContainer->CreateInstance** method—for example:

```
C++

CreateInstance<IDirectory>(Container(), ID_Directory, &pDirectory);
```

This code creates a new **ID_Directory** component to retrieve the **IDirectory** interface of that component.

GetService Template Function

```
C++

void GetService(IWizardPageContainer *pContainer, I **ppService)
```

This function is defined in IWizardPageContainer.h and provides a type-safe wrapper over the **IWizardPageContainer->GetService** method—for example:

```
C++

GetService<ITSVariableBag>(Container(), &pTsBag);
```

This function retrieves the task sequence component, which supports the **ITSVariableBag** interface. (For **ITSVariableBag**, you can use the TSVariables method of the **WizardPageImpl** class, instead.)

UDI Wizard Designer Configuration File Schema Reference

This file is consumed by the UDI Wizard Designer. A separate file is created for each custom .dll file, which can contain custom wizard page editors, custom tasks, or custom validators. The file must end with *.config* and reside in the *installation_folder*\Bin\Config folder (where *installation_folder* is the folder in which you installed MDT).

Table 54 lists the elements in the UDI Wizard Designer configuration file and their descriptions. The **DesignerConfig** element is the root node for this reference.

Table 54. Elements in the UDI Wizard Designer Configuration File and Their Descriptions

Expand table

Element Name	Description
DesignerConfig	Specifies the root for all other elements
DesignerMappings	Groups a set of Pageelements
Page	Specifies a wizard page editor to be loaded in the UDI Wizard Designer, which is used to edit the configuration settings for a wizard page
Param	Specifies a parameter that is passed to the parent Task or Validator element and corresponds to a Setter element in the UDI Wizard configuration file **Note:** The attributes for this element are different if the parent is the Task or Validator element.
Task	Specifies a task within the task library
TaskItem	Specifies a group of parameters that are passed to the task
TaskLibrary	Groups a set of Task elements
Validator	Specifies a validator within the validator library
ValidatorLibrary	Groups a set of Validator elements

DesignerConfig

This element specifies the root for all other elements.

Element Information

Table 55 provides information about the DesignerConfig element.

Table 55. DesignerConfig Element Information

Expand table

Attribute	Value
Number of occurrences	One: This element is required.
Parent elements	None
Contents	**DesignerMappings**, **TaskLibrary**, **ValidatorLibrary**

Element Attributes

This element has no attributes.

Remarks

None.

Example

```
C++

<DesignerConfig>
   + <TaskLibrary>
   + <ValidatorLibrary>
   + <DesignerMappings>
</DesignerConfig>
```

DesignerMappings

This element groups a set of Page elements.

Element Information

Table 56 provides information about the DesignerMappings element.

Table 56. DesignerMappings Element Information

Expand table

Attribute	Value
Number of occurrences	Zero or one within the DesignerConfig element (This element is optional if there are no custom wizard page in the DLL that corresponds to this UDI Wizard Designer configuration file.)
Parent elements	**DesignerConfig**
Contents	**Page**

Element Attributes

This element has no attributes.

Remarks

None.

Example

C++

```
<DesignerConfig>
   + <TaskLibrary>
   + <ValidatorLibrary>
   - <DesignerMappings>
        <Page DLL="SharedPages.dll"
           Description="Used to display text that describes the current
stagegroup"
           Type="Microsoft.SharedPages.WelcomePage"
           DisplayName="Welcome"
           Image="Welcome_188.png"

DesignerType="Microsoft.Enterprise.UDIDesigner.CoreModules.Views.WelcomePage
View"

DesignerAssembly="Microsoft.Enterprise.UDIDesigner.CoreModules.dll"/>
        <Page DLL="OSDRefreshWizard.dll"
           Description="Captures or restores user state data"
           Type="Microsoft.OSDRefresh.UserStatePage"
           DisplayName="User Data"
           Image="UserState_188.png"

DesignerType="Microsoft.Enterprise.UDIDesigner.CoreModules.Views.UserStatePa
geView"

DesignerAssembly="Microsoft.Enterprise.UDIDesigner.CoreModules.dll"/>
```

```
        <Page DLL="OSDRefreshWizard.dll"
           Description="Allows selecting the image to install, target drive,
and whether to format"
           Type="Microsoft.OSDRefresh.VolumePage"
           DisplayName="Volume"
           Image="Volume_188.png"

DesignerType="Microsoft.Enterprise.UDIDesigner.CoreModules.Views.VolumePageV
iew"

DesignerAssembly="Microsoft.Enterprise.UDIDesigner.CoreModules.dll"/>
     </DesignerMappings>
</DesignerConfig>
```

Page

This element specifies a wizard page editor to be loaded in the UDI Wizard Designer, which is in turn used to edit the configuration settings for a wizard page.

Element Information

Table 57 provides information about the Page element.

Table 57. Page Element Information

Expand table

Attribute	Value
Number of occurrences	One or more for each wizard page defined in the DesignerMappings element
Parent elements	**DesignerMappings**
Contents	Any well-formed XML content

Element Attributes

Table 58 lists the attributes of the Page element and a description for each.

Table 58. Attributes and Corresponding Values for the Page Element

Expand table

Attribute	Description
Description	Specifies text that provides information about the parameter, which is displayed in the UDI Wizard Designer
DesignerAssembly	Specifies the name of the .dll file associated with the wizard page editor (The .dll file must exist in the *installation_folder*\Bin folder (where *installation_folder* is the folder in which you installed MDT.)
DesignerType	Specifies the name of the wizard page editor within the .dll file specified in the **DesignerAssembly** attribute (This is the Microsoft .NET type for the wizard page editor, with the fully qualified Microsoft .NET namespace.)
DisplayName	Specifies the user-friendly name of the page editor, which is displayed in the UDI Wizard Designer
DLL	Specifies the name of the .dll file associated with the wizard page (The .dll file must exist in the *installation_folder*\Templates\Distribution\Tools*platform* folder (where *installation_folder* is the folder in which you installed MDT and *platform* is **x86** for the 32-bit version or **x64** is for the 64-bit version.) **Note:** Ensure that the DLL processor architecture matches the MDT processor architecture installed. For example, if you installed a 32-bit version of MDT, then ensure you use a 32-bit DLL for the wizard page.
Image	Specifies the name of an image of the page that is in Portable Network Graphics (PNG) format (The .png file must exist in the *installation_folder*\Bin\Images folder (where *installation_folder* is the folder in which you installed MDT.)
Type	Specifies the wizard page editor and must match the named used when the custom page was registered

Remarks

The UDI Wizard Designer uses the Page element like a template to create the initial XML for a new wizard. The UDI Wizard Designer performs schema validation to ensure that the Page and child elements have a valid format. This element provides a mapping between the UDI Wizard page type and the information that the UDI Wizard Designer needs to edit and create pages of this type using a custom page editor.

Example

None.

Param

This element specifies a parameter that is passed to the parent Task or Validator element and corresponds to a Setter element in the UDI Wizard configuration file.

> **Note**
>
> The attributes for this element are different if the parent is the **Task** or **Validator** element.

Element Information

Table 59 provides information about the Param element.

Table 59. Param Element Information

Expand table

Attribute	Value
Number of occurrences	One or more for each TaskItem or Validator parent element
Parent elements	**TaskItem**, **Validator**
Contents	Any well-formed XML content

Element Attributes

Table 60 lists the attributes of the Param element and provides a description of each.

Table 60. Attributes and Corresponding Values for the Param Element

Expand table

Attribute	Description
Description	Specifies text that provides information about the parameter, which is displayed in the UDI Wizard Designer **Note:** This attribute is valid only for the Validator element.
DisplayName	Specifies the user-friendly name of the validator parameter, which is displayed for the appropriate UDI Wizard page in the UDI Wizard Designer (This name is

Attribute	Description
	usually more descriptive than the **Name** attribute.) **Note:** This attribute is valid only for the Validator element.
Name	Specifies the name of the parameter that is passed to the task or validator, depending on the parent element (This attribute will become the **Property** attribute in a Setter element in the UDI Wizard configuration file.) **Note:** This parameter is used for both TaskItem and Validator parent elements.

Remarks

None.

Example

None.

Task

This element specifies a task within the task library.

Element Information

Table 61 provides information about the Task element.

Table 61. Task Element Information

Expand table

Attribute	Value
Number of occurrences	One or more within the TaskLibrary element (This element is not optional if the **TaskLibrary** element is specified.)
Parent elements	**TaskLibrary**
Contents	**TaskItem**

Element Attributes

Table 62 lists the attributes of the Task element and provides a description of each.

Table 62. Attributes and Corresponding Values for the Task Element

Expand table

Attribute	Description
Description	Specifies text that provides information about the task, which is displayed in the UDI Wizard Designer
DLL	Specifies the name of the .dll file associated with the task (The .dll file must exist in the *installation_folder*\Templates\Distribution\Tools*platform* folder (where *installation_folder* is the folder in which you installed MDT and *platform* is **x86** for the 32-bit version or **x64** for the 64-bit version.)
Name	Specifies the name of the task, which is displayed in the appropriate UDI Wizard page and in the UDI Wizard Designer
Type	Specifies the task type, which is registered with the factory registry and used to call a specific task within a .dll file

Remarks

None.

Example

None.

TaskItem

This element specifies a group of parameters that are passed to the task.

Element Information

Table 63 provides information about the TaskItem element.

Table 63. TaskItem Element Information

Expand table

Attribute	Value
Number of occurrences	One or more for each **Task** element
Parent elements	**Task**
Contents	**Param**

Element Attributes

Table 64 lists the attributes of the TaskItem element and provides a description of each.

Table 64. Attribute and Corresponding Values for the TaskItem Element

Expand table

Attribute	Description
Type	Specifies the of element type that will be created in the UDI Wizard configuration file. An XML element will be created that corresponds to the value of this attribute. For example, if the value for this attribute is File, then a **File** element will be created in the UDI Wizard configuration file. Currently, the only values supported are: - **File**, which requires two Param child elements (one **Param** child element with the **Name** attribute set to **Source** and another **Param** child element with the **Name** attribute set to **Dest**) - Setter, which requires one **Param** child element

Remarks

None.

Example

None.

TaskLibrary

This element groups a set of Task elements.

Element Information

Table 65 provides information about the TaskLibrary element.

Table 65. TaskLibrary Element Information

Expand table

Attribute	Value
Number of occurrences	Zero or one within the DesignerConfig element (This element is optional if there are no custom tasks in the DLL that correspond to this UDI Wizard Designer configuration file.)
Parent elements	**DesignerConfig**
Contents	**Task**

Element Attributes

This element has no attributes.

Remarks

None.

Example

C++

```
<DesignerConfig>
   - <TaskLibrary>
        +<Task DLL="" Description="Executes a process with the given command
line." Type="Microsoft.Wizard.ShellExecuteTask" Name="Shell Execute Task">
        +<Task DLL="OSDRefreshWizard.dll" Description="Discovers supported
applications for install." Type="Microsoft.OSDRefresh.AppDiscoveryTask"
Name="Application Discovery">
        +<Task DLL="SharedPages.dll" Description="Check to ensure a wired
network connection is available."
Type="Microsoft.SharedPages.WiredNetworkTask" Name="Wired Network Check">
        +<Task DLL="OSDRefreshWizard.dll" Description="Check to ensure power
source is AC (not battery)." Type="Microsoft.OSDRefresh.ACPowerTask"
Name="AC Power Check">
        +<Task DLL="" Description="Check to ensure power source is AC (not
battery)." Type="Microsoft.Wizard.CopyFilesTask" Name="Copy Files Task">
     </TaskLibrary>
   + <ValidatorLibrary>
```

```
    + <DesignerMappings>
</DesignerConfig>
```

Validator

This element specifies a validator within the validator library.

Element Information

Table 66 provides information about the Validator element.

Table 66. Validator Element Information

Expand table

Attribute	Value
Number of occurrences	Zero or more within the ValidatorLibrary element (This element is optional.)
Parent elements	**ValidatorLibrary**
Contents	**Param**

Element Attributes

Table 67 lists the attributes of the Validator element and provides a description of each.

Table 67. Attributes and Corresponding Values for the Validator Element

Expand table

Attribute	Description
Description	Specifies text that provides information about the validator, which is displayed in the UDI Wizard Designer
DisplayName	Specifies the user-friendly name of the validator displayed in the UDI Wizard Designer (This name is usually more descriptive than the **Name** attribute.)
DLL	Specifies the name of the .dll file associated with the validator (The .dll file must exist in the *installation_folder*\Templates\Distribution\Tools*platform* folder (where

Attribute	Description
	installation_folder is the folder in which you installed MDT and *platform* is **x86** for the 32-bit version or **x64** for the 64-bit version.)
Name	Specifies the name of the validator, which is displayed in the appropriate UDI Wizard page and in the UDI Wizard Designer
Type	Specifies the validator type, which is registered with the registry factor and used to call a specific validator within a .dll file

Remarks

None.

Example

None.

ValidatorLibrary

This element groups a set of Validator elements.

Element Information

Table 68 provides information about the ValidatorLibrary element.

Table 68. ValidatorLibrary Element Information

Expand table

Attribute	Value
Number of occurrences	Zero or one within the DesignerConfig element (This element is optional if there are no custom validators in the DLL that correspond to this UDI Wizard Designer configuration file.)
Parent elements	**DesignerConfig**
Contents	**Validator**

Element Attributes

This element has no attributes.

Remarks

None.

Example

<DesignerConfig> + <TaskLibrary> - <ValidatorLibrary> +<Validator DLL="" Description="Requires text in a field" Type="Microsoft.Wizard.Validation.NonEmpty" Name="NonEmpty"> +<Validator DLL="" Description="Doesn't allow certain characters to be in a field" Type="Microsoft.Wizard.Validation.InvalidChars" Name="InvalidChars"> +<Validator DLL="" Description="Must follow a pre-defined pattern" Type="Microsoft.Wizard.Validation.RegEx" Name="NamedPattern"> +<Validator DLL="" Description="Require the contents match a regular expression" Type="Microsoft.Wizard.Validation.RegEx" Name="RegEx"> </ValidatorLibrary> + <DesignerMappings></DesignerConfig>

UDI Wizard Designer Reference

Controls

The controls used to create custom wizard page editors for use in the UDI Wizard Designer are WPF **UserControl** instances. Table 69 lists the controls that you can use to create custom wizard page editors.

Table 69. Controls That Can Be Used to Create Custom Wizard Page Editors

Expand table

Control	Description
CollectionTControl	This control is used to edit data stored in the Data element within a Page element.
FieldElementControl	This control is used to edit a field, which is typically linked to a TextBox control on the .xaml page.
SetterControl	This control is used to modify the value of a setter element in the UDI Wizard configuration file.

CollectionTControl

This control provides many capabilities for editing data. The best way to learn how to use this control is to look at the sample, which shows how to edit data under a page's **Data** element. In particular, the sample shows how to add, remove, and edit items in this control.

FieldElementControl

Use this control to edit a field, which is typically linked to a **TextBox** control on the .xaml page.

Example

The following excerpt from an .xaml file illustrates the use of the **FieldElementControl** to configure the default value for a field on a wizard page using a child **TextBox** control:

XAML

```
<Controls:FieldElementControl
Width="450"
Margin="0,5"
FieldData="{Binding DataContext.Location, ElementName=ControlRoot}"
HeaderText="Location Combo Box"
InstructionText="Here you can configure the behavior of the location combo
box."
HideValidationTab="True">

<TextBox Text="{Binding FieldData.DefaultValue,
 UpdateSourceTrigger=PropertyChanged,
 Mode=TwoWay}"/>
</Controls:FieldElementControl>
```

Properties

FieldData

This string property contains information for connecting the **FieldElementControl** to the underlying XML for the field. The connection is made to a property of the page editor interface. The following excerpt from an .xaml file illustrates the use of the **FieldData** property:

C++

```
FieldData="{Binding DataContext.Location, ElementName=ControlRoot}"
```

In this excerpt, the page editor interface is called **ControlRoot** and is specified in the **ElementName** parameter. The binding is performed to the **DataContext.Location** property of the **ControlRoot** page editor interface. **DataContext** is a view model that points to the Page element within the UDI Wizard configuration file. **Location** is a property of the view that returns a list of the possible locations and is defined by a Data element within the UDI Wizard configuration file. Each location is defined by a DataItem element within the UDI Wizard configuration file.

HeaderText

This string property allows you to specify a header for the FieldElementControl control. The header acts as a title for the control and is formatted as bold, orange text displayed immediately above the control.

InstructionText

This string property allows you to specify informational text for the FieldElementControl control. Typically, the text is used to provide a brief description of the field and explain how configuring the field affects the corresponding wizard page.

HideEnableButton

This Boolean property allows you to control the visibility of the button that changes state between **Unlocked** and **Locked** (enabled or disabled). If set to:

- **True**, the button is not visible
- **False**, the button is visible (This is the default value.)

HideDefaultTab

This Boolean property allows you to control the visibility of the section that contains the control used to set the default value. Although the property refers to a tab, there is no tab on the FieldElementControl but rather a section that can be hidden. If set to:

- **True**, the section is not visible
- **False**, the section is visible (This is the default value.)

HideBorder

This Boolean property allows you to control the visibility of the border around the field control. If set to:

- **True**, the border is not visible
- **False**, the border is visible (This is the default value.)

HideImage

This Boolean property allows you to control the visibility of the image that the **FieldImageSource** property configures. If set to:

- **True**, the image is not visible
- **False**, the image is visible (This is the default value.)

HideValidationTab

This Boolean property allows you to control the visibility of the section where the list of validators is managed. Although the property refers to a tab, there is no tab on the FieldElementControl but rather a section that can be hidden. If set to:

- **True**, the section is not visible
- **False**, the section is visible (This is the default value.)

HideSummaryTab

This Boolean property allows you to control the visibility of the section in which you configure the field summary caption. The caption and corresponding value from the field are displayed on a **SummaryPage** wizard page type in a stage flow. Although the property refers to a tab, there is no tab on the FieldElementControl but rather a section that can be hidden. If set to:

- **True**, the section is not visible
- **False**, the section is visible (This is the default value.)

HideTaskSequenceTab

This Boolean property allows you to control the visibility of the section in which you configure the task sequence variable that corresponds to the field. Although the property refers to a tab, there is no tab on the FieldElementControl but rather a section that can be hidden. If set to:

- **True**, the section is not visible
- **False**, the section is visible (This is the default value.)

SetterControl

Use this control to modify the value of a Setter element in the UDI Wizard configuration file. This control contains a child control used to modify the value of the **setter** element.

Example

The following excerpt from an .xaml file illustrates the use of the **SetterControl** to modify a Setter element named **KeyLocationSetter** using a child **TextBox** control.

XAML

```
<Controls:SetterControl Margin="5"
        Width="450"
        HeaderText="Title text"
        SetterData="{Binding KeyLocationSetter}"
        InstructionText="What this means..."
        HorizontalAlignment="Left">

    <TextBox
                    Margin="0,3"
                    Text="{Binding SetterData.SetterValue, Mode=TwoWay,
UpdateSourceTrigger=PropertyChanged}"
    />

</Controls:SetterControl>
```

Properties

SetterData

You need to bind this to a property of your view or view model that connects to the setter. Doing so is similar to how you would bind to a field, as described for the **FieldElementControl**.

HeaderText

This property allows you to set the text that will appear in the header of the control. Think of this property as a title for the control; by default, it appears as bold, orange text.

InstructionText

Set this property to the text you want to appear below the header—typically instruction text that tells the user of your custom editor when and why they would want to modify the behavior of the field.

Interfaces

Table 70 lists the interfaces that you can use to create custom wizard page editors.

Table 70. Interfaces That Can Be Used to Create Custom Wizard Page Editors

Expand table

Interface	Description
IDataService	Use this interface to connect fields to the **Data** elements in the UDI Wizard configuration file.
IMessageBoxService	This interface provides access to methods that you can use to display message boxes.

IDataService

This interface contains several properties and methods, but there is only one property that you are like to need. That property is the only one documented here.

You can use dependency injection to obtain a pointer to this interface using code like this in your class:

```
C#

[Dependency]
public IDataService DataService { get; set; }
```

Properties

Table 71 lists the properties for the **IDataService** interface.

Table 71. Properties for the IDataService Interface

Expand table

Interface	Description
CurrentPage	This property provides access to the XML elements, attributes, and values beneath the context of the current page being edited in the UDI Wizard configuration file

CurrentPage

```
C#

XElement CurrentPage { get; set; }
```

This property provides access to the XML for the current page. You should never set this property, but you are free to modify the XML for your page. The sample page editor shows examples of modifying the XML. You use this property primarily when you have custom data. For fields and properties (setters), you can use prebuilt controls that take care of all the details.

IMessageBoxService

This interface provides access to methods that you can use to display message boxes. You may be wondering why you need an interface to display a message box. The reality is that you do not: Microsoft uses this interface with in code, because it aids in writing automated tests for designer pages.

However, using these methods does provide one useful benefit: The dialog boxes always have the "owner" set to the UDI Wizard, which ensures that the dialog box is grouped correctly with the main window.

You can use dependency injection to obtain a pointer to this interface using code like this in your class:

```
C#

[Dependency]
public IMessageBoxService MessageBoxes { get; set; }
```

Methods

Table 72 lists the methods for the **IMessageBoxService** interface.

Table 72. Methods for the IMessageBoxService Interface

Expand table

Method	Description
ShowMessageBox	This overloaded method is used to display a message box with the following members: - ShowMessageBox(String message, String caption, MessageBoxImage icon) - ShowMessageBox(string message, string caption, MessageBoxButton button, MessageBoxImage icon) - ShowMessageBox(Exception exception)
ShowDialogWindow	Use this method to create a new dialog box.
ShowWizardWindow	Use this method to display a custom editor inside a dialog box that includes **Next** and **Back** buttons for navigation.

ShowMessageBox

This method displays a message box that is a child of the custom wizard page editor. This member is overloaded: Table 73 contains a list of the members and a brief description of each. For complete information about each member (including syntax, usage, and examples), see the section that corresponds to each member.

Table 73. Overloaded Members for the ShowMessagBox Method

Expand table

Member	Description
ShowMessageBox(String message, String caption, MessageBoxImage icon)	Displays a message box with an icon and an **OK** button
ShowMessageBox(string message, string caption, MessageBoxButton button, MessageBoxImage icon)	Displays a message box with an icon and different possible combinations of buttons
ShowMessageBox(Exception exception)	Displays a message box that provides information about an exception and has an **OK** button

ShowMessageBox(String message, String caption, MessageBoxImage icon)

```
C#

void ShowMessageBox(String message, String caption, MessageBoxImage icon);
```

This method displays a message box with an **OK** button. See Table 74.

Table 74. Parameters for the ShowMessageBox(String message, String caption, MessageBoxImage icon) Method

Expand table

Parameter	Description
message	The message to display in the content area of the message box
caption	The text to show in the title bar of the dialog box
icon	The type of icon to show in the message box

ShowMessageBox(string message, string caption, MessageBoxButton button, MessageBoxImage icon)

```
C#

MessageBoxResult ShowMessageBox(string message, string caption,
MessageBoxButton button, MessageBoxImage icon);
```

This method displays a message box with the set of buttons you want shown and reports which button you selected. See Table 75.

Table 75. Parameters for the ShowMessageBox(string message, string caption, MessageBoxButton button, MessageBoxImage icon) Method

Expand table

Parameter	Description
message	The message to display in the content area of the message box

Parameter	Description
caption	The text to show in the title bar of the dialog box
button	Which buttons to show
icon	The type of icon to show in the message box

ShowMessageBox(Exception exception)

```
C#

void ShowMessageBox(Exception exception);
```

This method displays a message box that reports information about an exception. This message box has a single **OK** button. See Table 76.

Table 76. Parameters for the ShowMessageBox(Exception exception) Method

Expand table

Parameter	Description
exception	The exception that you want to report (The dialog box uses **exception.Message** as the contents.)

ShowDialogWindow

```
C#

void ShowDialogWindow(Type viewType, DialogInteraction dialogPayload);
```

This method creates a new dialog box, the contents of which is the text you supply in the **viewType** parameter. The UDI Designer creates a new instance of this type and wraps it in a dialog box that has **OK** and **Cancel** buttons.

You pass data to your control using the dialogPayload parameter. The **SampleEditor** solution in the SDK directory has an example of how to use this functionality.

ShowWizardWindow

```
C#

void ShowWizardWindow(Type viewType, DialogInteraction dialogPayload);
```

This method allows you to display a custom editor inside a dialog box that includes **Next** and **Back** buttons for navigation. Microsoft has not provided a sample for how to use this method.

UDI Wizard Configuration File Schema Reference

This file is consumed by the UDI Wizard and configured by the UDI Wizard Designer. This file is used to configure the:

- Wizard pages displayed in the UDI Wizard
- The sequence of the wizard pages in the UDI Wizard
- Settings for the fields on each wizard page
- Available StageGroups in the UDI Wizard Designer
- Available Stages within each deployment wizard in the UDI Wizard Designer

  77 lists the elements in the UDI Wizard Configuration File and their descriptions. The **Wizard** element is the root node for this reference.

Table 77. Elements in the UDI Wizard Configuration File and Their Descriptions

Expand table

Element name	Description
Data	Groups the individual DataItem elements within a Page element and is named by the **Name** attribute.
DataItem	Groups the individual Setter elements within a Page element. You can create hierarchical data by including one or more Data elements within a DataItem element. Each **DataItem** element represents an individual item. For example, a list of available drives might have a **DataItem** for the display name and another **DataItem** element for the corresponding drive letter.
Default	Specifies a default value for the field specified in the parent Field or RadioGroup element. The default is set to the value bracketed by this element.

Element name	Description
DLL	Specifies a DLL that is to be loaded and referenced by the UDI Wizard and the UDI Wizard Designer.
DLLs	Groups the individual DLL elements.
Error	Specifies a possible error code that can a task can return. The value of the error code is returned by the task's **HRESULT** and is trapped by this element to provide more specific error information.
ExitCode	Specifies a possible exit code for a task. The exit codes are return codes that the task expects. Create an **ExitCode** element for each possible exit code. Otherwise, you can specify an asterisk (*) in the **Value** attribute to handle return codes not listed in other **ExitCode** elements.
ExitCodes	Groups a set of ExitCode and Error elements for a Task element or an **Error** element.
Field	Specifies an instance of a control in a Page element that is used to provide customization with XML. Not all controls allow customization with XML—only controls that use the Field element.
Fields	Groups the individual Field elements within a Page element.
File	Specifies the source and destination for a file copy operation using the **Microsoft.Wizard.CopyFilesTask** task type. You can include a separate **File** element to copy more than one file in a single task.
Page	Specifies an instance of a page and includes all the configuration settings for the page.
PageRef	Specifies a reference to an instance of a page within a Stage within a StageGroup.
Pages	Groups the individual Page elements.
RadioGroup	Specifies a group of radio buttons within a Field element.
StageGroup	Specifies a group of one or more stages.
StageGroups	Groups a set of stage groups within a UDI Wizard configuration file.
Setter	Specifies a property setting of a value for a property that is named in the **Property** property.
Stage	Specifies a stage within a StageGroup and contains one or more PageRef elements.
Style	Groups the individual setter elements that configure the UDI Wizard look and feel, including the title shown at the top of the wizard and the banner image shown on the UDI Wizard.

Element name	Description
Task	Specifies a task that is to be run on the page specified in the parent Page element.
Tasks	Groups a set of tasks for a Page element.
Validator	Specifies a validator for the field control that is specified in the parent Field element.
Wizard	Specifies the root for all other elements.

Data

This element groups the individual DataItem elements within a Page element and is named by the **Name** attribute.

Element Information

Table 78 provides information about the Data element.

Table 78. Data Element Information

Expand table

Attribute	Value
Number of occurrences	Zero or more within each Page element (This element is optional.)
Parent elements	**Page**, **DataItem**
Contents	**DataItem**, **Setter**

Element Attributes

Table 79 lists the attributes of the Data element and provides a description of each.

Table 79. Attributes and Corresponding Values for the Data Element

Expand table

Attribute	Description
Name	Specifies the name of the Data element

Remarks

The **Name** attribute allows code to retrieve a specific set of data.

Example

None.

DataItem

This element groups the individual Setter elements within a Page element. You can create hierarchical data by including one or more Data elements within a DataItem element. Each **DataItem** element represents an individual item. For example, a list of available drives might have a **DataItem** for the display name and another **DataItem** element for the corresponding drive letter.

Element Information

Table 80 provides information about the DataItem element.

Table 80. DataItem Element Information

Expand table

Attribute	Value
Number of occurrences	Zero or more within each Data element (This element is optional.)
Parent elements	**Data**
Contents	**Data**, **Setter**

Element Attributes

This element has no attributes.

Remarks

None.

Example

None.

Default

This element specifies a default value for the field specified in the parent Field or RadioGroup element. The default is set to the value that this element brackets.

Element Information

Table 81 provides information about the Default element.

Table 81. Default Element Information

Expand table

Attribute	Value
Number of occurrences	Zero or more within a Field or RadioGroup element (This element is optional.)
Parent elements	**Field**, **RadioGroup**
Contents	Can be any well-formed XML content but is typically standard text

Element Attributes

This element has no attributes.

Remarks

None.

Example

In the following example, the default for the **TimeZone** field is set to "Pacific Standard Time":

```
XML
```

```
<Field Name="TimeZone" Enabled="true" VarName="OSDTimeZone" Summary="Time
Zone:">
  <Default>Pacific Standard Time</Default>
```

DLL

This element specifies a DLL for the UDI Wizard and UDI Wizard Designer to load and reference.

Element Information

Table 82 provides information about the DLL element.

Table 82. DLL Element Information

Expand table

Attribute	Value
Number of occurrences	One or more within the DLLs element
Parent element	**DLLs**
Contents	No content allowed for this element

Element Attributes

Table 83 lists the attributes of the DLL element and provides a description of each.

Table 83. Attributes and Corresponding Values for the DLL Element

Expand table

Attribute	Description
Name	Specifies the name of the DLL for the UDI Wizard and UDI Wizard Designer to reference

Remarks

None.

Example

```
XML
```

```xml
<DLLs>
  <DLL Name="OSDRefreshWizard.dll" />
  <DLL Name="SharedPages.dll" />
</DLLs>
```

DLLs

This element groups the individual DLL elements.

Element Information

Table 84 provides information about the DLLs element.

Table 84. DLLs Element Information

Expand table

Attribute	Value
Number of occurrences	One
Parent elements	**Wizard**
Contents	**DLL**

Element Attributes

This element has no attributes.

Remarks

None.

Example

```
XML
```

```
<DLLs>
   <DLL Name="OSDRefreshWizard.dll" />
   <DLL Name="SharedPages.dll" />
</DLLs>
```

Error

This element specifies a possible error code that a task can return. The value of the error code is returned and trapped by the task's **HRESULT** to provide more specific error information.

Element Information

Table 85 provides information about the Error element.

Table 85. Error Element Information

Expand table

Attribute	Value
Number of occurrences	Zero or more within each ExitCode element (This element is optional.)
Parent elements	**ExitCodes**
Contents	Any well-formed XML content

Element Attributes

Table 86 lists the attributes of the Error element and provides a description of each.

Table 86. Error Element Information

Expand table

Attribute	Description
State	Specifies the return state of a task that encountered an error. Typically, the value for this attribute is set to Error. This value is displayed in the **State** column on the wizard page in the UDI Wizard.
Text	Specifies the descriptive text about the error condition that the task encountered.

Attribute	Description
Type	Specifies whether this element represents an error, warning, or success. The value specified in**Type** must be unique within an ExitCodes element. The following are valid values for this element: - **0.**The element represent a success. - **1.** The element represents a warning. - **-1.** The element represents an error.
Value	Specifies the value of the code that the task returned as a numeric value. Specifying the value of an asterisk (*) indicates the default element for return codes that are not listed in other Error elements.

Remarks

None.

Example

None.

ExitCode

This element specifies a possible exit code for a task. The exit codes are return codes that the task expects. Create an **ExitCode** element for each possible exit code. Otherwise, you can specify an asterisk (*) in the **Value** attribute to handle return codes not listed in other **ExitCode** elements.

Element Information

Table 87 provides information about the ExitCode element.

Table 87. ExitCode Element Information

Expand table

Attribute	Value
Number of occurrences	Zero or more within each ExitCodes element (This element is optional.)
Parent elements	**ExitCodes**
Contents	At least one ExitCode element and zero or more Error elements

Element Attributes

Table 88 lists the attributes of the ExitCode element and provides a description of each.

Table 88. Attributes and Corresponding Values for the ExitCode Element

Expand table

Attribute	Description
State	Specifies the return state of a task. The value of this attribute is displayed in the **State** column on the corresponding wizard page in the UDI Wizard. You can use any values for this attribute that are meaningful for your task. The following are typical values used for this attribute: - Success - Warning - Error
Text	Specifies the descriptive text about the exist code of the task.
Type	Specifies whether this element represents an error, warning, or success. The value specified in type must be unique within an ExitCodes element. The following are valid values for this element: - **0.** The element represents a success. - **1.** The element represents a warning. - **-1.** The element represents an error.
Value	Specifies the value of the code that the task returned as a numeric value. Specifying the value of an asterisk (*) indicates the default element for return codes that are not listed in other ExitCode elements.

Remarks

None.

Example

None.

ExitCodes

This element groups a set of ExitCode and Error elements for a Task or an **Error** element.

Element Information

Table 89 provides information about the ExitCodes element.

Table 89. ExitCodes Element Information

Expand table

Attribute	Value
Number of occurrences	One within each Task element
Parent elements	**Task**
Contents	**Error**, **ExitCode**

Element Attributes

This element has no attributes.

Remarks

None.

Example

None.

Field

This element specifies an instance of a control in a Page element used to provide customization with XML. Not all controls allow customization with XML—only controls that use the Field element.

Element Information

Table 90 provides information about the Field element.

Table 90. Field Element Information

Expand table

Attribute	Value
Number of occurrences	Zero or more within each Field element (This element is optional.)
Parent elements	**Fields**
Contents	**Default**, **Validator**

Element Attributes

Table 91 lists the attributes of the Field element and provides a description of each.

Table 91. Attributes and Corresponding Values for the Field Element

Expand table

Attribute	Description
Enabled	Specifies whether the field is enabled for user input (The attribute can be set to True or False.)
Name	Specifies the name of the field
Summary	Specifies the descriptive text displayed on the **Summary** wizard page for the value that this field sets
VarName	Specifies the task sequence variable name read or configured using the field in the parent Field element

Remarks

This element can contain zero or more Default elements and zero or more Validator elements.

Example

None.

Fields

This element groups the individual Field elements within a Page element.

Element Information

Table 92 provides information about the Fields element.

Table 92. Fields Element Information

Expand table

Attribute	Value
Number of occurrences	Zero or more within each Page element (This element is optional.)
Parent elements	**Page**
Contents	**Field**, **RadioGroup**

Element Attributes

This element has no attributes.

Remarks

None.

Example

None.

File

This element specifies the source and destination for a file copy operation using the **Microsoft.Wizard.CopyFilesTask** task type. You can include a separate File element to copy more than one file in a single task.

Element Information

Table 93 provides information about the File element.

Table 93. File Element Information

Expand table

Attribute	Value
Number of occurrences	One or more for each task that has a task type of **Microsoft.Wizard.CopyFilesTask**
Parent elements	**Task**
Contents	None

Element Attributes

Table 94 lists the attributes of the File element and provides a description of each.

Table 94. Attributes and Corresponding Values for the File Element

Expand table

Attribute	Description
Dest	Specifies the fully qualified or relative path to the destination folder for the file specified in the **Source** attribute. Environment variables are allowed as a part of the path.
Source	Specifies the fully qualified or relative path to the source file that the **Microsoft.Wizard.CopyFilesTask** task type copies. This attribute supports wildcard characters so that multiple files can be copied using a single File element. Environment variables are allowed as part of the path.

Remarks

None.

Example

None.

Page

This element specifies an instance of a page and includes all the configuration settings for the page.

Element Information

Table 95 provides information about the Page element.

Table 95. Page Element Information

Expand table

Attribute	Value
Number of occurrences	One or more within each Pages element
Parent elements	**Pages**
Contents	**Data**, **Fields**, **Setter**, **Tasks**

Element Attributes

Table 96 lists the attributes of the Page element and provides a description of each.

Table 96. Attributes and Corresponding Values for the Page Element

Expand table

Attribute	Description
DisplayName	Specifies the user-friendly name of the wizard page displayed in the UDI Wizard Designer. This name is usually more descriptive than the **Name** attribute.
Name	Specifies the name of the wizard page displayed in the UDI Wizard Designer.
Type	Specifies the type of wizard page that directly relates to a specific wizard page within a DLL.

Remarks

None.

Example

None.

PageRef

This element specifies a reference to an instance of a page within a Stage within a StageGroup.

Element Information

Table 97 provides information about the PageRef element.

Table 97. PageRef Element Information

Expand table

Attribute	Value
Number of occurrences	One or more within a Stage element
Parent elements	**Stage**
Contents	None

Element Attributes

Table 98 lists the attribute of the PageRef element and provides a description of it.

Table 98. Attributes and Corresponding Values for the PageRef Element

Expand table

Attribute	Description
Page	Specifies the instance of a page within a Stage within a StageGroup. Set this value to the Name attribute of a Page element.

Remarks

None.

Example

None.

Pages

This element groups the individual Page elements.

Element Information

Table 99 provides information about the Pages element.

Table 99. Pages Element Information

Expand table

Attribute	Value
Number of occurrences	One
Parent elements	**Wizard**
Contents	**Page**

Element Attributes

This element has no attributes.

Remarks

None.

Example

C#

```
<Pages>
   + <Page Name="WelcomePage" DisplayName="Welcome"
Type="Microsoft.SharedPages.WelcomePage">
   + <Page Name="ConfigScanPage" DisplayName="Deployment Readiness"
Type="Microsoft.OSDRefresh.ConfigScanPage">
   + <Page Name="ConfigScanBareMetal" DisplayName="Deployment Readiness"
Type="Microsoft.OSDRefresh.ConfigScanPage">
   + <Page Name="RebootPage" DisplayName="Reboot"
Type="Microsoft.OSDRefresh.RebootPage">
   + <Page Name="WelcomePageReplace" DisplayName="Welcome"
Type="Microsoft.SharedPages.WelcomePage">
   + <Page Name="VolumePage" DisplayName="Volume"
Type="Microsoft.OSDRefresh.VolumePage">
   + <Page Name="UserRestorePage" DisplayName="Select Target"
```

```
Type="Microsoft.OSDRefresh.UserStatePage">
   + <Page Name="ComputerPage" DisplayName="New Computer Details"
Type="Microsoft.OSDRefresh.ComputerPage">
   + <Page Name="AdminAccounts" DisplayName="Administrator Password"
Type="Microsoft.SharedPages.AdminAccountsPage">
   + <Page Name="UDAPage" DisplayName="User Device Affinity"
Type="Microsoft.OSDRefresh.UDAPage">
   + <Page Name="LanguagePage" DisplayName="Language"
Type="Microsoft.OSDRefresh.LanguagePage">
   + <Page Name="ApplicationPage" DisplayName="Install Programs"
Type="Microsoft.OSDRefresh.ApplicationPage">
     <Page Name="SummaryPage" DisplayName="Summary"
Type="Microsoft.Shared.SummaryPage" />
   + <Page Name="UserCapturePageOldPC" DisplayName="Select Target"
Type="Microsoft.OSDRefresh.UserStatePage">
   + <Page Name="ProgressPage" DisplayName="Capture Data"
Type="Microsoft.OSDRefresh.ProgressPage">
   + <Page Name="RebootAfterCapture" DisplayName="Reboot"
Type="Microsoft.OSDRefresh.RebootPage">
</Pages>
```

RadioGroup

This element specifies a group of radio buttons with in a Field element.

Element Information

Table 100 provides information about the RadioGroup element.

Table 100. RadioGroup Element Information

Expand table

Attribute	Value
Number of occurrences	Zero or more within a Fields element (This element is optional.)
Parent elements	**Fields**
Contents	**Default**

Element Attributes

Table 101 lists the attributes of the RadioGroup element and provides a description of each.

Table 101. Attributes and Corresponding Values for the RadioGroup Element

Expand table

Attribute	Description
Locked	Specifies whether the group of radio buttons is enabled for user input. The attribute can be set to: - **True**. Specifies that the radio buttons are disabled and users cannot select a radio button in the group. - **False**. Specifies that the radio buttons are enabled and users can select a radio button in the group.
Name	Specifies the name of the radio option group.

Remarks

None.

Example

None.

StageGroup

This element specifies a deployment stage group.

Element Information

Table 102 provides information about the StageGroup element.

Table 102. StageGroup Element Information

Expand table

Attribute	Value
Number of occurrences	One or more within a StageGroups element
Parent elements	**StageGroups**

Attribute	Value
Contents	**Stage**

Element Attributes

Table 103 lists the attributes of the StageGroup element and a description of the attribute.

Table 103. Attributes and Corresponding Values for the StageGroup Element

Expand table

Attribute	Description
DisplayName	Specifies the user-friendly name of the stage group displayed in the UDI Wizard Designer. This name is usually more descriptive than the **Name** attribute.

Remarks

None.

Example

None.

StageGroups

This element groups a set of stage groups within a UDI Wizard configuration file.

Element Information

Table 104 provides information about the StageGroups element.

Table 104. StageGroups Element Information

Expand table

Attribute	Value
Number of occurrences	Zero or one within a Wizard element
Parent elements	**Wizard**
Contents	**StageGroup**

Element Attributes

This element has no attributes.

Remarks

None.

Example

None.

Setter

This element specifies a property setting for the value for a property that is named in the **Property** property.

Element Information

Table 105 provides information about the Setter element.

Table 105. Setter Element Information

Expand table

Attribute	Value
Number of occurrences	Zero or more within each parent element (This element is optional.)
Parent elements	**Data**, **DataItem**, **Page**, **Style**, **Task**, **Validator**
Contents	Contains a string value in the **Property** attribute

Element Attributes

Table 106 lists the attribute of the Setter element and provides a description of it.

Table 106. Attributes and Corresponding Values for the Setter Element

Expand table

Attribute	Description
Property	Specifies the property name being set. The property name is set to the value that this attribute brackets.

Remarks

None.

Example

None.

Stage

This element specifies a Stage within a StageGroup and contains one or more PageRef elements.

Element Information

Table 107 provides information about the Stage element.

Table 107. Stage Element Information

Expand table

Attribute	Value
Number of occurrences	One or more within a StageGroup element
Parent elements	**StageGroup**
Contents	**PageRef**

Element Attributes

Table 108 lists the attributes of the Stage element and provides a description of each.

Table 108. Attributes and Corresponding Values for the Stage Element

Expand table

Attribute	Description
DisplayName	Specifies the user-friendly name of the wizard page displayed in the UDI Wizard Designer. This name is usually more descriptive than the **Name** attribute.
Name	Specifies the name of the stage. The value of this element is used when starting the UDI Wizard with the **/stage: name** command line parameter.

Remarks

None.

Example

None.

Style

This element groups the individual Setter elements that configure the UDI Wizard look and feel, including the title shown at the top of the wizard and the banner image shown on the UDI Wizard.

Element Information

Table 109 provides information about the Style element.

Table 109. Style Element Information

Expand table

Attribute	Value
Number of occurrences	One

Attribute	Value
Parent elements	**Wizard**
Contents	**Setter**

Element Attributes

This element has no attributes.

Remarks

None.

Example

XML

```
<Style>
  <Setter Property="bannerFilename">UDI_Wizard_Banner.bmp</Setter>
  <Setter Property="title">Operating System Deployment (OSD) Refresh
Wizard</Setter>
</Style>
```

Task

This element specifies a task that is to be run on the page specified in the parent Page element.

Element Information

Table 110 provides information about the Task element.

Table 110. Task Element Information

Expand table

Attribute	Value
Number of occurrences	One or more within a Tasks element
Parent elements	**Tasks**

Attribute	Value
Contents	**ExitCodes**, **File**, **Setter**

Element Attributes

Table 111 lists the attributes of the Task element and provides a description of each.

Table 111. Attributes and Corresponding Values for the Task Element

Expand table

Attribute	Description
DependsOn	Specifies whether the task is dependent on another task. The value of this attribute is set to the **Name** attribute of another Task element. **Note:** This attribute cannot be configured using the UDI Wizard Designer. However, you can manually add this attribute to a Task element by directly modifying the .xml file.
DisplayName	Specifies the user-friendly name of the task displayed in the UDI Wizard Designer. This name is usually more descriptive than the **Name** attribute.
Name	Specifies the name of the task. This name must be unique.
Type	Specifies the task type for the task to be run, which is defined in the DLL that contains the task.

Remarks

None.

Example

None.

Tasks

This element groups a set of tasks for a Page element.

Element Information

Table 112 provides information about the Tasks element.

Table 112. Tasks Element Information

Expand table

Attribute	Value
Number of occurrences	Zero or one within each Page element (This element is optional.)
Parent elements	**Page**
Contents	**Task**

Element Attributes

Table 113 lists the attributes of the Tasks element and provides a description of each.

Table 113. Attributes and Corresponding Values for the Tasks Element

Expand table

Attribute	Description
NameTitle	Specifies the caption that appears at the top of the column that contains the name of the tasks in the appropriate wizard page.
StatusTitle	Specifies the caption that appears at the top of the column that contains the status of the tasks in the appropriate wizard page.

Remarks

None.

Example

None.

Validator

This element specifies a validator for the field control that is specified in the parent Field element.

Element Information

Table 114 provides information about the Validator element.

Table 114. Validator Element Information

Expand table

Attribute	Value
Number of occurrences	Zero or one within a Field element
Parent elements	**Field**
Contents	**Setter**

Element Attributes

Table 115 lists the attribute of the Validator element and provides a description of it.

Table 115. Attributes and Corresponding Values for the Validator Element

Expand table

Attribute	Description
Type	Specifies the type for the validator, which is defined in the DLL that contains the validator

Remarks

None.

Example

None.

Wizard

This element specifies the root for all other elements.

Element Information

Table 116 provides information about the Wizard element.

Table 116. Wizard Element Information

Expand table

Attribute	Value
Number of occurrences	One
Parent elements	None
Contents	**DLLs**, **Pages**, **StageGroups**, **Style**

Element Attributes

This element has no attributes.

Remarks

None.

Example

C#

```
<Wizard>
   + <DLLs>
   + <Style>
   + <Pages>
   + <StageGroups>
</Wizard>
```

Feedback

Was this page helpful? Yes No

Provide product feedback