Microsoft | Solution Accelerators

Microsoft Deployment Toolkit 2013

Microsoft Deployment Toolkit Samples Guide

Published: September 2013

For the latest information and to leave feedback, please visit Microsoft Connect at http://connect.microsoft.com.

Microsoft

Contents

Introduction to the Microsoft Deployment Toolkit Samples Guide

This guide is part of Microsoft® Deployment Toolkit (MDT) 2013 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 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.

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 Table 1 can further assist in customizing the configuration settings for the environment.

Table 1. Deployment Guidance in MDT

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	Further customize the configuration files used in Zero Touch Installation (ZTI) and LTI

Guide	This guide offers assistance to help
Deployment Toolkit	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. You can load the MDT Windows PowerShell snap-in using any one of the following methods:

- Load the MDT Windows PowerShell snap-in using the Window PowerShell Modules console as described in Load the MDT Windows PowerShell Snap-In Using the Import System Modules Task.
- 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 Import System Modules Task

The Import System Modules task automatically includes all the Windows PowerShell modules and snap-ins that are in the modules in the %Windir%\System32\WindowsPowerShell\1.0\Modules directory. MDT automatically installs the MDT Windows PowerShell snap-in—Microsoft.BDD.SnapIn—in that folder during the MDT installation process.

Note The Import System Modules task is available only in Windows 7 and Windows Server 2008 R2 when Windows PowerShell 3.0 is not installed on the computer. Beginning with Windows PowerShell 3.0, modules are imported automatically the first time that you use a cmdlet in the module.

You can start a Windows PowerShell console with the Import System Modules task by performing one of the following procedures:

- In the taskbar, right-click the **Windows PowerShell** icon, and then click **Import System Modules**.
- Click **Start**, point to **Administrative Tools** and then click **Windows PowerShell Modules**.

For more information on staring a Windows PowerShell console with Import System Modules, see Starting Windows PowerShell with Import System Modules.

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:

```
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 New-PSDrive 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:

   ```
   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:

   ```
   ([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 New-PSDrive cmdlet, as shown in the following example:

   ```
   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:

   ```
   dir 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:

```
$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 New-PSDrive and **Add-MDTPersistentDrive** cmdlets, as shown in the following example:

```
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:

```
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:

```
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:

   ```
   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:

   ```
   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:

   ```
   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:

   ```
   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:

   ```
   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:

   ```
   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:

   ```
   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:

```
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:

```
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:

```
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:

   ```
   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:

   ```
   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 item into a deployment share using Windows PowerShell, see Automating Population of a Deployment Share.

Table 2 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.

Table 2. MDT Windows PowerShell Cmdlets for Importing Deployment Share Items

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:

   ```
   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:

   ```
   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:

```
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:

```
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:

```
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:

```
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:

   ```
   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:

   ```
   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:

```
$List=Import-CSV "C:\MDT\Import-MDT-Apps.csv"
ForEach-Object ($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.

   For more information about how to create a .csv file for use with the Import-CSV cmdlet, see Using the Import-Csv Cmdlet.

2. 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.

3. 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.

4. 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.

5. 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.

6. 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:

   ```
   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:

   ```
   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 **mkdir** command, as shown in the following example:

   ```
   mkdir "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:

   ```
   dir "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:

   ```
   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:

   ```
   dir "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:

   ```
   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:

   ```
   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 **mkdir** command, as shown in the following example:

   ```
   rmdir "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:

   ```
   dir "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:

   ```
   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:

```
dir "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:

   ```
   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:

   ```
   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:

   ```
   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:

   ```
   dir "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:

   ```
   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:

   ```
   dir "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:

   ```
   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:

   ```
   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:

   ```
   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:

   ```
   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:

   ```
   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:

   ```
   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:

   ```
   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 example in Listing 1, 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 Listing 1, 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 example in Listing 1).
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 example in Listing 1.
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

```
[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 (as illustrated in Listing 2).

   Listing 2. SQL Query to Populate the Table

   ```
   use [MDTDB]
   go
   INSERT INTO [PackageMapping] (ARPName, Packages)
   VALUES('Office12.0', 'XXX0000F:Install Office 2010
   Professional Plus')
   go
   ```

   In Listing 2, 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 (as illustrated in Listing 3).

   Listing 3. SQL Query to Create a Stored Procedure to Simplify Joining the Information with Inventory Data

```
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 Listing 3 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` in Listing 3) 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 (as illustrated in Listing 4).

Listing 4. Excerpt from CustomSettings.ini That Illustrates the DynamicPackages Section in the Priority List

```
[Settings]
…
Priority=MacAddress, DefaultGateway, DynamicPackages, Default
```

…

5. Create a `[DynamicPackages]` section to specify the name of a database section (as illustrated in Listing 5).

   Listing 5. Excerpt from CustomSettings.ini That Illustrates the DynamicPackages Section with the Name of a Database Section

   ```
   [DynamicPackages]
   SQLDefault=DB_DynamicPackages
   ```

6. Create a database section to specify the database information and query details (as illustrated in Listing 6).

   Listing 6. Excerpt from CustomSettings.ini That Illustrates the Database Section

   ```
   [DB_DynamicPackages]
   SQLServer=SERVER1
   Database=MDTDB
   StoredProcedure=RetrievePackages
   Parameters=MacAddress
   SQLShare=Logs
   Instance=SQLEnterprise2005
   Port=1433
   Netlib=DBNMPNTW
   ```

   In the example in Listing 6, 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 (shown in Listing 7).

Listing 7. Sample SQL Query Created Automatically Based on the Database Section

```
EXECUTE RetrievePackages ?
```

The actual value of the **MACAddress** custom key will be substituted for the corresponding `?` in Listing 7. 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

Listing 8 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.

Note The properties used to skip wizard pages are in bold type in the listing.

Listing 8. CustomSettings.ini File for a Refresh Computer Scenario

```
[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

Listing 9 illustrates 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.

Note The properties that are used to skip wizard pages are in bold type in the listing.

Listing 9. CustomSettings.ini File for a New Computer Scenario

```
[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=http://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 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.

Listing 10 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.

Listing 10. CustomSettings.ini File to Call a Particular Web Service Scenario

```
[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 Listing 11.

Listing 11. CustomSettings.ini File to Call a Particular Web Service 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):

```
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. Click **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then click **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, click ***task_sequence*** (where *task_sequence* is the task sequence to modify).
4. In the Actions pane, click **Properties**.
5. Click the **Task Sequence** tab, browse to ***group*** (where *group* is the group in which to run the ZTIConnec.wsf script), and click **Add**. Click **General**, and then click **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 information listed in Table 3 (accept default values if none are specified), and then click **Apply**.

   Table 3. Completing the Properties Tab of the New Task

In this box	Do this
Name	Type **Connect to *server*** (where *server* is the name of the server to which to connect).

In this box	Do this
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 information listed in Table 4 (accept default values if none are specified), and then click **OK**.

Table 4. Completing the Options Tab of the New Task

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. Click **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then click **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, click **Properties**.
4. In the **Properties** dialog box, click 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.

    Listing 12 and Listing 13 show CustomSettings.ini and the user exit script, respectively.

    Listing 12. CustomSettings.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
```

Listing 13. User Exit Script

```
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. Click **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then click **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, click ***task_sequence*** (where *task_sequence* is the name of the task sequence to configure).
4. In the Actions pane, click **Properties**.
5. In the ***task_sequence*** **Properties** dialog box, on the **Task Sequence** tab, click ***step*** (where *step* is the name of the task sequence step to configure), and then click the **Options** tab.

On the **Options** tab of the task sequence step, perform the following actions:

- **Add.** Click this button to add a condition to the task sequence step.
- **Remove.** Click this button to remove an existing condition in a task sequence step.
- **Edit.** Click 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 Table 7, which are configured in the **IF Statement Properties** dialog box.

Table 5. Conditions Available in IF Statements

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), click **Add**, and then click **If statement**.
2. In the **If Statement Properties** dialog box, click ***condition*** (where *condition* is one of the conditions listed in Table 7), and then click **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), click **Add Condition**, and then click **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, click **equal**.
4. In the **Value** box, type **1**, and then click **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, click the **Capture Network Settings** task.
2. Click **Add Condition**, and then click **Operating System Version**.
3. In the **Architecture** box, click the relevant server. For this example, click **x86**.
4. In the **Operating system** box, click the operating system and version for which to set a condition. For this example, click **x86 Windows 2003**.
5. In the **Condition** box, click the relevant condition, and then click **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 2007 R3

1. In Configuration Manager 2007 R3, 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. Click the **Install Software** task sequence step created in step 2, and then click the **Options** tab.
4. Click **Add Condition**, and then click **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 click **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 click **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. Click the task created in step 1.
4. Click **Add Condition**, and then click **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. Click **Less than or equals** for the condition.
8. In the **Date** box, click **12/31/2007**.

9. In the **Time** box, click **12:00:00 AM**, and then click **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. Click the first task in the sequence, and then click the **Options** tab.
3. Click **Add Condition**, and then click **Registry Setting**.
4. In the **Root key** list, click **HKEY_LOCAL_MACHINE**.
5. In the **Key** box, type **SOFTWARE\WOODGROVE**.
6. Click **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. Click **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. Click the first task in the sequence, and then click the **Options** tab.

3. Click **Add Condition**, and then click **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%"**. Click **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. Click the first task in the sequence, and then click the **Options** tab.
3. Click **Add Condition**, and then click **Installed Software**.
4. Click **Browse**, and then click 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. Click **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. Click the task sequence step to which to add the condition, and then click the **Options** tab.

3. Click **Add condition**, click **If Statement**, and then click **All conditions**. Click **OK**.
4. Click the condition statement, click **Add condition**, and then click **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%”**. Click **OK**.
6. Click the **IF** statement, and then click **Add condition**. Click **If statement**, and then click **Any condition**. Click **OK**.
7. Click the second **IF** statement. Click **Add condition**, and then click **Operating System Version**.
8. In the **Architecture** box, click the architecture for the servers. For this example, click **x86**.
9. In the **Operating system** box, click the operating system and version. For this example, click **x86 Windows 2003 original release**. Click **OK**.
10. Click the second **IF** statement. Click **Add condition**, and then click **Operating System Version**.
11. In the **Architecture** box, click the architecture for the servers. For this example, click **x86**.
12. In the **Operating system** box, click the operating system and version. For this example, click **x86 Windows 2008 original release**. Click **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. Table 8 lists the required components for the master deployment server.

Table 6. Master Deployment Server

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

Required component	Purpose/comment
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. Table 9 lists the required components for the child deployment server.

Table 7. Child Deployment Servers

Required component	Purpose/comment
Windows Server 2003 R2	Required to support DFS-R
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 click **New Replication Group**.
4. In the New Replication Group Wizard, on the **Replication Group Type** page, click **New Multipurpose Replication Group**.
5. Click **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. Click **Next**.
8. On the **Replication Group Members** page, perform these steps:
   a. Click **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. Click **OK**.
9. Click **Next**.
10. On the **Topology Selection** page, click **Hub and Spoke**, and then click **Next**.
11. On the **Hub Members** page, click the master deployment server, and then click **Add**.
12. Click **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. Click **Next**.
15. On the **Replication Group Schedule and Bandwidth** page, specify a schedule for replicating the content between servers.
16. Click **Next**.
17. On the **Primary Member** page, in the **Primary Member** box, click the master deployment server.
18. Click **Next**.
19. On the **Folders to Replicate** page, click **Add**, and then perform these steps:
   a. In the **Local Path of the folder to replicate** box, click **Browse** to go to the *X:*\Deployment folder (where *X* is the drive letter on the deployment server).
   b. Click **Use name based on path**.
   c. Click **OK**.
   d. Click **Add**.
   e. In the **Add Folder to Replicate** dialog box, click **Browse** to go to the *X:*\RemoteInstall\Boot folder.
   f. Click **Use name based on path**.
20. Click **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 click **Edit**.
   b. In the **Edit Local Path** dialog box, click **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. Click **OK**.

22. Click **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 click **Edit**.
    b. In the **Edit Local Path** dialog box, click **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. Click **OK**.
24. Click **Next**.
25. On the **Remote Settings and Create Replication Group** page, click **Create** to complete the New Replication Group Wizard.
26. On the **Confirmation** page, click **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 Securing 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, click **Local and Remote connections**.

To prepare for SQL Server replication on the child deployment server

1. In the **SQL Server Surface Area Configuration** box, click **Local and Remote connections**.
2. 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 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 click **Configure Distribution**.
3. In the Configure Distribution Wizard, click **Next**.
4. On the **Distributor** page, click **will act as its own Distributor; SQL Server will create a distribution database and log**, and then click **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, click **Next**.
7. On the **Publishers** page, click the master deployment server to set it as the distributor, and then click **Next**.
8. On the **Wizard Actions** page, click **Configure Distribution**, and then click **Next**.
9. Click **Finish**, and then click **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 click **Publisher Properties**.
2. On the **Publisher Properties** page, perform these steps:

a. Click **Publisher Databases**.

b. Click the MDT DB, and then click **Transactional**.

c. Click **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 click **New Publication**.
2. In the New Publication Wizard, click **Next**.
3. On the **Publication Database** page, click the MDT DB, and then click **Next**.
4. On the **Publication Type** page, click **Snapshot publication**, and then click **Next**.
5. On the **Articles** page, select all **Tables**, **Stored Procedures**, and **Views**, and then click **Next**.
6. On the **Articles Issues** page, click **Next**.
7. On the **Filter Table Rows** page, click **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. Click **Schedule the Snapshot Agent to run at the following times**.

   c. Click **Change**.

   Note Specify a schedule that will occur one hour before the database replicates.

9. Click **Next**.
10. On the **Agent Security** page, click the account under which the snapshot agent will run, and then click **Next**.
11. On the **Wizard Actions** page, click **Create the publication**, and then click **Next**.
12. On the **Complete the Wizard** page, in the **publication name** box, type a descriptive publication name.
13. Click **Finish** to complete the wizard, and then click **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 click **New Subscriptions**.
3. In the New Subscriptions Wizard, click **Next**.
4. On the **Publication** page, click the publication created in the previous section.
5. On the **Distribution Agent Location** page, click **Run all agents at the Distributor SERVERNAME (push subscriptions)**, and then click **Next**.
6. On the **Subscribers** page, add each of the child deployment servers by performing the following steps:
   a. Click **Add Subscriber**, and then click **Add SQL Server Subscriber**.
   b. Add each child deployment server.
   c. For each child deployment server added, in the **Subscription Database** box, click 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. Click **Next**.
8. On the **Distribution Agent Security** page, click **…** to open the **Distribution Agent Security** dialog box.
9. Type the details of the account to use for the distribution agent, and then click **Next**.
10. On the **Synchronization Schedule** page, perform these steps:
    a. In the **Agent Schedule** box, click **<Define schedule>**.
    b. Specify the schedule that should be used to replicate the database between master and child deployment servers, and then click **Next**.
11. On the **Initialize Subscription** page, click **Next**.
12. On the **Wizard Actions** page, click **Create the subscription(s)**, and then click **Next**.

13. Click **Finish**, and then click **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. Click **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then click **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, click **Properties**.
4. Click 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. Click **Edit Bootstrap.ini**.

6. Configure BootStrap.ini to use the **%WDSServer%** property by adding or changing the **DeployRoot** value to **DeployRoot=\\%WDSServer%\Deployment$**.
7. Click **File**, and then click **Save** to save the changes to the BootStrap.ini file.
8. Click **OK**.

   The deployment share and LiteTouchPE_x86.wim Windows PE environment need to be updated.

9. In the Actions pane, click **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 click **Next**.
11. On the **Summary** page, verify the details are correct, and then click **Next**.
12. On the **Confirmation** page, click **Finish**.

Listing 15 illustrates CustomSettings.ini after performing the steps outlined in this section.

Listing 14. Sample CustomSettings.ini Configured for Scalable LTI Deployment Infrastructure

```
[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 Table 10.

Table 8. Methods for Automatically Determining the Deployment Server

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 Table 10 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

Listing 16 illustrates how the LocationServer.xml file is created using each of these properties using a sample LocationServer.xml file configured for multiple locations.

Listing 15. Example LocationServer.xml File to Support Multiple Locations

```
<?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 (as shown in Listing 16) 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 Listing 17.

Listing 16. Example LocationServer.xml File to Support Multiple Servers at Multiple Locations

```
<?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.

Listing 18 illustrates a sample LocationServer.xml file configured for multiple servers at different locations.

Listing 17. Example LocationServer.xml File for Different Locations

```
<?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 Listing 18, 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. Click **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then click **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, click **Properties**.
7. In the ***deployment_share*** **Properties** dialog box (where *deployment*_share is the name of the deployment share), perform these steps:
   a. Click 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 click **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. Click **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then click **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, click **Properties**.
4. In the ***deployment_share*** **Properties** dialog box (where *deployment_share* is the name of the deployment share), click the **Rules** tab, and then click **Edit BootStrap.ini**.
5. Remove the **DeployRoot** value (for example, **DeployRoot=\\Server\Deployment$**).
6. Click **File**, and then click **Save** to save the changes to the BootStrap.ini file.
7. Click **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. Click **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then click **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, click **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 click **Next**.
5. On the **Summary** page, verify the details are correct, and then click **Next**.
6. On the **Confirmation** page, click **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. Click **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then click **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, click **New Task Sequence**.

   The New Task Sequence Wizard starts.
4. Complete the New Task Sequence Wizard by using the information in Table 11. Accept the default values unless otherwise specified.

   Table 9. Information for Completing the New Task Sequence Wizard

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. Click **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 click **Next**.
Summary	Verify that the configuration details are correct, and then click **Next**.
Confirmation	Click **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. Click **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then click **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, click **New Task Sequence**.

   The New Task Sequence Wizard starts.
4. Complete the New Task Sequence Wizard by using the information in Table 12. Accept the default values unless otherwise specified.

   Table 10. Information for Completing the New Task Sequence Wizard

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. Click **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 click **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 click **Next**.
Specify Product Key	Select **Do not specify a product key at this time**, and then click **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. Click **Next**.
Admin Password	In **Administrator Password** and **Please confirm Administrator Password**, type **P@ssw0rd**, and then click **Finish**.

On this wizard page	Do this
Confirmation	Click **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. Click **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then click **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, click **Properties**.

   The **Properties** dialog box appears.
4. In the **Properties** dialog box, click the **Rules** tab.
5. On the **Rules** tab, modify the CustomSettings.ini file to reflect the necessary changes as shown in Listing 19. Make any additional modifications the environment requires.

   Listing 18. Customized CustomSettings.ini File

```
[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, click **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. Click **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then click **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, click **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 click **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. Click **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then click **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, click **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 click **Next**.
5. On the **Summary** page, verify the details are correct, and then click **Next**.
6. On the **Confirmation** page, click **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 information in Table 13. Accept the default values unless otherwise specified.

   Table 11. Information for Completing the Windows Deployment Wizard

On this wizard page	Do this
Welcome to Deployment	Click **Run the Deployment Wizard** to install a new operating system, and then click **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. Click **OK**.
Select a task sequence to execute on this computer.	Click **Perform Replace Computer Scenario on Existing Computer**, and then click **Next**.
Specify where to save your data and settings	Click **Next**.
Specify where to save a complete computer backup	Click **Do not back up the existing computer**, and then click **Next**.
Ready to begin	Click **Begin**.

   If any errors or warnings occur, consult the MDT document *Troubleshooting Reference.*

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

   If any errors or warnings occurred, review the errors or warnings and record any diagnostic information.

4. In the **Deployment Summary** dialog box, click **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 information in Table 14. Accept the default values unless otherwise specified.

Table 12. Information for Completing the Windows Deployment Wizard

On this wizard page	Do this
Welcome to Deployment	Click **Run the Deployment Wizard to install a new operating system**, and then click **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. Click **OK**.
Select a task sequence to execute on this computer.	Click **Perform Replace Computer Scenario on New Computer**, and then click **Next**.
Configure the computer name	In **Computer name**, type **WDG-NEW-02**, and then click **Next**.
Join the computer to a domain or workgroup	Click **Next**.
Specify whether to restore user data	1. Click **Specify a location**. 2. In **Location**, type **\\WDG-MDT-01\UserStateCapture$\WDG-EXIST-01**. 3. Click **Next**.
Locale Selection	Click **Next**.

On this wizard page	Do this
Set the Time Zone	Click **Next**.
Specify whether to capture an image	Click **Do not capture an image of this computer**, and then click **Next**.
Specify the BitLocker configuration	Click **Do not enable BitLocker for this computer**, and then click **Next**.
Ready to begin	Click **Begin**.

If any errors or warnings should occur, consult the MDT document *Troubleshooting Reference.*

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

   If any errors or warnings occurred, review the errors or warnings and record any diagnostic information.

4. In the **Deployment Summary** dialog box, click **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 TRACE32; this tool is available in the System Center Configuration Manager 2007 Toolkit V2.
- **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.

Listing 20 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.

Listing 19. Example Script Using ZTIUtility Logging: ZTI_Example.wsf

```
<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 Listing 20, **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.

Table 15 details some useful functions available, and their output. For a full list of available functions, refer to the ZTIUtility.vbs file.

Table 13. Useful ZTI Utility Functions

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
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

Function	Output
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
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. Click **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then click **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, click **New Application**.

   The New Application Wizard starts.

5. Complete the New Application Wizard using the information in Table 16; accept defaults unless otherwise specified.

   Table 14. Information for Completing the New Application Wizard

On this wizard page	Do this
Application Type	Click **Application without source files or elsewhere on the network**, and then click **Next**.
Details	Complete this page based on the information from the application, and then click **Next**.
Command Details	1. In the **Command line** box, type **cscript.exe %SCRIPTROOT%*custom_code*** (where

On this wizard page	Do this
	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. Click **Next**.
Summary	Verify that the configuration settings are correct, and then click **Next**.
Confirmation	Click **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. Click **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then click **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, click ***task_sequence*** (where *task_sequence* is the name of the task sequence that runs the custom code).
5. In the Actions pane, click **Properties**.
6. In the ***task_sequence*** **Properties** dialog box, click the **Task Sequence** tab.
7. In the console tree, go to *group* (where *group* is the group to add the task sequence step).
8. Click **Add**, click **General**, and then click **Run Command Line**.
9. In the console tree, click **Run Command Line**, and then click 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 click **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. Click **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then click **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, click **Import Drivers**.

   The Import Device Driver Wizard starts.
5. On the **Specify Directory** page, in the **Drive source** directory section, click **Browse** to go to the folder that contains the new device drivers, and then click **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 click **Next** to import the drivers into the Deployment Workbench.
7. On the **Confirmation** page, click **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

1. Click **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then click **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, click **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 click **Next**.
5. On the **Summary** page, verify that the details are correct, and then click **Next**.
6. On the **Confirmation** page, click **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. Click **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then click **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, click **New Application**.

   The New Application Wizard starts.
5. On the **Application Type** page, click **Application with source files**, and then click **Next**.
6. On the **Details** page, type relevant details about the application, and then click **Next**.

7. On the **Source** page, in the **Source directory** section, click **Browse** to go to, and then click the directory that contains the device driver application source files. Click **OK**.
8. Click **Next**.
9. On the **Destination** page, type a name for the destination directory, and then click **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 click **Next** to import the device driver application into the Deployment Workbench.
12. On the **Confirmation** page, click **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. Click **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then click **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, click ***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, click **Properties**.
5. In the ***task_sequence*** **Properties** 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, click **Add**, and then click **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, click **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, click **Add**, and then click **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. Click **OK** to submit the query, and then click **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. Click **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then click **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, click ***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, click **Properties**.
5. In the ***task_sequence*** **Properties** dialog box, click 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, click **Add**, click **General**, and then click **Install Application**.

   The **Install Application** task sequence step appears in the details pane.
8. In the details pane, click **Install Application**.
9. On the **Properties** tab, click **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. Listing 21 shows an example of the CustomSettings.ini file.

Listing 20. Sample CustomSettings.ini Configured for a Hardware-Specific Application Installation

```
[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.

Listing 21 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. Click **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then click **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, click **Properties**.
4. Click 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. Click **OK** to submit the changes.
7. In the details pane, click *deployment_share* (where *deployment_share* is the name of the deployment share to configure).
8. In the Actions pane, click **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 click **Next**.
10. On the **Summary** page, verify the details are correct, and then click **Next**.
11. On the **Confirmation** page, click **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

1. Click **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then click **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 configure).
3. In the details pane, click ***device_driver_application*** (where *device_driver_application* is the application to be hidden from the Deployment Wizard).
4. In the Actions pane, click **Properties**.
5. On the **General** tab, select the **Hide the application in the Deployment Wizard** check box.
6. Click **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. Click **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then click **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, click **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. Click **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then click **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, click **Properties**.
4. In the ***deployment_share*** **Properties** dialog box, click the **Windows PE *platform* Settings** tab (where platform is the architecture of the Windows PE image to be configured).
5. In the **Lite Touch Boot Image Settings** area, select the **Generate a Lite Touch bootable RAM disk ISO image** check box.
6. Click the **Windows PE *platform* Components** tab (where *platform* is the architecture of the Windows PE image to be configured).
7. In the **Driver Injection** section, click the appropriate driver types to include.

   Note This step is not necessary if Windows PE already includes the necessary device drivers.

8. In the **Driver Injection** section, in the **Selection profile** list, select the appropriate driver selection profile.
9. In the **Properties** dialog box, click **OK**.

   Note This step is not necessary if Windows PE already includes the necessary device drivers.

10. In the details pane, click ***deployment_share*** (where *deployment_share* is the name of the deployment share to configure).
11. In the Actions pane, click **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 click **Next**.
13. On the **Summary** page, verify the details are correct, and then click **Next**.
14. On the **Confirmation** page, click **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

1. Start the Windows Deployment Services console, and then connect to Windows Deployment Services.
2. In the console tree, right-click **Boot Images**, and then click **Add Boot Image**.
3. Browse to the WIM image to be imported—for example, D:\Production Deployment Share\Boot\LiteTouchPE_x86.wim.
4. 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.
5. 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. Click **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then click **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, click **Properties**.
4. Click 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$**.
6. Click **Edit Bootstrap.ini**.
7. Configure BootStrap.ini to use the **%WDSServer%** property by adding or changing the **DeployRoot** value to **DeployRoot=\\%WDSServer%\Deployment$**.
8. On the **File** menu, click **Save** to save the changes to the BootStrap.ini file.
9. Click **OK**.

   The deployment share needs to be updated.

10. In the details pane, click ***deployment_share*** (where *deployment_share* is the name of the deployment share to configure).
11. In the Actions pane, click **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 click **Next**.
13. On the **Summary** page, verify the details are correct, and then click **Next**.
14. On the **Confirmation** page, click **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. Click **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then click **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, click **Import Operating System**.

   The New OS Wizard starts.
5. On the **OS Type** page, click **Windows Deployment Services images**, and then click **Next**.
6. On the **WDS Server** page, type the name of the Windows Deployment Services server to be referenced—for example, **WDSSvr001**—and then click **Next**.
7. On the **Summary** page, verify the settings are correct, and then click **Next**.
8. On the **Confirmation** page, click **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. Click **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then click **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, click **Properties**.
4. On the **General** tab, select the **Enable multicast for this deployment share (requires Windows Server 2008 Windows Deployment Services)** check box.
5. Click **OK**.
6. In the Actions pane, click **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 click **Next**.
8. On the **Summary** page, verify the details are correct, and then click **Next**.
9. On the **Confirmation** page, click **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

1. Click **Start**, point to **Administrative Tools**, and then click **Windows Deployment Services**.
2. In the Windows Deployment Services console tree, right-click **Servers**, and then click **Add Server**.
3. In the **Add Servers(s)** dialog box, click **Local computer**, and then click **OK**.
4. In the Windows Deployment Services console tree, click **Servers**, then click ***server_name*** (where *server_name* is the name of the computer running Windows Deployment Services). Click **Multicast Transmissions**.
5. In the details pane, a new Auto-Cast transmission for the deployment share will be listed—for example, **BDD Share Deployment$**.
6. Verify that the status of the **BDD Share Deployment$** 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, click 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.

- `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.

- `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.

- `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.

- `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.

```
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.

- `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.

```
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.

- `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.

```
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.

- `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.

```
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.

- `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.

```
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.

- `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.

- `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.

- `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.

- `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.

- `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.

- `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.

- `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.

- `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.

```
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. Click **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then click **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, click **Properties**.
4. On the **OS Info** tab, click **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 click **Delete**.
6. Click **Yes**.
7. Save the answer file, and then exit Windows SIM.
8. Click **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

1. Click **Start**, and then point to **All Programs**. Point to **Microsoft Deployment Toolkit**, and then click **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, click **Properties**.
4. On the **Task Sequence** tab, go to and expand the State Restore node.
5. Verify that the **Recover From Domain** task sequence step is present. If yes, proceed to step 9.
6. In the task sequence **Properties** dialog box, click **Add**, go to **Settings**, and click **Recover From Domain**.
7. 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.
8. Verify that the settings for the **Recover From Domain** task sequence step are configured to meet your needs.
9. Click **OK** on the task sequence **Properties** dialog box to save the task sequence.