IBM ServerGuide Scripting Toolkit, Windows Edition

Altiris Deployment Solution User's Reference

Version 9.51

IBM ServerGuide Scripting Toolkit, Windows Edition

Altiris Deployment Solution User's Reference

Version 9.51

Note:
Before using this information and the product it supports, read the information in "Notices".

Contents

Chapter 1. Introducing ServerGuide Scripting Toolkit

The IBM ServerGuide Scripting Toolkit enables you to tailor and build custom hardware deployment solutions. It provides hardware configuration utilities and operating system (OS) installation examples for IBM® System x® and BladeCenter® x86-based hardware. The ServerGuide Scripting Toolkit, Windows Edition enables you to create a bootable Windows Preinstallation Environment (Windows PE) 2.1 CD, DVD, or USB key that supports the following:

- Network and mass storage devices
- Policy based RAID configuration
- Configuration of system settings using the Advanced Settings Utility (ASU)
- Configuration of Fibre Host Bus Adapters (HBAs) under WinPE
- Automated Network Operating System (NOS) installation support for:
  - Microsoft Windows Server 2008, Standard, Enterprise, Datacenter, and Web Editions
  - Microsoft Windows Server 2008 x64, Standard, Enterprise, Datacenter, and Web Editions
  - Microsoft Windows Server 2008, Standard, Enterprise, and Datacenter Editions without Hyper-V
  - Microsoft Windows Server 2008 x64, Standard, Enterprise, and Datacenter Editions without Hyper-V
  - Microsoft Windows Server 2008 R2 x64, Standard, Enterprise, Datacenter, and Web Editions
  - Microsoft Windows Server 2012
  - Microsoft Windows Server 2012 R2
- Local self-contained DVD deployment scenarios
- Local CD/DVD and network share based deployment scenarios
- RSA II, IMM, and BladeCenter MM/AMM remote disk scenarios
- UpdateXpress System Packs installation integrated with scripted NOS deployment
- IBM Director Agent installation integrated with scripted NOS deployment. The ServerGuide Scripting Toolkit, Windows Edition supports these versions of the Director Agent:
  - Director Agent 5.1 or higher
  - Common Agent 6.1 or higher
  - Core Services 5.20.31 or higher

The Toolkit provides sample jobs for Altiris Deployment Solution for hardware configuration and OS deployment using Windows PE. The Scripting Toolkit supports Altiris Deployment Solution 6.9, SP1, SP2, SP3, SP4 or SP5 and the Hewlett-Packard Rapid Deployment Pack (RDP) versions 3.81, 3.82, 3.83, 6.0, and 6.2. The RDP includes Altiris Deployment Solution, the Hewlett-Packard Scripting Toolkit, andWindows PE 2.1 boot environments packaged together.

Chapter 2. Installing the ServerGuide Scripting Toolkit

You can install the Toolkit into the Altiris Deployment Solution by using either a pre-built Windows PE 2.1 environment or by including the files required to build Windows PE 2.1. The prerequisites and installation depend on the method you select.

The ServerGuide Scripting Toolkit, Windows Edition for Altiris Deployment Solution has three components:

1. The IBM Pre-boot environment
2. The IBM Scripting Toolkit Scripts and Utilities
3. The IBM Scripting Toolkit Altiris jobs within the Altiris Deployment Solution console

When you install or reinstall the ServerGuide Scripting Toolkit, Windows Edition for Altiris Deployment Solution, these components are updated as a group. After the update, the Altiris Deployment Solution jobs associated with the previous version of the ServerGuide Scripting Toolkit, Windows Edition for Altiris Deployment Solution will no longer work.

Installing with pre-built Windows PE

This section describes the requirements and method for installing Toolkit using a pre-built Windows PE 2.1 environment.

Prerequisites

To install the ServerGuide Scripting Toolkit, Windows Edition by using a pre-built Windows PE 2.1 environment, you will need the following files:

- Altiris Deployment Solution, obtained via one of the following:
  - Altiris Deployment Solution 6.9, SP1, SP2, SP3, SP4 or SP5 with the Altiris provided Windows PE 2.1 environments already installed, with the following executable files:
    - `Altiris_DS_Preboot_WinPE2.1_x86.exe`
    - `Altiris_DS_Preboot_WinPE2.1_x64.exe` - required only for deployment of x64 operating systems
  - Hewlett Packard Rapid Deployment Pack 3.81 or higher
- Toolkit, extracted to the Deployment Server folder. The default location of the folder is `C:\Program Files\Altiris\eXpress\Deployment Server\`.

Installation

To integrate the Toolkit into the Altiris Deployment Solution, follow these steps:

1. If you want to suppress prompts or suppress the display of user settings before installing the Scripting Toolkit into Altiris Deployment Solution, open the Altiris_SGTKWinPE.ini file and modify the appropriate values shown in the table below.

Table 1. Build settings

Value	Description
Section: [SGTKWinPE Build Settings]	
TK_Build_SuppressPrompts	Suppress interactive prompts that pause the process to enable the user settings to be read after they are displayed. Valid values are: • Yes • No **Note:** Interactive prompts regarding required removal of jobs during upgrade or reinstallion scenarios are always displayed.
TK_Build_DisplayBuildSettings	Display the user settings before installing the Toolkit into Altiris Deployment Solution. Valid values are: • Yes • No

2. Open a command window and change the current directory to the directory containing the `Altiris_SGTKWinPE.cmd` file. The default file location is `C:\Program Files\Altiris\eXpress\Deployment Server\sgdeploy\SGTKWinPE\Altiris\Altiris_SGTKWinPE.cmd`.
3. At the command window enter the **`Altiris_SGTKWinPE.cmd`** command. It will automatically process the Toolkit settings in the `Altiris_SGTKWinPE.ini` file.

   Note: When you issue the **`Altiris_SGTKWinPE.cmd`**, you will receive a security warning from Windows that the publisher could not be verified for the**`SetRamDiskSize.cmd`** script. This script is part of the Toolkit installation package, and you can safely run it.

   To avoid receiving this warning multiple times during the installation, clear the **Always ask before opening this file** check box.

Depending upon the server setup, this process can take between 5 and 20 minutes to complete. Once completed, the Scripting Toolkit is integrated into Altiris Deployment Solution.

After the Scripting Toolkit is installed, there will be a new folder with the product name, version, and buildID under the Jobs frame in the Altiris Deployment Solution console. For example: **IBM ServerGuide Toolkit, Windows Edition 2.1 (yyyy-mm-dd)**

Installing with Windows PE 2.1 build files

This section describes the process for installing the IBM ServerGuide Scripting Toolkit with Altiris Deployment Solution using the files required to build theWindows PE 2.1 environment.

Prerequisites

Installing the Toolkit into Altiris Deployment Solution and building the Windows PE 2.1 environment requires the following:

- Altiris Deployment Solution 6.9, SP1, SP2, SP3, SP4 or SP5.

- IBM ServerGuide Scripting Toolkit, Windows Edition extracted to the Deployment Server folder. The default location of the folder is `C:\Program Files\Altiris\eXpress\Deployment Server\`.
- The English version of the Windows Automated Installation Kit (WAIK) for Windows Vista SP1 and Windows Server 2008. The default location for WAIK is `%ProgramFiles%\Windows AIK`, normally `C:\Program Files\Windows AIK`. If you have installed the WAIK to a different location, you must modify the *TK_Path_WAIK_Source* variable to point to the new location.

Installation

To install the IBM ServerGuide Scripting Toolkit, follow these steps:

1. Open the Altiris_SGTKWinPE.ini file and modify the appropriate values shown in the table below. The default file location is `C:\Program Files\Altiris\eXpress\Deployment Server\sgdeploy\SGTKWinPE\Altiris\Altiris_SGTKWinPE.ini`.

Value	Description
[SGTKWinPE Build Settings]	
TK_Build_SuppressPrompts	Suppress interactive prompts that pause the process to enable the user settings to be read after they are displayed. Default Value: **No** **Note:** Interactive prompts regarding required removal of jobs during upgrade or reinstallation scenarios are always displayed.
TK_Build_DisplayBuildSettings	Display the user settings before installing the Toolkit into Altiris Deployment Solution. Default Value: **Yes**
TK_Altiris_ShareLetter	Specifies the drive letter to use for the Altiris share in Windows PE. Valid values are the letters **F** through **Z**. Default: **Y**
[SGTKWinPE Paths]	
TK_Path_WAIK_Source	This variable has been deprecated. The AIK information is found automatically in the registry. You can uncomment this variable and use it to bypass the registry check, but it is not required. Default (if used): %ProgramFiles%\Windows AIK

2. Open a command window and change the current directory to the directory containing the `Altiris_SGTKWinPE.cmd` file. The default location for the file is `C:\Program Files\Altiris\eXpress\Deployment Server\sgdeploy\SGTKWinPE\Altiris\Altiris_SGTKWinPE.cmd`.
3. At the command window, enter the **`Altiris_SGTKWinPE.cmd`** command. It will automatically process the Toolkit settings in the `Altiris_SGTKWinPE.ini` file.

   Note: When you issue the **`Altiris_SGTKWinPE.cmd`**, you will receive a security warning from Windows that the publisher could not be verified for

the**SetRamDiskSize.cmd** script. This script is part of the Toolkit installation package, and you can safely run it.

To avoid receiving this warning multiple times during the installation, clear the **Always ask before opening this file** check box.

The process can take between 5 and 20 minutes to complete depending on the server setup. Once completed, the Scripting Toolkit is integrated into Altiris Deployment Solution.

After the Scripting Toolkit is installed, there will be a new folder with the product name, version, and buildID under the Jobs frame in the Altiris Deployment Solutionconsole. For example: **IBM Scripting Toolkit, Windows Edition 9.51**

Reinstalling the ServerGuide Scripting Toolkit, Windows Edition

This section describes the process for reinstalling the Toolkit.

The ServerGuide Scripting Toolkit, Windows Edition for Altiris Deployment Solution has three components:

1. The IBM Pre-boot environment
2. The IBM Scripting Toolkit Scripts and Utilities
3. The IBM Scripting Toolkit Altiris jobs within the Altiris Deployment Solution console

When you install or reinstall the ServerGuide Scripting Toolkit, Windows Edition for Altiris Deployment Solution, these components are updated as a group. After the update, the Altiris Deployment Solution jobs associated with the previous version of the ServerGuide Scripting Toolkit, Windows Edition for Altiris Deployment Solution will no longer work.

To reinstall the Toolkit, follow these steps:

1. Follow the instructions from Chapter 2, “Installing the ServerGuide Scripting Toolkit,” on page 3 that apply to your method of installation.
2. The installer displays the following message to indicate that a previous installation has been detected:

```
The following installed items were detected:

IBM Windows PE Toolkit, Windows Edition 2.1

This must be uninstalled before this process can continue. Please select
an option below:

1) Exit The Reinstaller.
2) Automatically Reinstall The Scripting Toolkit, Windows Edition.

Select an above option:
```

Note:

- To exit the current installation process, select option 1. You can manually remove the previous installation of the toolkit, and then install the new toolkit.
- To automatically remove your previous installation of the toolkit and reinstall the new toolkit, select option 2.

Manually removing the Toolkit

To manually remove the Toolkit, follow these steps:

1. Select option number 1 from the prompt to exit the installation process.
2. Remove all jobs associated with the IBM WinPE Toolkit PXE image in the Altiris Deployment Solution console.
3. Open the PXE configuration utility from the Tools menu within the Altiris Deployment Solution Console.
4. Select the image named `IBM WinPE Toolkit 2.1`.
5. Make certain that the column labeled **In Use By DS** does not say **Yes** for that image. If it does, there are additional jobs associated with that image. Find and remove them.
6. Click the **Delete** button to remove the PXE image.
7. Click **Save** to save the changes.
8. Click **OK** to exit the PXE Configuration utility.

Automatically removing the Toolkit

To automatically remove the Toolkit, follow these steps:

1. Select option 2 from the prompt. You will receive the following prompt:

```
*******************************************************************************
You have chosen to reinstall the following:

IBM Windows PE Toolkit, Windows Edtition 2.1

You may keep the jobs associated with the currently installed
Scripting Toolkit, however, they will be moved to a folder labeled
IBM Scripting Tooklit, Windows Edition (Archived w/ver. 2.1).
*******************************************************************************

Would you like to keep those jobs?<Y/N>
```

2. Select **Y** to remove the current Windows PE Image and move the current IBM ServerGuide Toolkit, Windows Edition jobs to a folder named IBM ServerGuide Toolkit, Windows Edition (Archive). Select **N** to remove all previously installed jobs, with the exception of user-created jobs.

Add files to the source tree

This section provides information about adding files to the source tree. You must add the files to be included in your deployments to the ServerGuide Scripting Toolkit, Windows Edition source tree.

The ServerGuide Scripting Toolkit, Windows Edition provides a Graphical User Interface (GUI) configuration program to add IBM Director Agent files to the source tree. The Toolkit Configuration Utility (`TKConfig.exe`) is located in the `sgdeploy\tkconfig` directory. You can start `TKConfig.exe` either from a command prompt or by double-clicking it.

Adding Windows installation files

Follow these steps to add Windows installation files to the source tree.

About this task

If you intend to use the source server as an OS repository for network deployments or create an OS deployment image bundled with Windows installation files for local deployments, follow these steps to add Windows installation files to the source tree.

Procedure

1. Start the Toolkit Configuration Utility.
2. Select **Add Operating System Installation Files** from the task list.
3. Follow the GUI Wizard for the operating system type you want.
4. Insert the correct OS installation media into the optical drive of the source system running Windows, or select the specific directory containing the OS installation media.
5. Modify the target path if necessary.
6. Copy the files from the source location to the target location.
7. Exit the Operating System Installation Files wizard.

Adding Windows device drivers

The Toolkit allows you to download System Enablement Packs (SEPs), which include a driver library that contains all of the drivers necessary to complete the installation of Windows Server 2008 and Windows Server 2012.

To ensure that all devices are installed, the drivers are up to date, and no errors remain in Device Manager, you must deploy the latest UpdateXpress System Packs. You can add support for systems released after the current version of the ServerGuide Scripting Toolkit, Windows Edition by downloading the applicable UpdateXpress System Packs. For more information, see Chapter 5, “System Enablement Packs,” on page 35.

Adding UpdateXpress System Packs

Follow these steps to add IBM UpdateXpress System Packs (UXSPs)to the source tree.

Procedure

Download the UpdateXpress System Pack for the desired machine-type and operating system combination into the source tree in the `updates\uxsp` directory. When downloading multiple UXSPs for multiple machine types, place them all in this directory. When the UXSP installer runs, it automatically selects the appropriate files. If you are prompted to overwrite existing files, click **OK** .

Note: Do not change the file name of any UXSP files, including changing the case of upper- and lowercase letters. Changing the file name or case can cause the UXSP deployment to fail.

Results

The UXSP will be installed after the operating system is installed. .

Adding IBM Director Agent files

To install IBM Systems Director Agent during Windows OS deployment, follow these steps to add the installation files to the source tree.

Procedure

1. Download the IBM Systems Director Agent files from http://www.ibm.com/systems/management/director/downloads/

   Note: This download requires registration with ibm.com.
2. Unpack the Director Agent files to a convenient location.
3. Start the Toolkit Configuration Utility.
4. Select **Add Operating System Application Files** from the task list.
5. Browse to the location where you unpacked the Director Agentfiles.
6. Copy the files from the source location to the target location.
7. Exit the Operating System Application files wizard.

Customizing the Windows answer file

The Windows answer file provides responses to prompts encountered during installation, allowing you to perform unattended installations. Four sample answer files are provided with Toolkit, and another is provided by Microsoft.

The Toolkit sample answer files, `win2008.xml`, `win2008x64.xml`, `win2011x64.xml`, and `win2012x64.xml` are located in the `Program Files\Altiris\eXpress\Deployment Server\sgdeploy\SGTKWinPE\Altiris\AnswerFiles` directory.

Customizing the Windows Server 2008 answer file

To customize the Windows Server 2008 answer file:

1. Open the file, `sgdeploy\SGTKWinPE\Altiris\AnswerFiles\win2008x64.xml`.
2. Add the settings you want to customize. In this example:

   ```
   <TimeZone>%TK_TimeZone%</TimeZone>
   ```
3. Set the value of the setting you have added to a variable that you will add to the appropriate Scripting Toolkit scenario INI file. Scripting Toolkit environment variables are surrounded by the percent sign (%), as shown in this example:

   ```
   [GuiUnattended]
   ...
   <TimeZone>%TK_TimeZone%</TimeZone>
   ...
   ```

   Typically you will add this value to the `[NOS Installation Settings]` section of the INI file.
4. For each environment variable you have assigned, include a value in the corresponding scenario INI file. Using the format *variable_name=value*, where *variable_name* is the name you selected in the answer file and *value* is a valid value for the variable you are using. In this example:

   ```
   TK_TimeZone=035
   ```

   or:

   ```
   TK_TimeZone=Pacific Standard Time
   ```

   Typically you will add this value to the `[NOS Installation Settings]` section of the INI file.

During deployment, the environment variables specified in the answer file are replaced with the corresponding variables from the scenario INI file.

Customizing the Windows Server 2012 answer file

To customize the Windows 2012 answer file:

1. Open the file, `sgdeploy\SGTKWinPE\Altiris\AnswerFiles\win2012x64.xml`.
2. Add the settings you want to customize. In this example:

   ```
   <TimeZone>%TK_TimeZone%</TimeZone>
   ```

3. Set the value of the setting you have added to a variable that you will add to the appropriate Scripting Toolkit scenario INI file. Scripting Toolkit environment variables are surrounded by the % sign, as shown in this example:

   ```
   [GuiUnattended]
   ...
   <TimeZone>%TK_TimeZone%</TimeZone>
   ...
   ```

   Typically you will add this value to the `[NOS Installation Settings]` section of the INI file.

4. For each environment variable you have assigned, include a value in the corresponding scenario INI file. Using the format *variable_name=value*, where *variable_name* is the name you selected in the answer file and *value* is a valid value for the variable you are using. In this example:

   ```
   TK_TimeZone=035
   ```

   or:

   ```
   TK_TimeZone=Pacific Standard Time
   ```

   Typically you will add this value to the `[NOS Installation Settings]` section of the INI file.

Chapter 3. Quick start deployment scenarios

This section contains basic information about deployment scenarios to help you to begin using the Toolkit with Altiris Deployment Solution as quickly as possible.

The Toolkit provides a collection of sample jobs for use by the Altiris Deployment Solution. The following topics provide additional information on the most commonly used jobs. For more information about the sample jobs provided by theScripting Toolkit, see "Sample job definitions" on page 26

The Toolkit for use with Altiris Deployment Solution provides two types of jobs:

- "Modular deployment tools"
- "Deployment solutions" on page 16

For more information about tailoring deployments to your needs, see Chapter 4, "Customizing deployment scenarios," on page 19.

Modular deployment tools

The sample jobs provided in the **Modular Deployment Tools** folder are designed to be used individually to complete common deployment tasks. This section describes how and in what order to use the modular deployment tools.

The modular deployment tools provided by the Toolkit are each designed to complete one step in the deployment process. The following sections provide information about using the sample jobs for:

- Configuring hardware before installation
- Installing the operating system
- Configuring the system after installation

Validating the Altiris environment

After installing the Toolkit for use with Altiris Deployment Solution, you should validate that the environment is operating properly. This topic describes the process for doing so, using the sample job provided.

Before you begin

Complete the installation method appropriate for your environment as described in Chapter 2, "Installing the ServerGuide Scripting Toolkit," on page 3.

About this task

When the installation is complete, validate it by performing the following steps:

Procedure

1. Open the Altiris Deployment Solution console.
2. Run the job **Windows PE Boot Test**, located in the Pre-Boot OS Connectivity Test folder, in the IBM ServerGuide Toolkit, Windows Edition 2.1 folder. The job boots into the Windows PE x86 environment on the target server. After

connectivity is established, the job performs a **dir** command on the eXpress share of the Altiris Server and then pauses, requiring your input to continue.

3. Provide the requested input to complete the job.

Preinstallation tasks

This section provides information about the tasks available before you install an operating system using the modular tools.

Before installing an operating system, you can use the modular tools to perform the following tasks:

- Capture and deploy system settings
- Configure RAID
- Replicate RAID
- Perform a server disk analysis

The following sections describe how to perform these tasks.

Capture and deploy system settings

Before installing an operating system, you can capture system settings from a compatible server to a file and deploy them to your new server by using the Toolkit jobs and the Advanced Settings Utility (ASU).

The Toolkit provides sample jobs to capture and deploy settings using ASU. These jobs are located in the **IBM Scripting Toolkit, Windows Edition 9.51** > **Modular Deployment Tools** > **Step 1 - Preinstallation Hardware Configuration** > **System Settings Replication** folder.

Capture system settings

The **Capture System Settings** sample job captures the system settings of the server and stores them in a file named `ASU_Capture.ini`. By default all captured system settings are stored in `C:\Program Files\Altiris\eXpress\Deployment Server\sgdeploy\SGTKWinPE\ASUFiles\`*`Machine_Type`* where *Machine_Type* is the machine type of the captured system.

Deploy system settings

The **Deploy System Settings** sample job deploys the system settings from the `ASU_Capture.ini` file located in the specified directory.

You can customize the following variables for these jobs from the Altiris Console. The values shown are the defaults.

- TK_ASU_File=ASU_Capture.ini
- TK_ASU_FileLocation="*TK_Altiris_Path*\ASUFiles*Machine_Type*"
- TK_ASU_Mode=save
- TK_ASU_Flags= –group-bios

Note: The *TK_ASU_File* variable should not include any path information.

RAID configuration

Toolkit provides sample jobs to perform a number of RAID configurations. These jobs are located in the **IBM Scripting Toolkit, Windows Edition 9.51** > **Modular Deployment Tools** > **Step 1 - Preinstallation Hardware Configuration** > **RAID**

Configuration folder, and are named according to the RAID configuration they will create. For example, **Configure RAID [RAID 0]**and **Configure RAID [RAID 1 + Hot Spare]**.

To perform RAID configuration using the modular tools:

1. Start the Altiris Deployment Solution console.
2. Navigate to the **IBM Scripting Toolkit, Windows Edition 9.51** > **Modular Deployment Tools** > **Step 1 - Preinstallation Hardware Configuration** > **RAID Configuration** folder.
3. Run the appropriate job against the target system to create the RAID configuration you want.

RAID replication

If you have an existing RAID configuration that you want to replicate to other systems, you can use the sample jobs **RAID Capture** and **RAID configuration**, located in the **IBM Scripting Toolkit, Windows Edition 9.51** > **Modular Deployment Tools** > **Step 1 - Preinstallation Hardware Configuration** > **RAID Replication** folder in the Altiris Deployment Solution console.

To capture an existing RAID configuration, follow these steps:

1. Start the Altiris Deployment Solution console.
2. Navigate to the **IBM Scripting Toolkit, Windows Edition 9.51** > **Modular Deployment Tools** > **Step 1 - Preinstallation Hardware Configuration** > **RAID Replication** folder.
3. Run the **Capture RAID configuration** job against the system whose RAID configuration you want to replicate.

   This job will create a RAID configuration file and store it on the source server. By default, it uses the following values:

   RAID configuration filename
   RAID_Configuration.ini

   Location
   `C:\Program Files\Altiris\eXpress\Deployment Server\sgdeploy\SGTKWinPE\PolicyFiles\`*`machinetype`*, where *machinetype* is the machine type of the captured system.

When you have captured the RAID configuration, you can deploy it using the **Deploy RAID configuration** job in the **IBM Scripting Toolkit, Windows Edition 9.51** > **Modular Deployment Tools** > **Step 1 - Preinstallation Hardware Configuration** > **RAID Replication** folder.

To deploy the captured RAID configuration:

1. Start the Altiris Deployment Solution console.
2. Navigate to the **IBM Scripting Toolkit, Windows Edition 9.51** > **Modular Deployment Tools** > **Step 1 - Preinstallation Hardware Configuration** > **RAID Replication** folder.
3. Run the **Deploy RAID configuration** job against the target system.

   By default, this job uses the same default file name and location as the **Capture RAID configuration** job. See “Customizing sample jobs” on page 19 for information about customizing the file name and location.

Server disk analysis

The ServerGuide Scripting Toolkit provides a sample job to capture the disk information for all disks attached to a given server. You can use this job after performing RAID configuration to ensure that when you install a supported Windows operating system, you install to disk 0 on the target server.

Note: This job must be run after RAID configuration is complete, because disk information is changed by the RAID configuration task.

The job is located in: **IBM Scripting Toolkit, Windows Edition 9.51** > **Modular Deployment Tools** > **Step 1 - Preinstallation Hardware Configuration** > **Server Disk Analysis** in the Altiris Deployment Solutions console.

To perform server disk analysis, follow these steps:

1. Start the Altiris Deployment Solution console.
2. Navigate to the **IBM Scripting Toolkit, Windows Edition 9.51** > **Modular Deployment Tools** > **Step 1 - Preinstallation Hardware Configuration** > **Server Disk Analysis** folder.
3. Run the job **Capture Disk Data** against the system for which you want to capture disk information.

The job creates a disk information file and stores it on the source server at: `C:\Program Files\Altiris\eXpress\Deployment Server\temp\ID\disk_details.txt`.

Operating system installation

After you have configured RAID on the target server, you can use the jobs provided by the Toolkit to install a supported Windows operating system. This section describes that process.

The ServerGuide Scripting Toolkit, Windows Edition provides two types of jobs for deploying an operating system:

- Operating system cloning
- Scripted operating system installation

The following sections describe how to perform each type of installation.

Operating system cloning

The ServerGuide Scripting Toolkit, Windows Edition provides sample jobs to capture and deploy a supported Windows operating system. To run the capture Windows image sample job, the Altiris Agent must first be installed on the donor system. The scripted operating system installation jobs provided by the Toolkit install the Altiris Agentautomatically. If you are cloning an operating system that was not installed through the Toolkit, you must install the Altiris Agent manually.

To capture a supported operating system from one server and deploy it to another, follow these steps:

1. Start the Altiris Deployment Solution console, and navigate to the Operating System Imaging folder at the following folder:**IBM ServerGuide Toolkit, Windows Edition 2.1** > **Modular Deployment Tools** > **Step 2 - Operating System Installation** > **Operating System Imaging**.
2. Open the **Capture Windows Image** job.
3. Select the **Create Disk Image** task, and click **Modify**.

4. Change the path and file name for the captured image, and click **Finish** to save your changes.
5. Open the **Deploy Windows Image** job.
6. Select the **Distribute Disk Image** task, and click **Modify**.
7. Change the path and file name to match the one that you entered for the captured image, and click **Finish** to save your changes.
8. Run the **Capture Windows Image** job against the donor server.
9. When the job completes, run the **Deploy Windows Image** job against the target server.

Scripted operating system installation

If you do not want to deploy a clone of an existing operating system installation, the Toolkit provides sample jobs to deploy each of the supported operating systems. Use the following procedure to install a supported Windows operating system by using the sample jobs provided:

1. Ensure that you have a properly configured Toolkit source server with the required operating system files. See "Adding Windows installation files" on page 7 for more information.
2. Open the Altiris Deployment Solution console and ensure that the name of the system in the Console is the name you want for the computer name of that system after your deployment. If this is not the case, add the appropriate computer name to the answer file.
3. Navigate to the **IBM Scripting Toolkit, Windows Edition 9.51** > **Modular Deployment Tools** > **Step 2 - Operating System Installation** > **Scripted Operating System Installation** folder.
4. Select the job that corresponds to the Windows operating system you want to install, and run it against the target server.

Post installation tasks

When you have completed the RAID configuration and operating system installation, you can perform post installation tasks. This section describes the methods for installing the IBM Director Agent and applying IBM UpdateXpress System Pack (UXSP) updates.

The Toolkit provides sample jobs for installing the IBM Director Agent and applying IBM UpdateXpress System Pack (UXSP) updates to a target system on which RAID has been configured and a supported Windows operating system installed. Note that in order to complete these jobs, Altiris Agent must be installed on the target system. Altiris Agent is installed automatically by the operating system installation tasks provided by the Toolkit. If the operating system on the target system was not installed using these jobs, you might have to install Altiris Agent manually. See "Operating system installation" on page 14 for more information on using the modular deployment tools to install operating systems.

Note: Installing IBM Director Agent and UpdateXpress System Pack updates on the client system requires Administrator privileges. To successfully run the Altiris post installation tasks for IBM Director Agent and UpdateXpress System Pack updates, you must change the security context of the task by adding the password for the Administrator account on the client system. See "Changing the security context of an Altiris task" on page 100 for more information.

The following sections describe the process for using the modular deployment tools to install the IBM Director Agent and apply UXSP updates to a target system.

Installing the IBM Director Agent

Follow these steps to install the IBM Director Agent on a target system:

1. Ensure that the IBM Director Agent source files have been added to the source tree on the source server. For more information, see "Adding IBM Director Agent files" on page 8.
2. Open the Altiris Deployment Solution console and navigate to the **IBM ServerGuide Toolkit, Windows Edition 9.51** > **Modular Deployment Tools** > **Step 3 - Post-Installation Configuration** > **Application Installation** folder.
3. Run the **IBM Director Agent for Windows Install** job against the target server.

   Note: If the IBM Director Agent files are not located in the default directory in the source tree, you must modify the TK_DirAgent_DirectorAgent job variable to point to the correct location. For more information on modifying job variables, see "Customizing sample jobs" on page 19.

Installing UpdateXpress System Pack updates

Follow these steps to install UXSP updates on the target system:

1. Ensure that the UpdateXpress System Pack source files have been added to the source tree on the source server. For more information, see "Adding UpdateXpress System Packs" on page 8.
2. Open the Altiris Deployment Solution console and navigate to the **IBM Scripting Toolkit, Windows Edition 9.51** > **Modular Deployment Tools** > **Step 3 - Post-Installation Configuration** > **IBM UpdateXpress System Packs** folder.
3. Run the **Run UpdateXpress System Pack Installer** job against the target server.

Deployment solutions

The sample jobs provided in the **Combined Deployment Solutions** folder are designed to consolidate multiple deployment elements in a single Altiris job. This section describes how to use these jobs to perform complete deployments.

The deployment solutions provided by the Toolkit are designed to perform a combination of deployment steps in a single job. The following sections provide information about using the sample jobs for:

- Image based deployment (image cloning)
- Scripted deployment

Image based deployment

The Toolkit provides a set of tools to capture and deploy an existing system configuration, including RAID configuration, operating system, and installed applications and updates. These jobs require Altiris Agent to be installed on the system that is being captured. Altiris Agent is installed automatically as part of the Scripting Toolkit operating system installation jobs. If the donor system operating system was not installed through the Scripting Toolkit, you might need to install Altiris Agent manually.

To capture and deploy a system configuration, follow these steps:

1. Open the Altiris Deployment Solution console and navigate to the **IBM ServerGuide Toolkit, Windows Edition 9.51** > **Combined Deployment Solutions** > **Image-based Deployment** folder.
2. Run the **Capture RAID Configuration and Windows Image** job against the donor system.
3. When that job completes, run the **Deploy RAID Configuration and Windows Image** job against the target system.

Scripted deployment

The scripted deployment sample jobs provided by the ServerGuide Scripting Toolkit are designed to integrate all of the steps for deploying a system configuration in a single Altiris job.

To use a scripted deployment job to configure RAID, install an operating system, install the IBM Director Agent, and install UpdateXpress System Pack updates on the target system, follow these steps:

1. Ensure that the necessary operating system, Director Agent, and UXSP files are properly included in the source tree. See "Add files to the source tree" on page 7 for more information.
2. Open the Altiris Deployment Solution console and navigate to the **IBM ServerGuide Toolkit, Windows Edition 9.51** > **Combined Deployment Solutions** > **Scripted Deployment** folder.
3. From this folder, select the job to deploy the operating system of your choice. Modify the following job variables:

   TK_NOS_PerformDirectorAgentInstallation="Yes"
   To install the IBM Director Agent.

   TK_NOS_PerformPostOSInstallUXSPUpdates="Yes"
   To install UpdateXpress System Pack updates.

   See "Customizing sample jobs" on page 19 for more information on configuring job variables.
4. Run the job against the target server.

Chapter 4. Customizing deployment scenarios

This section provides information about customizing deployment scenarios.

You can customize your deployment scenarios in the following ways:

- Customize source server settings
- Add PRAID policy files
- Add ASU files
- Customize Fibre HBA boot configuration
- Customize your Windows installation
- Add installation of the IBM Director Agent to your deployment
- Add installation of UpdateXpress System Packs to your deployment
- Modify the Windows PE image
- Add custom scripts to the SGTKWinPE process
- Add files to theWindows PE image.
- Automate the deployment process

Customizing sample jobs

This section provides information on customizing the sample Altiris jobs provided with the ServerGuide Scripting Toolkit, Windows Edition.

Each sample job provided within the Altiris Deployment Console that contains customizable settings begins with a task that sets all variables for the job. This task is labeled **Customize Job Variables.** The variables that you can modify are contained within a block that is similar to the following:

```
[User_Customizable_Variables]

Variables

[End_User_Customizable_Variables]
```

You cannot modify variables outside this block. When you modify a job, it is a good practice to create a copy of the job and modify the copy, rather than modifying the original job.

Table 2 provides a list of the variables in the sample Altiris jobs that you can customize. Note that all values for variables must be enclosed within quotation marks.

Table 2. Customizable variables in Altiris sample jobs

Variable	Description
[PRAID Settings]	
TK_PRAID_PolicyFile	Specifies the PRAID policy file to be used for the job. This value should not contain any path information. Path information is specified in TK_Path_PolicyFiles, described below.

Table 2. Customizable variables in Altiris sample jobs (continued)

Variable	Description
TK_Path_PolicyFiles	Specifies the location in the Altiris shared directory of the policy file specified by the *TK_PRAID_PolicyFile* variable. If no value is specified, then the policy file must be in the Altiris shared directory. The default Altiris shared directory is "C:\Program Files\Altiris\eXpress\ Deployment Server". For convenience, the *TK_Altiris_Path* variable specifies the path to the Altiris directory within the Toolkit, by default: `C:\Program Files\Altiris\eXpress\Deployment Server\sgdeploy\SGTKWinPE\Altiris`. For example, if the policy file is in the `C:\Program Files\Altiris\eXpress\ Deployment Server\sgdeploy\SGTKWinPE\PolicyFiles` directory, you can specify the following: `%TK_Altiris_Path%\PolicyFiles`.**Default**: Varies per job, see next section.
[Fibre Settings]	
TK_FIBRE_COUNT	Specifies the number of HBA ports to configure. Valid values are 1–*n*, where *n* is the number of HBA ports available. This variable affects the use of the following variables: • TK_FIBRE_N_HBA_ID • TK_FIBRE_N_BOOT_DISABLE • TK_FIBRE_N_BOOT_PRIM • TK_FIBRE_N_BOOT_ALT1 • TK_FIBRE_N_BOOT_ALT2 • TK_FIBRE_N_BOOT_ALT3 Where *N* is the HBA number to be configured. **Note:** You must complete one of each of these variables for every HBA port you configure. So if TK_FIBRE_COUNT=2, you must complete one set of these variables for the first port and one for the second.

Table 2. Customizable variables in Altiris sample jobs (continued)

Variable	Description

TK_FIBRE_*N*_HBA_ID	Identifies the Qlogic/Emulex HBA to be configured, where *N* is the HBA number to be configured.
	
	Valid values are:
	
	hba_instance
	
	the instance number of an HBA port. Valid values are integers from 0 to *n*-1, where *n* is the number of HBAs in the system.
	
	For example, to configure HBA instance 0, you would use`TK_FIBRE_1_HBA_ID=0`.
	
	hba_wwpn
	
	the World Wide Port Name of an HBA port, in the format xx-xx-xx-xx-xx-xx-xx-xx or xxxxxxxxxxxxxxxx.
	
	For example, to configure HBA: 90-87-AA-BB-65-34-BB-E0:
	
	`TK_FIBRE_1_HBA_ID=`
	`90-87-AA-BB-65-34-BB-E0`
	
	Default: 0

	Identifies the Brocade HBA to be configured, where *N* is the HBA number to be configured.
	
	Valid values are:
	
	hba_instance
	
	the instance number of an HBA port. A valid format is *N/P*, where *N* is the adapter number from 1 to *N*, and *P* is the port number from 0 to p-1.
	
	For example: `TK_FIBRE_1_HBA_ID=1/0`.
	
	hba_wwpn
	
	the World Wide Port Name of an HBA port, in the format xx-xx-xx-xx-xx-xx-xx-xx or xxxxxxxxxxxxxxxx.
	
	For example, to configure HBA: 90-87-AA-BB-65-34-BB-E0:
	
	`TK_FIBRE_1_HBA_ID=`
	`90-87-AA-BB-65-34-BB-E0`
	
	Default: 0

Table 2. Customizable variables in Altiris sample jobs (continued)

Variable	Description
TK_FIBRE_*N*_BOOT_DISABLE	Disable the selected current boot device settings on the specified HBA port, where *N* is the HBA number to be configured. Valid values are **No** Does not clear or disable any boot settings. **All** Disables the primary and all alternate boot settings - Prim, Alt1, Alt2, and Alt3. **Prim** Disables only the primary boot setting. **Alt1** Disables the Alternative 1 boot setting. **Alt2** Disables the Alternative 2 boot setting. **Alt3** Disables the Alternative 3 boot setting. Default: No.

TK_FIBRE_*N*_BOOT_PRIM = *target_wwnn target_wwpn lun_id*	Defines the primary boot target settings, where *N* is the HBA number to be configured, and:
	• *target_wwnn* - is the World Wide Node Name of a device, in the format xx-xx-xx-xx-xx-xx-xx-xx or xxxxxxxxxxxxxx.
	• *target_wwpn* - is the World Wide Port Name of a device, in the format xx-xx-xx-xx-xx-xx-xx-xx or xxxxxxxxxxxxxx.
	• *lun_id* - is the Logical Unit Number of a device.
	
	Default: 0 0 0
	
	Example:
	`TK_FIBRE_1_BOOT_PRIM=`
	`BB-CC-AA-BB-65-34-BB-F1`
	`BB-CC-AA-BB-FF-34-BB-F1 9`

TK_FIBRE_*N*_BOOT_ALT1 = *target_wwnn target_wwpn lun_id*	Configures the operating system to use the indicated target as the first alternate boot device, where *N* is the HBA number to be configured, and
	• *target_wwnn* - is the World Wide Node Name of a device, in the format xx-xx-xx-xx-xx-xx-xx-xx or xxxxxxxxxxxxxx.
	• *target_wwpn* -is the World Wide Port Name of a device, in the format xx-xx-xx-xx-xx-xx-xx-xx or xxxxxxxxxxxxxx.
	• *lun_id* - is the Logical Unit Number of a device.
	
	Default: blank.
	
	Example:
	`TK_FIBRE_1_BOOT_ALT1=`
	`BB-CC-AA-BB-65-34-BB-FD`
	`BB-CC-AA-BB-FF-40-BB-F1 5`

Table 2. Customizable variables in Altiris sample jobs (continued)

Variable	Description

TK_FIBRE_*N*_BOOT_ALT2 = *target_wwnn target_wwpn lun_id*	Configures the operating system to use the indicated target as the second alternate boot device, where *N* is the HBA number to be configured, and
	• *target_wwnn* - is the World Wide Node Name of a device, in the format xx-xx-xx-xx-xx-xx-xx-xx or xxxxxxxxxxxxxx.
	• *target_wwpn* -is the World Wide Port Name of a device, in the format xx-xx-xx-xx-xx-xx-xx-xx or xxxxxxxxxxxxxx.
	• *lun_id* - is the Logical Unit Number of a device.
	Default: blank.
	Example:
	`TK_FIBRE_1_BOOT_ALT2=`
	`BB-CC-AA-BB-65-34-BB-FD`
	`BB-CC-AA-BB-FF-40-BB-F1 5`

TK_FIBRE_*N*_BOOT_ALT3 = *target_wwnn target_wwpn lun_id*	Configures the operating system to use the indicated target as the third alternate boot device, where *N* is the HBA number to be configured, and
	• *target_wwnn* - is the World Wide Node Name of a device, in the format xx-xx-xx-xx-xx-xx-xx-xx or xxxxxxxxxxxxxx.
	• *target_wwpn* -is the World Wide Port Name of a device, in the format xx-xx-xx-xx-xx-xx-xx-xx or xxxxxxxxxxxxxx.
	• *lun_id* - is the Logical Unit Number of a device.
	Default: blank
	Example:
	`TK_FIBRE_1_BOOT_ALT3=`
	`BB-CC-AA-BB-65-34-BB-FD BB-CC-AA-BB-FF-40-BB-F1 5`

[ASU Settings]	
TK_ASU_File	Specifies the file to be used for a Capture System Settings or Deploy System Settings operation. If you are capturing system settings, they will be written to a file with this name. If you are deploying system settings, they will be read from this file. The default value is **`ASU_Capture.ini`**
TK_Path_ASUFiles	Specifies the path to the file specified by **`TK_ASU_File`**. The default is `asufiles\`*`machine_type`*, where *machine_type* is the four-digit machine type of the target system.
TK_ASU_Mode	Designates the capture class. This variable is used only when capturing system settings. The default value is **`save`**.
TK_ASU_Flags	Specifies additional flags for the current mode. The default value is **`–group-bios`**.
[Partition Settings]	

Table 2. Customizable variables in Altiris sample jobs (continued)

Variable	Description
TK_Partition_DiskNum	Specifies the disk number on which to create new partition. Valid values are the disk numbers found by diskpart.exe. Default: AUTO **Note:** AUTO setting is the first disk on the system.
TK_Partition_Size	Specifies the partition size in MB. Valid values are: • "Max" - uses all available space • *"number"* - specifies the partition size **Default:** "Max"
TK_Partition_FileSystem	Specifies the file system type to use when formatting the drive. Valid values are: • "NTFS" • "FAT32" **Default:** "NTFS"
TK_Partition_SR_Size	Specifies the partition size, in MB, for a System Reserved Partition. The System Reserved Partition is a primary active partition created during the partitioning step. BitLocker Drive Encryption function requires this partition active partition and formatted ntfs. Valid values are integers greater than 108. **Default:** 108MB **Note:** This setting is supported only for Windows Server 2008 R2. This setting is ignored when booting WinPE in native uEFI mode. For more information on BitLocker Drive Encryption, see: http://technet.microsoft.com/en-us/library/cc731549%28WS.10%29.aspx
[NOS Installation Settings]	
TK_NOS_NetworkOperatingSystem	Specifies the NOS to use for the deployment. This must be a valid directory name within the sgdeploy\OS directory in the Toolkit Source Server.**Default:** Varies per job. For example, `Win2008_x64` is used for Windows 2008 Enterprise Edition (x64).
TK_NOS_AnswerFile	Specifies the answer file to use for the deployment. This must be a valid filename within the SGTKWinPE\AnswerFiles directory in the Toolkit Source Server.**Default:** `win2008.xml` for Windows 2008 x86 jobs and `win2008x64.xml`for Windows 2008 x64 jobs.
TK_NOS_ProductKey	Specifies the product key to be used for unattended installations of Windows operating systems.**Default:** blank.

Table 2. Customizable variables in Altiris sample jobs (continued)

Variable	Description
TK_NOS_DeploymentDriverLibrary	Specifies the Deployment Driver library to use for the deployment. Valid values are: • "Auto" - searches all directories within the sgdeploy\drvs directory and selects the newest DDL present that supports the machine and OS being deployed. • "*dirname*" - must be a valid directory name within the sgdeploy\drvs directory in the Toolkit Source Server. For example, "w23_drv" would be specified for sgdeploy\drvs\w23_drv. **Default:** "Auto"
TK_NOS_PerformDirectorAgentInstallation	Automatically install the IBM Director Agent. Valid values are: • "Yes" • "No" **Default:** "No"
TK_NOS_PerformPostOSInstallUXSPUpdates	Automatically installUpdateXpress System Packs after the OS is installed. Valid values are: • "Yes" • "No" **Default:** "No"
[Director Agent Settings]	
TK_DirAgent_DirectorAgent	Specifies the location of the IBM Director Agent application files on the source server. Valid values are: • dawin • A user-supplied directory in the `sgdeploy\apps` directory Default: dawin
[UXSP Settings]	
TK_UXSP_ApplyLatest	Specifies whether USXPI should apply latest updates to the target system if no UXSPs are found for the target system. Setting this variable to *Yes* will force the UpdateXpress System Pack Installer to apply the latest updates to the target system if no UXSPsare found for that system. Valid values: Yes, No Default: No
TK_UXSP_UXSPIUpdateFlags	Specifies user provided command line arguments for processing by the UpdateXpress System Pack Installer in Update mode. To provide command line arguments to be processed by UXSPi, set this variable to the command line arguments. See "UpdateXpress System Pack Installer" on page 43 for a list of command line arguments to use with UXSPi in Update mode. Default: blank

Table 2. Customizable variables in Altiris sample jobs (continued)

Variable	Description
TK_UXSP_UpdateXpressSystemPacks	Specifies the location of the UpdateXpress System Packs on the source server. Valid values are: • uxsp • a user-specified directory in the SGTKWinPE\updates directory Default: uxsp
TK_UXSP_ForceReboot	Specifies whether to reboot the system after executing the UpdateXpress System Pack Installer. Valid vlues are Yes and No. Default: No

Sample job definitions

This section provides descriptions of the sample Altiris jobs provided by the Toolkit, including a description of the variables that can be customized for each job.

All of the sample jobs, except imaging jobs without RAID configuration and the Windows PE boot test, add logging information to following location`C:\Program Files\Altiris\eXpress\Deployment Server\temp\ID\IBM_WinPEToolkit.log`.

Customizing Fibre HBA boot configuration

You can use Toolkit variables to customize the configuration of Fibre HBAs on the target system, allowing them to boot from SAN targets.

By default, the ServerGuide Scripting Toolkit will configure the first QLogic HBA on the system to boot from the first available SAN target for QLogic Fibre HBAs only. Emulex Fibre HBAs are not supported. For more information, see “Known problems and limitations” on page 102). The BIOS configures the first disk drive that it finds that is also a LUN 0 as a boot device. The ServerGuide Scripting Toolkit uses the variables in the following table to configure Fibre HBAs.

Note: In some examples that follow, single lines are broken into multiple lines for formatting reasons. When using these settings, you must present all of the information for each variable on a single line.

Table 3. Fibre HBA boot configuration variables

Variable	Description
TK_FIBRE_COUNT	Specifies the number of HBA ports to configure. Valid values are 1–*n*, where *n* is the number of HBA ports available. This variable affects the use of the following variables: • TK_FIBRE_N_HBA_ID • TK_FIBRE_N_BOOT_DISABLE • TK_FIBRE_N_BOOT_PRIM • TK_FIBRE_N_BOOT_ALT1 • TK_FIBRE_N_BOOT_ALT2 • TK_FIBRE_N_BOOT_ALT3 Where *N* is the HBA number to be configured. **Note:** You must complete one of each of these variables for every HBA port you configure. So if TK_FIBRE_COUNT=2, you must complete one set of these variables for the first port and one for the second.

Table 3. Fibre HBA boot configuration variables (continued)

Variable	Description

TK_FIBRE_*N*_HBA_ID	Identifies the Qlogic/Emulex HBA to be configured, where *N* is the HBA number to be configured.
	
	Valid values are:
	
	hba_instance
	
	the instance number of an HBA port. Valid values are integers from 0 to *n*-1, where *n* is the number of HBAs in the system.
	
	For example, to configure HBA instance 0: `TK_FIBRE_1_HBA_ID=0`.
	
	hba_wwpn
	
	the World Wide Port Name of an HBA port, in the format xx-xx-xx-xx-xx-xx-xx-xx or xxxxxxxxxxxxxxxx.
	
	For example, to configure HBA: 90-87-AA-BB-65-34-BB-E0:
	
	`TK_FIBRE_1_HBA_ID=`
	`90-87-AA-BB-65-34-BB-E0`
	
	Default: 0

	Identifies the Brocade HBA to be configured, where *N* is the HBA number to be configured.
	
	Valid values are:
	
	hba_instance
	
	the instance number of an HBA port. Valid format should be *N/P*, where *N* is the adapter number from 1 to N, and *P* is the port number from 0 to p-1.
	
	For example: `TK_FIBRE_1_HBA_ID=1/0`.
	
	hba_wwpn
	
	the World Wide Port Name of an HBA port, in the format xx-xx-xx-xx-xx-xx-xx-xx or xxxxxxxxxxxxxxxx.
	
	For example, to configure HBA: 90-87-AA-BB-65-34-BB-E0:
	
	`TK_FIBRE_1_HBA_ID=`
	`90-87-AA-BB-65-34-BB-E0`
	
	Default: 0

Table 3. Fibre HBA boot configuration variables (continued)

Variable	Description
TK_FIBRE_*N*_BOOT_DISABLE	Disable the selected current boot device settings on the specified HBA port, where *N* is the HBA number to be configured. Valid values are **No** Does not clear or disable any boot settings. **All** Disables the primary and all alternate boot settings: Prim, Alt1, Alt2, and Alt3. **Prim** Disables only the primary boot setting. **Alt1** Disables the Alternative 1 boot setting. **Alt2** Disables the Alternative 2 boot setting. **Alt3** Disables the Alternative 3 boot setting. Default: No

TK_FIBRE_*N*_BOOT_PRIM = *target_wwnn target_wwpn lun_id*	Defines the primary boot target settings, where *N* is the HBA number to be configured, and:
	• *target_wwnn* is the World Wide Node Name of a device, in the format xx-xx-xx-xx-xx-xx-xx-xx or xxxxxxxxxxxxxx.
	• *target_wwpn* is the World Wide Port Name of a device, in the format xx-xx-xx-xx-xx-xx-xx-xx or xxxxxxxxxxxxxx.
	• *lun_id* is the Logical Unit Number of a device.
	
	Default: 0 0 0
	
	Example:
	`TK_FIBRE_1_BOOT_PRIM=`
	`BB-CC-AA-BB-65-34-BB-F1 BB-CC-AA-BB-FF-34-BB-F1 9`

TK_FIBRE_*N*_BOOT_ALT1 = *target_wwnn target_wwpn lun_id*	Configures the operating system to use the indicated target as the first alternate boot device, where *N* is the HBA number to be configured, and
	• *target_wwnn* is the World Wide Node Name of a device, in the format xx-xx-xx-xx-xx-xx-xx-xx or xxxxxxxxxxxxxx.
	• *target_wwpn* is the World Wide Port Name of a device, in the format xx-xx-xx-xx-xx-xx-xx-xx or xxxxxxxxxxxxxx.
	• *lun_id* is the Logical Unit Number of a device.
	
	Default: blank
	
	Example:
	`TK_FIBRE_1_BOOT_ALT1=`
	`BB-CC-AA-BB-65-34-BB-FD BB-CC-AA-BB-FF-40-BB-F1 5`

Table 3. Fibre HBA boot configuration variables (continued)

Variable	Description
TK_FIBRE_*N*_BOOT_ALT2 = *target_wwnn target_wwpn lun_id*	Configures the operating system to use the indicated target as the second alternate boot device, where *N* is the HBA number to be configured, and • *target_wwnn* is the World Wide Node Name of a device, in the format xx-xx-xx-xx-xx-xx-xx-xx or xxxxxxxxxxxxxx. • *target_wwpn* is the World Wide Port Name of a device, in the format xx-xx-xx-xx-xx-xx-xx-xx or xxxxxxxxxxxxxx. • *lun_id* is the Logical Unit Number of a device. Default: blank Example: `TK_FIBRE_1_BOOT_ALT2= BB-CC-AA-BB-65-34-BB-FD BB-CC-AA-BB-FF-40-BB-F1 5`
TK_FIBRE_*N*_BOOT_ALT3 = *target_wwnn target_wwpn lun_id*	Configures the operating system to use the indicated target as the third alternate boot device, where *N* is the HBA number to be configured, and • *target_wwnn* is the World Wide Node Name of a device, in the format xx-xx-xx-xx-xx-xx-xx-xx or xxxxxxxxxxxxxx. • *target_wwpn* is the World Wide Port Name of a device, in the format xx-xx-xx-xx-xx-xx-xx-xx or xxxxxxxxxxxxxx. • *lun_id* is the Logical Unit Number of a device. Default: blank Example: `TK_FIBRE_1_BOOT_ALT3= BB-CC-AA-BB-65-34-BB-FD BB-CC-AA-BB-FF-40-BB-F1 5`

Capture and Deploy RAID Configuration and Windows image

The ServerGuide Scripting Toolkit provides sample Altiris jobs to perform operating system (OS) imaging tasks using Altiris. OS Imaging tasks includes capturing a copy of an installed OS and deploying an image to a target server. You might need to configure the target server before you deploy the OS image.

Note: The donor and target systems must be configured identically in order to deploy the captured RAID configuration and Windows image.

These jobs capture a RAID Configuration along with the OS image, so that you can use both to redeploy the image. In order to run the capture RAID configuration and Windows image sample job, you must first install Altiris Agent on the donor system. The Toolkit's scripted OS install jobs automatically install Altiris Agent.

The capture sample job captures the RAID configuration of the server, and then captures an OS image from the target server and stores both on the Altiris server.

Each deployment sample job deploys the captured RAID configuration, and OS image to the target server. The sample job configures RAID, calls a reboot, and then performs the image deployment.

The following variables may be customized for these jobs within the Altiris Console:

- TK_PRAID_PolicyFile=”Raid_Configuration.ini”
- TK_Path_PolicyFiles=”PolicyFiles\%Machine_Type%”

Configure RAID and Install Windows OS

This sample job performs a RAID configuration, restarts the system, and then performs a scripted operating system installation. You must set up the Windows operating system being deployed in the Toolkitsource tree on the Altiris server. See “Adding Windows installation files” on page 7 for details.

The following variables may be customized for this job within the Altiris Console:

- TK_Partition_DiskNum="0"
- TK_UXSP_ApplyLatest="No"
- TK_UXSP_UXSPIUpdateFlags=""
- TK_PRAID_PolicyFile=””
- TK_Path_PolicyFiles=”PolicyFiles”
- TK_Partition_Size=”8192”
- TK_NOS_NetworkOperatingSystem="Win2008_x64_EE"
- TK_NOS_AnswerFile="win2008x64.xml"
- TK_NOS_ProductKey=“
- TK_NOS_DeploymentDriverLibrary=”Auto”
- TK_NOS_PerformDirectorAgentInstallation=”No”
- TK_NOS_PerformPostOSInstallUXSPUpdates=”No”
- TK_UXSP_ForceReboot="No"

Configuring RAID with a policy file

You can configure RAID controllers by using stand-alone jobs that configure only the RAID controllers and then stop, and jobs integrated with scripted and image-based deployment. The jobs that specify a specific RAID level in them use a pre-configured policy file to create a RAID array of that type on the target machine. The other jobs either deploy a previously captured RAID configuration or create the default RAID configuration if no policy file is specified.

To use a user-defined policy file:

1. Create the policy file in the `sgdeploy\SGTKWinPE\PolicyFiles` directory, for example. UserPolicy1.ini. This is the default location specified by the following variable: TK_Path_PolicyFiles=”PolicyFiles”
2. Copy a pre-existing Toolkit job with RAID configuration.
3. Modify the TK_PRAID_PolicyFile variable in the copied job to point to your policy file. In this example: `TK_PRAID_PolicyFile="UserPolicy1.ini"`.

Capture and deploy RAID configuration

The ServerGuide Scripting Toolkit provides sample Altiris jobs to capture and deploy a RAID Configuration, allowing the RAID configuration of a single server to provide a configuration basis for any number of servers. The capture sample job captures the RAID configuration of the server and stores it in a file named RAID_Configuration.ini.

By default, all captured RAID configurations are stored in the following directory on the Deployment Server:

AltirisPath\sgdeploy\SGTKWinPE\PolicyFiles*machinetype*

where:

- *AltirisPath* is the fully qualifies path to the Altiris Deployment Solution directory, for example C:\Program Files\Altiris\eXpress\Deployment Server.
- *machinetype* is the machine type of the captured system.

The deployment sample job deploys the captured RAID configuration from RAID_Configuration.ini located in the above directory by default. You can customize the following variables for these jobs within the Altiris Console:

- TK_PRAID_PolicyFile=”Raid_Configuration.ini”
- TK_Path_PolicyFiles=”PolicyFiles\%Machine_Type%”

Server data disposal

These sample jobs perform a server data disposal. The first performs a disposal and resets the RAID Configuration. The second performs only a server data disposal.

The variables in the following table can be customized for these jobs within the Altiris Console.

Variable	Values
TK_Wipe_Level	• "quick" - performs a quick wipe of the disks. • "dod" - performs a multipass wipe of the disks that conforms to DOD standards.
TK_Wipe_Repeat_Number	*number* - indicates the number of passes to complete.
TK_Wipe_Disk	*disk_number* - indicates the number of disks to wipe. By default, "1" is the first disk in the system. **Note:** Disk numbers from **diskpart.exe** do not translate directly to **TK_Wipe_Disk**, because **diskpart.exe** numbers the first disk as disk zero, while **TK_Wipe_Disk** numbers it as disk one. You must add one to the disk number returned by **diskpart.exe** to wipe the correct disk with **TK_Wipe_Disk**.

Note: Depending on the size of the drives involved and the wipe level, these jobs can take up to several hours to complete.

Microsoft Windows unattended scripted installation

These sample jobs perform a scripted installation of Windows Server 2008. You must set up the version of the operating system to be installed on the Altiris server before running these jobs. See “Adding Windows installation files” on page 7 for more information.

You can customize the following variables for these jobs (values shown are defaults for Windows Server 2008, Enterprise Edition):

- TK_Partition_DiskNum="0"
- TK_UXSP_ApplyLatest="No"
- TK_UXSP_UXSPIUpdateFlags=""
- TK_Partition_Size=”8192”
- TK_NOS_NetworkOperatingSystem="Win2008_x86"
- TK_NOS_AnswerFile="win2008.xml"
- TK_NOS_DeploymentDriverLibrary=”Auto”
- TK_NOS_PerformDirectorAgentInstallation=”No”
- TK_NOS_PerformPostOSInstallUXSPUpdates=”No”
- TK_UXSP_ForceReboot="No"

Capture and deploy a Windows image

The ServerGuide Scripting Toolkit provides sample Altiris jobs to perform operating system (OS) imaging tasks using Altiris, such as capturing a copy of an installed OS and deploying an image to a target server.

Before deploying an OS image to a target server, you might need to perform RAID configuration on the server. To run the Capture Windows image sample job, Altiris Agent must be installed on the donor system. The Toolkit's scripted OS install jobs automatically install Altiris Agent.

To modify the file name and location of the captured image, perform the following steps:

1. Double-Click on the **Capture Windows Image** job.
2. Select the **Create Disk Image** task, and click **Modify**.
3. Change the path and file name for the captured image.
4. Click **Finish** to save.
5. Double-click on the **Deploy Windows Image** job.
6. Select the **Distribute Disk Image** task, and click **Modify**.
7. Change the path and file name to the same values used above for the captured image.
8. Click **Finish** to save the changes.

Windows PE Boot Test

This sample job performs a basic task to test connectivity between the target server and the Altiris Deployment Solution Server. The job boots into the Windows PE x86 environment on the target server. After establishing a connection, it performs a **dir** command on the eXpress share of the Altiris DS Server. The job then pauses, requiring input from you to continue. There are no variables to be customized for this job within the Altiris Console.

Chapter 5. System Enablement Packs

System Enablement Packs (SEPs) are a collection of files and utilities required to support a specific set of machine types. You can use SEPs to add support for systems that were released after the most current version of the Toolkit.

The files in a System Enablement Pack (SEP) include system specific utilities, drivers, and scripts. For the ServerGuide Scripting Toolkit to support a specific machine type, you must download and install the corresponding SEP.

Downloading SEPs

The Scripting Toolkit provides the **Altiris_DownloadSEPs.cmd** command to download System Enablement Packs (SEPs) for use in creating deployments. The command file is located in the `C:\Program Files\Altiris\eXpress\Deployment Server\sgdeploy\SGTKWinPE\Altiris\Altiris_DownloadSEPs.cmd` directory. When you run the **Altiris_DownloadSEPs.cmd** command, it downloads the SEP for the specified system. For example:

```
Altiris_DownloadSEPs.cmd 7327,7328
```

After you have downloaded the SEP, you must install it on the Toolkit source server in order to create deployments for the supported machine types.

Installing SEPs

The Toolkit provides three ways to install System Enablement Packs (SEPs)after they have been downloaded:

1. The **Altiris_DownloadSEPs.cmd** command can install the SEPs that it has downloaded.
2. The **Altiris_InstallSEPs.cmd** command will install SEPs that are stored in the default location - `sgshare\sgdeploy\updates\uxsp`, or from another location specified in the command line.
3. Running the **Altiris_SGTKWinPE.cmd** command to create a deployment scenario automatically prompts you to install any required SEPs.

Appendix A. Supported target server hardware and software

The ServerGuide Scripting Toolkit supports deployment of Windows operating systems on IBM eServer™ and IBM eServer™ xSeries servers. In general, the ServerGuide Scripting Toolkit provides support for ServerProven® IBM or third-party adapters in the following categories:

- Ethernet
- Fibre Channel
- IDE and IDE RAID
- SAS and SAS RAID
- SATA and SATA RAID
- SCSI and SCSI RAID (includes Ultra-SCSI)

This section contains the following information about specific hardware and software support for deployment scenarios:

- System support information
- Operating system support information
- RAID controller information
- Fibre Channel host bus adapters

Additional information about these topics is contained in the `readme.htm` file.

You can download the latest version of the `readme.htm` file from the ServerGuide Scripting Toolkit Web page. See IBM deployment resources on the Web for information.

System support

This section details what systems are supported by the ServerGuide Scripting Toolkit.

You can use the Scripting Toolkit to deploy supported operating systems to any IBM System x, BladeCenter, iDataPlex server, or Flex system that are in the ServerGuide Scripting Toolkit support list.

Operating system support

This section details what operating system deployment/server combinations are supported by the ServerGuide Scripting Toolkit.

You can use the Scripting Toolkit to deploy supported operating systems to systems supported by the ServerGuide Scripting Toolkit. To determine what operating system and server combinations are supported, see IBM ServerProven.

RAID controller support

This section details what RAID controller and server combinations are supported by the ServerGuide Scripting Toolkit.

To determine what RAID controller and server combinations are supported, see Storage Controllers and IBM ServerProven.

Fibre Channel HBA support

This section details what FC HBA and server combinations are supported by the ServerGuide Scripting Toolkit.

To determine what FC HBA and server combinations are supported, see Shared Storage Adapters and IBM ServerProven.

Appendix B. ServerGuide Scripting Toolkit utilities and tools

This section contains information about the utilities that are included in the ServerGuide Scripting Toolkit, and the tools that are shipped with it. For each utility there is a description of parameters, along with examples.

For each included tool there is a brief description of the tool and instructions on using it with the ServerGuide Scripting Toolkit, as well as pointers on where to get more information on the tool and its use.

Altiris_SGTKWinPE.CMD

Purpose

This script builds the IBM specific Windows Preinstallation Environment for Altiris Deployment Solution and integrates the Toolkit into the Altiris Deployment Solution.

Note:

1. This script must be executed from the directory in which it resides.
2. The Altiris Deployment Solution and Toolkit must be installed in a path that does not contain parentheses.

Altiris_SGTKWinPE

Parameters

There are no parameters

Sample

```
Altiris_SGTKWinPE
```

Altiris_DownloadSEPs.cmd

TheAltiris_ DownloadSEPs.CMD script is used to download System Enablement Packs for specific machine types to the ServerGuide Scripting Toolkit. The syntax is:

```
Altiris_DownloadSEPs machine_types [/?]
```

Parameters

machine_types
A comma-separated list of machine types for which to download System Enablement Packs. To download all available System Enablement Packs, use *all* as the machine type.

/? Displays the help.

Altiris_InstallSEPs.cmd

The Altiris_InstallSEPs.CMD script is used to install System Enablement Packs on the ServerGuide Scripting Toolkit source server. The syntax is:

```
Altiris_InstallSEPs sep_path [/F | /?]
```

Parameters

sep_path
The fully-qualified path to the folder containing the System Enablement Packs.

/F Forces the installation of all detected System Enablement Packs without prompting.

/? Displays the help.

Altiris_RefreshWinPEImage.cmd

The **Altiris_RefreshWinPEImage.cmd** script is used to update the Windows PE drivers and regenerate the Windows PE PXE image after installing a new System Enablement Pack on the ServerGuide Scripting Toolkit source server.

Use the **Altiris_SGTKWinPE.cmd** script when you install a new version of the ServerGuide Scripting Toolkit on your source server. Use the **Altiris_RefreshWinPEImage.cmd** script when you install a new System Enablement Pack on your source server. The **Altiris_RefreshWinPEImage.cmd** script maintains the job associations with Windows PE PXE image.

The syntax of the **Altiris_RefreshWinPEImage.cmd** is:

```
Altiris_RefreshWinPEImage
```

Tools included with the ServerGuide Scripting Toolkit

The ServerGuide Scripting Toolkit includes several additional tools to make the toolkit more efficient. This section describes the additional tools provided by this release of the ServerGuide Scripting Toolkit, Windows edition:

- Advanced Settings Utility
- QAUCLI
- UpdateXpress System Pack Installer

Advanced Settings Utility

For convenience, the ServerGuide Scripting Toolkit, Windows edition, includes the IBM Advanced Settings Utility (ASU). You can use the IBM® Advanced Settings Utility (ASU) to modify firmware settings from the command line on multiple operating-system platforms.

The ServerGuide Scripting Toolkit uses a subset of the ASU function to capture and deploy firmware settings as part of your scripted deployments.

Usage

This section describes the ASU functions used by the ServerGuide Scripting Toolkit.

Table 4. ASU functions in ServerGuide Scripting Toolkit

Command	Description
asu.exe show bios	Is used to display and capture CMOS settings. You can use redirection to store this output in a file as shown here: `asu.exe show bios > bios_settings.ini`

Table 4. ASU functions in ServerGuide Scripting Toolkit (continued)

Command	Description
asu.exe replicate *filename*	Is used to apply CMOS settings from a file. ASU looks for the filename specified by *filename*, and reads the contents. If the contents are valid CMOS settings, they are applied, one line at a time, to the server. This example applies the settings captured above: `asu.exe replicate bios_settings.ini` **Note:** Only settings captured from an identical model can be replicated, due to a difference in BIOS settings and valid values between models.

Updating the ASU executable

This section describes how to update the Advanced Settings Utility executable file used by the ServerGuide Scripting Toolkit, Windows Edition.

Before you begin

You need the following to update the ASU executable file used by the scripting toolkit:

- The file: `ibm_utl_asu_asut`*version*`_windows_i686.exe`, where *version* is the updated version of ASU.
- The file: `ibm_utl_asu_asut`*version*`_windows_x86-64.exe`, where *version* is the updated version of ASU.
- A scripting toolkit source server

Procedure

1. On the source server, navigate to `..\sgdeploy\SGTKWinPE\Bin\win32`.
2. Copy the file `ibm_utl_asu_asut`*version*`_windows_i686.exe` to that directory.
3. Execute the file.
4. When prompted, enter **A** to overwrite all of the old files.
5. When the update is complete, navigate to `..\sgdeploy\SGTKWinPE\Bin\winx64`.
6. Copy the file `ibm_utl_asu_asut`*version*`_windows_x86-64.exe` to that directory.
7. Delete the file `asu.exe`.
8. Execute `ibm_utl_asu_asut`*version*`_windows_x86-64.exe`.
9. When prompted, enter **A** to overwrite all of the old files.
10. Rename `asu64.exe` to `asu.exe` in this directory.

QAUCLI.EXE

You can use the QAUCLI utility to configure Fibre Host Bus Adapters (HBAs). 32– and 64–bit versions of this utility come with the ServerGuide Scripting Toolkit, Windows Edition. You can download this utility from QLogic at http://www.qlogic.com. You can also view the QAUCLI documentation in the `sgdeploy\SGTKWinPE\Docs\qaucli` directory.

Usage

Table 5. QAUCLI usage

Command	Description
qaucli.exe -pr fc –e (view | ?)	Shows the current boot device information on all HBAs
qaucli.exe -pr fc -e (hba_instance | hba_wwpn target_wwnn target_wwpn lun_id [prim | alt1 | alt2 | alt3])	Configures the operating system to boot from a particular target, where: **hba_instance** The HBA instance number of an HBA port. **hba_wwpn** The World Wide Port Name of an HBA port. **target_wwnn** The World Wide Node Name of a target device, in the format *nn-nn-nn-nn-nn-nn-nn-nn* or *nnnnnnnnnnnnnnnn*. **target_wwpn** The World Wide Port Name of a target device, in the format *nn-nn-nn-nn-nn-nn-nn-nn* or *nnnnnnnnnnnnnnnn*. **lun_id** The Logical Unit Number of a LUN. **prim** The primary boot port name. **alt***n* The name of the alternate boot port. You can specify up to three alternate boot ports.
qaucli.exe -pr fc -e (*hba_instance* | *hba_wwpn*) (view | ?)	Shows the current boot device information for the specified HBA port.
qaucli.exe -pr fc -e (*hba_instance* | *hba_wwpn*) (enable | 0 0 0)	Configures the operating system to boot from the first target found by the BIOS. The default LUN is 0.
qaucli.exe -pr fc -e (*hba_instance* | *hba_wwpn*) disable [prim | alt1 | alt2 | alt3]	Clears the selected boot device settings on the indicated HBA port.
qaucli.exe -pr fc -l (*hba_instance* | *hba_wwpn*)	Displays information about the LUNs attached to the specified HBA port.

Examples

The following examples illustrate qaucli.exe utility usage.

Note: While some of these examples are broken across multiple lines, when using QAUCLI.EXE, you must enter all of the parameters on a single line.

Example	Description
qaucli.exe -pr fc -e view	Displays the current boot device information on all HBAs.
qaucli.exe -pr fc –e E0-FF-EE-DE-CD-34-56-30 E0-00-ED-DE-CD-34-56-30 E0-10-ED-DE-CD-34-56-30 1 prim	Configures HBA E0-FF-EE-DE-CD-34-56-30 E0-00-ED-DE-CD-34-56-30 E0-10-ED-DE-CD-34-56-30 to boot from the primary target.
qaucli.exe -pr fc –e E0-FF-EE-DE-CD-34-56-30 view	Displays the current boot setting information for HBA port E0-FF-EE-DE-CD-34-56-30.
qaucli.exe -pr fc –e E0-FF-EE-DE-CD-34-56-30 disable prim	Clears the selected boot device setting on HBA port E0-FF-EE-DE-CD-34-56-30.

Example	Description
`qaucli.exe -pr fc –l E0-FF-EE-DE-CD-34-56-30`	Displays information about the LUNs attached to HBA port E0-FF-EE-DE-CD-34-56-30.

UpdateXpress System Pack Installer

For convenience, the ServerGuide Scripting Toolkit, Windows edition includes the UpdateXpress System Pack Installer (UXSPI) to help you acquire updates to include in your deployment scenarios.

The UpdateXpress System Pack Installer can perform these functions:

- Acquire firmware and driver updates for supported machine type/operating system combinations from a remote location, such as the IBM Support web site.
- Inventory a system to be updated and compare the inventory to the list of available updates, and then recommend and deploy a set of updates for the system.
- Create bootable media on CD-ROM, DVD, or USB key to use in applying firmware to supported systems.

For more information on running the UpdateXpress System Pack Installer, refer to the UXSPI User's Guide in the `sgdeploy\SGTKWinPE\Docs\uxspi` directory.

WINLPCFG.EXE

Use the WINLPCFG utility to configure Fibre Host Bus Adapters (HBAs). 32-bit and 64-bit versions of this utility come with the ServerGuide Scripting Toolkit, Windows Edition. You can download this utility from Emulex at http://www.emulex.com. You can also view the Emulex documentation in the `sgdeploy\SGTKWinPE\Docs\winlpcfg` directory.

Usage

Table 6. WINLPCFG usage

Command	Description

`winlpcfg.exe help`
`winlpcfg.exe ?`
`winlpcfg.exe help` *`command`*
`winlpcfg.exe ?` *`command`*
	Displays help for the **`winlpcfg.exe`** command.
	
	To view a list of all available commands, enter: **`winlpcfg.exe help`** or **`winlpcfg.exe ?`**.
	
	To view the help for a specific command, add the command name. For example:
	`winlpcfg.exe help download`
	
	or
	`winlpcfg.exe ? download`

`winlpcfg.exe listwwn`	Lists all adapters installed in the system and shows the factory-assigned WWN, the nonvolatile WWPN, and the WWNN used to identify the adapter in the SAN.

Table 6. WINLPCFG usage (continued)

Command	Description
`winlpcfg.exe listhba`	Lists the following information for all installed adapters in the system: • Adapter number • IEEE address assigned by the manufacturer • Firmware version • Adapter type • Possible mailbox errors
`winlpcfg.exe readbootdevice n=`*`adapter_number`*	Displays the WWN, LUN, and the topology in use for the indicated boot device.
`winlpcfg.exe enableboot n=`*`adapter_number`* `i=`*`index`*	Enables or disables the BootBIOS specified by the index number on the specified adapter.
`winlpcfg.exe setbootdevice n=`*`adapter_number`* `w0=` *`wwpn_word_0`* `w1=`*`wwpn_word_2`* `l=`*`lun`* `t=` *`topology`*	Sets the boot device to the indicated adapter, WWPN, and topology (select 0 for Arbitrated Loop or 1 for Point to Point).
`winlpcfg.exe readaltboot n=`*`adapter_number`*	Displays the WWN and LUN of all possible alternate boot devices. Up to seven alternate boot devices are supported.
`winlpcfg.exe setaltboot i=`*`index`* `w0=`*`wwpn_word_0`* `w1=` *`wwpn_word_2`* `l=`*`lun`*	Specifies an alternate boot device. You can set up to seven boot devices by specifying indexes from 1 to 7.

Examples

The following examples illustrate winlpcfg.exe utility usage.

Note: While some of these examples are broken across multiple lines, when using WINLPCFG.EXE, you must enter all of the parameters on a single line.

Example	Description
`winlpcfg.exe help`	Displays all available commands.
`winlpcfg listwwn`	Displays the WWNs of all adapters in the system
`winlpcfg listhba`	Lists all adapters in the system.
`winlpcfg readbootdevice n=1`	Displays the WWN, LUN, and topology for adapter number one.
`winlpcfg enableboot n=6 i=1`	Enables BootBios on adapter number 6.
`winlpcfg setbootdevice n=1 w0=a1b2c3d4 w1=b946a4e8 l=46 t=0`	Sets the boot device to adapter number one, LUN 46, with an Arbitrated Loop topology.
`winlpcfg readaltboot n=1`	Displays the WWN and LUN number of all possible alternate boot devices.
`winlpcfg setaltboot n=1 i=1 w0=12345678 w1=a842b6 l=3`	Sets the alternate boot device on adapter 1, LUN 3.

ServerGuide Scripting Toolkit utilities

This section contains information about the utilities that are included in the ServerGuide Scripting Toolkit. Each utility is describes, along with examples.

The command-line syntax examples in this documentation use the following conventions:

- Variables are shown in *italics*
- Required parameters are shown within <> brackets
- Optional parameters are shown within [] brackets
- Required or optional parameters from which you must make a unique choice are separated by a vertical bar (|) character

You must enter all parameters for a utility on a single command line, even when the information in this documentation is shown on multiple lines.

CLINI.EXE

The Command Line INI utility can perform the following functions:

- Write information to an INI file:
  - Add new sections, items, or values
  - Remove sections, items, or values
  - Change existing sections, items, or values
  - Change or append to values of existing sections or items
  - Comment or uncomment sections, items, or values
- Read information from an INI file:
  - Read items and store all or part of the value as an environment variable
  - Read items and check all or part of the value for strings, substrings, or tokens
- Merge information from one INI to another.

Two versions of the Command Line INI utility come with the ServerGuide Scripting Toolkit:

- A 32-bit version for use on Microsoft Windows 32-bit operating systems and the 32-bit version of Windows Preinstallation Environment (Windows PE) 2.1/3.0. The 32-bit version was formerly named clini32.exe.
- A 64-bit version for Windows x64 operating systems and for Windows PE 2.1/3.0 (x64).

Storing a value as an environment variable is done by creating a batch file that contains a command to set the environment variable. You must then call the batch file to set the environment variable. By default, the batch file is named cliniset.bat. If the batch file already exists, it is deleted and recreated with the new information.

Note: Because the media is read-only, this feature cannot be used on a bootable Windows PE CD or DVD.

In addition to setting values, the clini.exe program can append values to existing items in an INI file. By default, no delimiter is used to append values. A delimiter can be specified, if required. Appending values provides the ability to 'build' values in the INI file by issuing multiple commands. When reading values from an INI file to set an environment variable, the values can be tokenized to specify a particular token.

The clini.exe program checks the number of characters on the command line and displays a message if the characters exceed the limit. The /O parameter overrides character-limit checking.

The clini.exe utility has the following command-line syntax:

```
clini <filename> <[filename2 [/ES] [/A|/U|/P]]> <[/S:section]
[/I:item] [/V:value|/A:value|/U:value|/E:variable
|/=:string|/C:string|/CT:string]> [/B:file_name]
[/D:delimiter] [/T:n] [/R] <[/CMT|/UCMT| [/AI]
[/CC:character]]> [/NS] [/N] [/O]
```

Parameter	Description
filename	Defines the fully qualified path to the INI file to process
filename2	Defines the fully qualified path to an INI file to merge information into from *filename*. All values in *filename* are copied into *filename2*, replacing the value of any preexisting items in *filename2*.
/ES	Specifies to merge only the items or values in the empty section.
/A	Specifies to append values from items in *filename* to the items in *filename2* instead of replacing them. An optional delimiter can be specified using the /D:*delimiter* parameter.
/U	Specifies to uniquely append values from items in *filename* to the items in *filename2* instead of replacing them; only if the value doesn't already exist. An optional delimiter can be specified using the /D:*delimiter* parameter.
/P	Specifies that the data in *filename2* is persistent. If duplicate items are found, they are not replaced.
/S:*section*	Specifies the name of the section within the INI file to write or to read.
/I:*item*	Specifies the name of the item within the INI file to write or to read.
/V:*value*	Specifies the value to write to the INI file.
/A:*value*	Specifies the value to append to the existing item in the INI file. The /I parameter is required to use the /A:*value* parameter.
/U:*value*	Specifies a unique value to append to the existing item in the INI file, only if this value does not already exist for the item. The /I parameter is required to use the /U:*value* parameter.
/E	Convert multiple Items to Environment Variables. The Item name is used for the environment variable name. Use the /NS parameter to replace any spaces in the item names with underscore characters when creating the Environment Variables, if spaces are not desired.
/E:*variable*	Specifies the environment variable used to store the value of the item from the INI file. The /I parameter is required to use the /E:*variable* parameter. If the item specified by the /I parameter does not exist, or the section specified by the /S parameter does not exist, the environment variable has no value in the batch file created by clini.exe. If the environment variable exists on the system, it is deleted when the batch file runs.
/=:*string*	Verifies that the value of the item is equal to *string*, returning a value of 0 if true and 100 if false.
/C:*string*	Verifies that value of the item has *string* as a substring, returning a value of 0 if true and 100 if false.
/CT:*string*	Verifies that the value of the item has *string* as one of the tokens, returning a value of 0 if true and 100 if false. The default delimiter is a comma unless the /D:delimiter option is specified.
/B:*filename*	Defines the fully qualified path and file name of the batch file to create for setting the environment variable. The default is CLIniSet.bat if no file name is specified for this parameter. This parameter is only valid when the /E parameter is used.

Parameter	Description
/D:*delimiter*	Specifies a delimiter to use when appending values to an item in an INI file or reading tokens from an INI file. This parameter is not valid if the /V parameter is used. The /D parameter is valid only with the /A, /U, or /E parameters. Using the /D parameter without one of these three parameters results in a syntax error.
/T:*n*	Specifies the token in a delimited value to set as the specified environment variable, where *n* is a positive integer. The default delimiter is a comma unless otherwise specified with the /D parameter. This parameter is only valid with the /E parameter.
/R	Removes the specified section, item, or value from the INI file. Removing the last item in a section also removes the section.
/CMT	Specifies to comment out the line indicated by the Section, Item, or Value parameter, if it exists in the INI file. It also allows use of the /AI parameter.
/UCMT	Specifies to Uncomment the line indicated by the Section, Item, or Value parameter, if it exists in the INI file. It also allows use of the /AI parameter.
/CC:*character*	Specifies the comment character to use when commenting or uncommenting lines. If omitted, the default comment character is the semicolon. This parameter is only valid with the /CMT or /UCMT parameters.
/AI	Specifies to explicitly treat the /V parameter as the value to all items when commenting or uncommenting. This parameter is only valid when using the /CMT or /UCMT parameters.
/N	Deletes an existing INI file and creates a new INI file. This parameter is not valid with the /E parameter.
/NS	Omits spaces around "=" when writing items into INI files. By default, the clini.exe utility concatenates spaces around "=" when writing items.
/O	Overrides the command-line character count. The number of characters on the command line is automatically determined by this utility. An error message is displayed when the character limit is reached, unless you override this feature. The Windows command line is limited to 8189 characters.

The clini.exe utility returns the following values to indicate status:

Value	Description
0	Success or true
1	Syntax error
2	Program error
3	Destination is read-only
4	Current® working directory is read-only.
5	File not found
100	False

The following examples illustrate Command Line INI utility usage.

Example	Description
`clini info.ini /S:Hardware /I:Machine Type /V:8549 /N`	Deletes any existing info.ini file and creates a new INI file named info.ini with a section called Hardware that contains one item, Machine Type, which has a value of "8549"

Example	Description
`clini info.ini /S:Hardware /I:Machine Name /V:Server1`	Adds the item Machine Name with a value of Server1 to the existing Hardware section of the info.ini file

`clini info.ini /S:Hardware /I:Machine Type /E:MachineType`
 	
`call CLIniSet.bat`
	Reads the Machine Type value from the info.ini file, and stores it as an environment variable called MachineType

`clini info.ini /S:Hardware /I:Machine Type2 /V:%MachineType%`	Writes the value of the environment variable *MachineType* to the INI file named info.ini, using section Hardware and item Machine Type2

`clini info.ini /S:Hardware /I:Machine Type2 /E:MachineType2 /B:d:\EnvSet1.bat`
 	
`call d:\EnvSet1.bat`
	Reads the machine type value from the info.ini file and stores it as an environment variable called MachineType2 using a custom path and name for the batch file created to set the environment variable

`Clini info.ini /S:MySection /E`
 	
`Call cliniset.bat`
	This example creates environment variables for all the items found in section MySection.

`Clini info.ini /AI /E /B:setthem.bat`
 	
`Call setthem.bat`
	This example creates environment variables for all the items found in any section of the info.ini file and uses an alternate name for the CLIniSet.bat file.

`Clini info.ini /S:MySection /I:MyItem /E`
 	
`Call cliniset.bat`
	This example creates an environment variable called MyItem if it exists in the info.ini file.

`Clini info.ini /S:MySection /I:My Item /E /NS`
 	
`Call cliniset.bat`
	This example creates an environment variable called My_Item (converts the space to an underscore for the environment variable name) if the item exists in the info.ini file.

After running the first five examples above, in sequence, the info.ini file would contain the following information:
`[Hardware]`
`Machine Type = 8549`
`Machine Type2 = 8549`
`Machine Name = Server1`
 	
Also, two new environment variables would be created as indicated below:
`MachineType = 8549`
`MachineType2 = 8549`	

`clini info.ini /S:User /I:Name /V:Toolkit /N`
`clini info.ini /S:User /I:Name /A: User`
 	
`or`
 	
`clini info.ini /S:User /I:Name /V:Toolkit /N`
`clini info.ini /S:User /I:Name /A:User /D:" "`
	Creates a new file named info.ini with a section called User and one item called Name, which is set equal to "Toolkit User". The resulting info.ini file contains:
	`[User]`
	`Name = Toolkit User`

`clini info.ini /S:Section /I:Item /A:Value1 /D:, /N`
`clini info.ini /S:Section /I:Item /A:Value2 /D:,`
`clini info.ini /S:Section /I:Item /A:Value3 /D:,`
`clini info.ini /S:Section /I:Item /A:Value2 /D:,`
	Creates a new file named info.ini with a comma delimited list of values. The resulting info.ini file contains:
	`[Section]`
	`Item = Value1,Value2,Value3,Value2`

`clini info.ini /S:Section /I:Item /U:Value1 /D:, /N`
`clini info.ini /S:Section /I:Item /U:Value2 /D:,`
`clini info.ini /S:Section /I:Item /U:Value3 /D:,`
`clini info.ini /S:Section /I:Item /U:Value2 /D:,`
	Creates a new file named info.ini with a comma delimited list of unique values. The resulting info.ini file contains:
	`[Section]`
	`Item = Value1,Value2,Value3`

Example	Description

`clini info.ini /S:Section /I:Item /E:MyEVariable /T:2`
 	
or
 	
`clini info.ini /S:Section /I:Item /E:MyEVariable /T:2 /D:,`
	Reads information from the info.ini file created in the previous example, and sets the second value of the item to the MyEVariable environment variable. The resulting CLIniSet.bat file contains:
	`Set MyEVariable=Value2`

Content of doit.bat:
 	
`@Echo off`
`clini info.ini /S:Secton /I:Item /V:Value1 /N`
`clini info.ini /S:Section /I:Item /=:Value1`
`if errorlevel 100 goto itsfalse`
`if errorlevel 1 goto error`
`if errorlevel 0 goto itstrue`
`:error`
`Echo Error occurred`
`Goto end`
`:itsfalse`
`Echo It's false`
`Goto end`
`:itstrue`
`Echo It's true`
`:end`
	This example creates a file called info.ini with the following content:
	`[Section]`
	`Item = Value1`
	
	Then it checks to see if the value of Item in `[Section]` is equal to `Value1` and displays a message.
	
	After running `doit.bat`, the follow message is displayed:
	`It's true`

Content of doit.bat:
 	
`@Echo off`
`clini info.ini /S:Section /I:Item /V:Value1 /N`
`clini info.ini /S:Section /I:Item /C:alu`
`if errorlevel 100 goto itsfalse`
`if errorlevel 1 goto error`
`if errorlevel 0 goto itstrue`
`:error`
`Echo Error occurred`
`Goto end`
`:itsfalse`
`Echo It's false`
`Goto end`
`:itstrue`
`Echo It's true`
`:end`
	This example creates a file called info.ini with the following content:
	`[Section]`
	`Item = Value1`
	
	Then it checks to see if the value of Item in `[Section]` contains substring `alu` and displays a message.
	
	After running `doit.bat`, the follow message is displayed:
	`It's true`

Content of doit.bat:
 	
`@Echo off`
`clini info.ini /S:Section /I:Item /V:V1,V2,V3 /N`
`clini info.ini /S:Section /I:Item /CT:V2`
`if errorlevel 100 goto itsfalse`
`if errorlevel 1 goto error`
`if errorlevel 0 goto itstrue`
`:error`
`Echo Error occurred`
`Goto end`
`:itsfalse`
`Echo It's false`
`Goto end`
`:itstrue`
`Echo It's true`
`:end`
	This example creates a file called info.ini with the following contents:
	`[Section]`
	`Item = V1,V2,V3`
	
	Then it checks to see if the value of Item in `[Section]` contains token `V2` in a comma delimited list and displays a message.
	
	After running doit.bat, the follow message is displayed:
	`It's true`

`Clini info1.ini info2.ini`	This example copies all the sections, items, and values from info1.ini into info2.ini. Any existing values for items in info2.ini are replaced.

Example	Description
`Clini info1.ini info2.ini /P`	This example copies all the sections, items, and values from info1.ini into info2.ini. Any values for existing items in info2.ini are kept. Only new items and values are copied over from info1.ini.
`Clini info1.ini info2.ini /S:MySection`	This example copies all the items and values from the section called MySection in info1.ini into the section called MySection in info2.ini replacing any values that may already exist in the section called MySection in info2.ini.
`Clini info1.ini info2.ini /S:MySection /I:MyItem`	This example copies the value from the section called MySection, for the Item called MyItem in info1.ini into the same section and item in info2.ini replacing the existing value in info2.ini if it already exists.
`Clini info1.ini info2.ini /ES`	This example copies all the items and values from the empty section (items and values that are not in a section) in info1.ini into info2.ini replacing any existing Items in the empty section in info2.ini.
`Clini info1.ini info2.ini /A`	This example appends all the values from the sections and items from info1.ini to info2.ini.
`Clini info1.ini info2.ini /U`	This example uniquely appends all the values from the sections and Items from info1.ini to info2.ini if the value does not already exist in info2.ini.
`Clini info1.ini info2.ini /U /D:`	This example uniquely appends all the values from the sections and items from info1.ini to info2.ini using a comma as the delimiter if the value does not already exist in info2.ini.
`Clini info.ini /V:My Ini Line /CMT`	This example comments out the line `My Ini Line` in the empty section in the info.ini file with a semicolon if the line exists.
`Clini info.ini /S:MySection /V:My Ini Line /UCMT`	This example uncomments the line `My Ini Line` in the MySection section of the info.ini if the line exists.
`Clini info.ini /I:MyItem /CMT`	This example comments out the line idicated by the item MyItem in the empty section of the info.ini file if the item exits.
`Clini info.ini /S:MySection /I:MyItem /CMT /CC:#`	This example comments out the line indicated by the item MyItem in the section MySection in the info.ini file with a # sign if the item exists.
`Clini info.ini /s:MySection /AI /V:My Value /CMT`	This example comments out the lines indicated by any item that has a value of *My Value* of all the items in the section MySection in the info.ini file if the item exists.
`Clini info.ini /s:MySection /CMT`	This example comments out the section header indicated by MySection in the info.ini file if the section exists.

DDCOPY.EXE

The Device Driver Copy (ddcopy.exe) utility can copy only those drivers in the driver set that support a specific machine.

Two versions of the utility come with the ServerGuide Scripting Toolkit:

- A 32-bit version for Windows 32-bit operating systems and for the Windows Preinstallation Environment (Windows PE) 2.1/3.0 (32-bit)

- A 64-bit version for Windows x64 operating systems and for Windows PE 2.1/3.0 (x64).

Microsoft uses the term *device-driver directory* to refer to the directory that contains drivers for an individual device. All device-driver files are located in individual device-driver directories that contain the files for an individual device driver. The following directories are device-driver directories:

- \adaptec
- \asm
- \hal
- \ideraid
- \srvraid

However, when specifying the source path for the ddcopy command, always specify the directory that contains the `drvset.ini` file. In this case, the source directory is the `C:\w03_drv\$oem$\$1\drv directory`.

When you issue a ddcopy command against a directory that contains device-driver directories. Ddcopy copies all of the device drivers that are specific to the specified machine types and any other files located in the drv directory to the new location.

The SupportedSystems keyword in the drvset.ini file is modified to reflect the new machine list.

The ddcopy.exe utility has the following command-line syntax:

```
ddcopy <source_path> <destination_path> [/M:machine_types or platform_ids]
[/C:category] | [/V:n] [/?]
```

Parameter	Description
source_path	Defines the fully qualified path to the directory that contains the device-driver directories and the drvset.ini file.
destination_path	Specifies the fully qualified path of the target directory for copying the device drivers.
/M:*machine_types* or *platform_ids*	Specifies machine types or platform IDs to limit the number of drivers that are copied. Multiple machine types or platform IDs are allowed when delimited by commas.
/C:category	Specifies the driver categories to limit the device drivers to be copied. Multiple categories can be specified using a comma as the delimiter. If omitted, then all the device driver categories are copied. Valid values are: • Network • Video • Management • Chipset • Mass Storage • Application • Tape • Hotfix. This is only valid with driver sets from ServerGuide 7.4.12 or greater.

Parameter	Description
/V:*n*	Specifies the verbose level used to report status during the deployment process. Valid values for n are: • **0** - quiet mode • **3** - default • **5** - maximum information

The ddcopy.exe utility returns the following values to indicate status:

Value	Description
0	Success
1	Syntax error
2	Program error
3	Failed to copy
4	Machine type or platform ID not found
5	Destination is read-only
6	File not found

The following examples illustrate ddcopy.exe utility usage.

Example	Description
`ddcopy d:\drivers\$oem$\$1\drv c:\wininst\$oem$\$1\drv`	Copies all the drivers from the d:\drivers\$oem$\$1\drv directory to the c:\wininst\$oem$\$1\drv directory
`ddcopy d:\drivers\$oem$\$1\drv c:\wininst\$oem$\$1\drv /M:8832`	Copies the drivers that are specifically for machine type 8832 from the d:\drivers\$oem$\$1\drv directory to the c:\wininst\$oem$\$1\drv directory and updates the supported systems field in the drivers' DrvInfo.ini file and the c:\wininst\$oem$\$1\drv\drvset.ini file.
`ddcopy d:\drivers\$oem$\$1\drv c:\wininst\$oem$\$1\drv /M:8832,8865`	Copies the drivers for machine type 8832 and machine type 8865 from the d:\drivers\$oem$\$1\drv directory to the c:\wininst\$oem$\$1\drv directory and updates the supported systems field in the drivers' DrvInfo.ini file and the c:\wininst\$oem$\$1\drv\drvset.ini file.
`ddcopy d:\drivers\$oem$\$1\drv c:\mydrvs /m:8832 /c:network,video`	Copies the Network and Video drivers specific to the machine type 8832 from the d:\drivers\$oem$\$1\drv directory to the c:\mydrvs directory and updates the supported systems field in the drivers' DrvInfo.ini file as well as the c:\mydrvs\drvset.ini file.

DSCAN.EXE

The Driver Scan utility can perform the following functions:

- Scan a device driver or set of device drivers to determine the installation mode (text mode, Plug and Play, or executable) and write this information to the drvinfo.ini file that is located in each device-driver directory. The drvinfo.ini file is used by the unattend.exe command during the installation of Windows operating systems.
- Create a text mode directory, copy all text mode device drivers into that directory, then dynamically create a master txtsetup.oem file that contains all of the unique information that is in the individual txtsetup.oem files. Known unattended installation defects are automatically addressed.

Two versions of the utility come with the ServerGuide Scripting Toolkit:

- A 32-bit version for Windows 32-bit operating systems and for the Windows Preinstallation Environment (Windows PE) 2.1/3.0 (32-bit)
- A 64-bit version for Windows x64 operating systems and for Windows PE 2.1/3.0 (x64).

Microsoft uses the term *device-driver directory* to refer to the directory that contains drivers for an individual device. All device-driver files are located in individual device-driver directories that contain the files for an individual device driver. The following directories are device-driver directories:

- \adaptec
- \asm
- \hal
- \ideraid
- \srvraid

However, when specifying the source path for the ddcopy command, always specify the directory that contains the `drvset.ini` file. In this case, the source directory is the `C:\w03_drv\$oem$\$1\drv directory`.

When you issue a dscan command against a directory that contains device-driver directories, dscan performs its tasks against all of the subdirectories that the directory contains, with the exception of the drvutils directory. The drvutils directory contains two utilities, Holdit.exe and Reboot.exe, that are used by the unattend utility.

The Driver Scan utility stores information in an INI file named drvinfo.ini in the device-driver directory, for use by the unattend.exe utility. If the drvinfo.ini file already exists for the device driver, it is left unchanged. See "DRVINFO.INI" on page 55 for information about the drvinfo.ini file. See "UNATTEND.EXE" on page 89 for information about the unattend.exe utility.

The Driver Scan utility can also merge text mode device drivers into a single directory. This merges the device-driver files and the txtsetup.oem files for use in unattended installations. If the destination directory for text mode drivers already exists, it is automatically deleted and recreated.

The Driver Scan utility automatically assumes that the device driver being scanned is applicable to all target servers. To make a device driver server-specific, you must modify the drvinfo.ini file to reflect the servers that the device driver supports.

The dscan.exe utility has the following command-line syntax:

```
dscan <driver_path> [/S|/SS|/T[:path]] [/M:machine_type/platform_ID] [/H:filename
[/OW] [/V:n] [/W:n] [/O:file_name] [/?]
```

Parameter	Description
driver_path	Defines the fully qualified path to the directory to scan for device drivers. Each driver is assumed to be in a separate subdirectory within this path. If *driver_path* has oem in the path, the Driver Scan utility creates the oem\textmode directory and merges the text mode device drivers. If the /SS parameter is used, the path is assumed to be the path to a single device driver.
/S	Specifies to scan device drivers and create drvinfo.ini files only, if necessary. Text mode device drivers are not merged when this parameter is used.
/SS	Specifies to scan a single device driver and create the drvinfo.ini file only, if necessary. Text mode device drivers are not merged when this parameter is used.
/T[:*path*]	Specifies to build the text mode device drivers only. Other device drivers are not scanned, and drvinfo.ini files are not created when this parameter is used. If *path* is specified, the text mode device drivers are merged to the specified path. Otherwise, the *driver_path* parameter must have oem in the path so that the text mode device drivers are merged into the oem\textmode directory.
/M:*machine_type/platform_ID*	Specifies a machine type, where *machine_type* is the machine type of the target server or platform ID, where *platform_ID* is the platform ID of the target server, that is used to limit merging of the text-mode device drivers. If this parameter is not specified, all text-mode device drivers are merged. The /T parameter is required to use this parameter.
/H:*filename*	Specifies a fully-qualified path and file name for the hwdetect.ini file that was created by the hwdetect.exe utility. This will limit the merging of the text mode device drivers to only those drivers detected in the system.
/OW	Overwrites the text-mode drivers without deleting and recreating the text-mode directory. This parameter is not valid with the /S parameter or the /SS parameter.
/V:*n*	Specifies the verbose level used to report status during the deployment process. Valid values for *n* are: • **0** - quiet mode • **3** - default • **5** - maximum information
/W:*n*	Specifies the version of Microsoft Windows for the device drivers: • **0** for Windows 2000 • **1** for Windows Server 2003 • **2** for Windows 2000 Professional • **3** for Windows XP • **4** for Windows Server 2003 x64
/O:*file_name*	Combines the information in the DrvInfo.Ini files into a single file specified by the *file_name* value
/?	Displays usage information

The dscan.exe utility returns the following values to indicate status:

Value	Description
0	Success
1	Syntax error
2	Program error
3	Destination is read-only

The following examples illustrate Driver Scan utility usage.

Example	Description
`dscan c:\insttemp\$oem$\$1\drv`	Scans a device-driver set in c:\insttemp\$oem$\$1\drv, creates the drvinfo.ini files for each device driver, and builds the text-mode directory
`dscan c:\drv /S`	Scans a device-driver set in c:\drv and creates drvinfo.ini files for each device driver, but does not build the text mode device drivers
`dscan c:\drv\mydriver /SS`	Scans a single device driver in d:\drv\mydriver and creates the drvinfo.ini file for that device driver, but does not build the text mode device driver
`dscan c:\w2\$oem$\$1\drv /T`	Builds the text mode directory in c:\w2\$oem$\ textmode using device drivers found in c:\w2\$oem$\$1\drv, but does not create any drvinfo.ini files
`dscan c:\drivers /T:c:\other\textmode`	Builds the text mode directory in c:\other\textmode using device drivers found in c:\drivers, but does not create drvinfo.ini files

DRVINFO.INI

The drvinfo.ini file contains information specific to each device driver. The unattend.exe utility uses this information to add device-driver information to the answer file for Windows deployment scenarios. You can create this file, or have the Driver Scan utility create it automatically.

The drvinfo.ini file contains one section, called [Driver Information], and can contain the following valid variables:

Variable name	Description
Automatically Reboots	Specifies whether the executable device driver automatically restarts (reboots) the target server after the device-driver installation has completed. This variable is only valid when Installation Mode is set to “Executable”. Valid values are True or False. The default value is False. If an executable-device-driver installation program restarts the server and this variable is set to False, then any remaining installation procedures are not completed. To use this variable, the drvutils directory must contain the Holdit.exe and Reboot.exe utilities. This variable is only supported for Windows 2000.
Installation Mode	Specifies the installation method for the device driver. Valid values are: Executable, Manual, PnP, or Textmode. If set to *Manual*, the unattend.exe utility does not install the device driver.

Variable name	Description
Order Before	Specifies that the device driver is added to the answer file before another specified device driver. This variable is valid only when Installation Mode is set to “PnP” or “Executable”. Valid values are a comma-delimited list of the names of the device-driver directories, or the special keyword “All”. If more than one device driver has a value of “All”, the device drivers are installed in alphabetical order before those that do not have the specification.
Order After	Specifies that the device driver is added to the answer file after all other device drivers. This variable is only valid when Installation Mode is set to “PnP” or “Executable”. The only valid value is the special keyword “All”. If more than one device driver has this value set to “All”, the device drivers are installed in alphabetical order after those that do not have the specification.
Parameters	Defines any required command-line parameters required by the executable device driver. This variable is only valid when Installation Mode is set to *Executable*.
Path	Specifies the path to the installation file. This variable has a different function, depending on the setting of the Installation Mode variable, as indicated below: • **Executable** - Path specifies the path to the executable installation file • **Manual** - Path variable is ignored • **PnP** - Path specifies the path to the INF installation files • **Textmode** - Path specifies the path to the txtsetup.oem file
PCIVenDevID	Specifies the PCI Vendor ID or Device ID information used to limit the installation of executable device drivers to only when the specified device is in the target server. This variable is only valid when Installation Mode is set to *Executable*. Entries must be in the same format as those in the txtsetup.oem file, with multiple entries delimited by commas. For example: `PCIVenDevID = PCI\VEN_1002&DEV_5159&SUBSYS_029A1014`
Reboot Required	Specifies whether the executable device driver requires the target server to restart (reboot) after the installation of the device driver is completed. This variable is only valid when Installation Mode is set to *Executable*. Valid values are True or False. To use this variable, the drvutils directory must contain the Holdit.exe and Reboot.exe utilities. This variable is only supported for Windows 2000.
Supported Locales	Specifies the locales supported by this device driver. This value can be All, or a comma-delimited list of locales, as specified by the Localization variable in the ProdSpec.ini file from the i386 directory of the applicable operating system installation directory. You cannot use both Supported Systems and Unsupported Systems in the same drvinfo.ini file.
Supported Systems	Specifies the servers supported by this device driver. This value can be All, None, or a comma-delimited list of server machine types or platform IDs You cannot use both Supported Systems and Unsupported Systems in the same drvinfo.ini file.
Unsupported Locales	Specifies locales that are not supported by this device driver. This value must be a comma-delimited list of locales, as specified by the Localization variable in the ProdSpec.ini file from the i386 directory of the applicable operating system installation directory. You cannot use both Supported Locales and Unsupported Locales in the same drvinfo.ini file.

Variable name	Description
Unsupported Systems	Specifies servers that are not supported by this device driver. This value must be a comma-delimited list of server machine types or platform IDs. You cannot use both Supported Systems and Unsupported Systems in the same drvinfo.ini file.

The following examples illustrate drvinfo.ini file contents.

Example	Description

`[Driver Information]`
`Installation Mode=PnP`
`Path=`
`Parameters=`
`Automatically Reboots=`
`Reboot Required=`
`PCIVenDevID=`
`Order Before=`
`Supported Systems=All`
`;Unsupported Systems=`
`Supported Locales=`
`;Unsupported Systems=`
	Supports a plug-and-play device driver with INF files in the root of the device driver directory, and supports all target servers

`[Driver Information]`
`Installation Mode=PnP`
`Path=win2000`
`Parameters=`
`Automatically Reboots=`
`Reboot Required=`
`PCIVenDevID=`
`Order Before=`
`Supported Systems=8673,8679,8687`
`;Unsupported Systems=`
`Supported Locales=`
`;Unsupported Systems=`
	Supports a plug-and-play device driver with INF files in the win2000 directory in the root of the device driver directory, and only supports target servers with machine types 8673, 8679, and 8687

`[Driver Information]`
`Installation Mode=PnP`
`Path=win2000`
`Parameters=`
`Automatically Reboots=`
`Reboot Required=`
`PCIVenDevID=`
`Order Before=All`
`Supported Systems=8673,8679,8687`
`;Unsupported Systems=`
`Supported Locales=`
`;Unsupported Systems=`
	Supports a plug-and-play device driver that must be installed before any other plug-and-play device driver, with INF files in the win2000 directory in the root of the device driver directory, and supports only target servers with machine types 8673, 8679, and 8687

`[Driver Information]`
`Installation Mode=Executable`
`Path=win2000\setup.exe`
`Parameters=-Q`
`Automatically Reboots=False`
`Reboot Required=True`
`PCIVenDevID=PCI\VEN_1002&DEV_5159&SUBSYS_1014029A`
`Order Before=All`
`;Supported Systems=`
`Unsupported Systems=8687`
`Supported Locales=`
`;Unsupported Systems=`
	Supports an executable device-driver installation using setup.exe, with a -Q parameter, in the win2000 directory in the root of the device driver directory, and supports all target servers (except those of machine type 8687) that have a device installed matching the specified Vendor/Device ID.

HWDETECT.EXE

HWDETECT is used to perform basic hardware detection functions that are typically obtained using SMBIOS and a PCI scan. This utility contains options that can be used to dump all of the hardware information to an output file, or it can be used to query hardware information and return values that set the *errorlevel* environment variable or the return code in Windows PE.

HWDETECT has basic hardware scan functions, and more complex PCI device detection options. The basic hardware scan functions can only be used singularly. The PCI device detection functions may be used in combination with each other to produce a query based on multiple restrictions.You can only use the hwdetect.exe utility basic hardware scan functions one at a time. The PCI-device detection functions can be combined or used more than once on the same command line.

Two versions of HWDETECT.EXE are provided with the ServerGuide Scripting Toolkit:

- A 32-bit version for use with Windows 32-bit operating systems and Windows PE 2.1/3.0 (32-bit).
- An x64 version for use with Windows x64 operating systems and Windows PE 2.1/3.0 (x64)

The hwdetect.exe utility has the following command-line syntax:

```
hwdetect [/s|/p|/i|/m:type] [/vid:vendor_id] [/did:device_id] [/svid:sub_vendor_id]
[/sdid:sub_device_id] [bn:bus_number] [/dn:device_number] [/add:num]
```

Parameter	Description
Basic hardware scan functions	
`/s`	Determines if the target server is an IBM eServer or IBM eServer xSeries server. The return values are: • **0** for an IBM eServer or IBM eServer xSeries server • **1** for a non-IBM eServer or IBM eServer xSeries server
`/p`	Displays all hardware information for the target server in a variable=value format. The return value is 0 when successful. You can use the > output-redirect option to save the output to an output file. For example, `hwdetect /p > filename`
`/i`	Displays all hardware information for the target server in an INI-file format. The return value is 0 when successful. You can use the > output-redirect option to save the output to an output file.
`/m:type`	Compares the machine type of the target xSeries server to the specified machine type, *type*. The return values are: • **0** when the machine types are different or no basis for comparison exists • **1** when the machine types match

Parameter	Description
PCI-device detection functions	
`/vid:vendor_id`	Determines whether there is a PCI adapter in the target server that matches the specified vendor ID, where *vendor_id* is a hexadecimal value. The return value is the number of matching adapters in the target server, or 0 if there are no matches.

Parameter	Description
/did:*device_id*	Determines whether there is a PCI adapter in the target server that matches the specified device ID, where *device_id* is a hexadecimal value. The return value is the number of matching adapters in the target server, or 0 if there are no matches.
/svid:*sub_vendor_id*	Determines whether there is a PCI adapter in the target server that matches the specified sub-vendor ID, where *subvendor_id* is a hexadecimal value. The return value is the number of matching adapters in the target server, or 0 if there are no matches.
/sdid:*sub_device_id*	Determines whether there is a PCI adapter in the target server that matches the specified sub-device ID, where *subdevice_id* is a hexadecimal value. The return value is the number of matching adapters in the target server, or 0 if there are no matches.
/bn:*bus_number*	Causes the PCI scan to begin at the specified bus number, instead of starting at bus 0, by default. This parameter is only valid when more than one /vid, /did, /svid, or /sdid parameter is specified on the command line.
/dn:*device_number*	Causes the PCI scan to begin at the specified device number, instead of starting at device number 0, by default. This parameter is only valid when the /bn parameter is specified on the command line.
/add:*num*	Adds an integer number, *num*, to the return value before exiting. This is useful to obtain a sum of different PCI adapters, with different PCI IDs, in a target server. The return value is the resultant sum of all other return values plus *num*.

The following examples illustrate hwdetect.exe utility usage.

Example	Description

`hwdetect /s`
`if errorlevel 1 goto NONIBM`
`if errorlevel 0 goto IBM`
 	
`:NONIBM`
`rem Perform non-IBM equipment specific steps here`
`goto FINISH`
 	
`:IBM`
`rem Perform IBM equipment specific steps here`
 	
`:FINISH`
	Determines if the target server is an IBM server or not, and branches accordingly to perform equipment-specific steps

`hwdetect /m:8676`
`if errorlevel 1 goto 8676`
 	
`hwdetect /m:8669`
`if errorlevel 1 goto 8669`
 	
`echo System not supported!`
`goto done`
 	
`:8676`
`call 8676.bat`
`goto done`
 	
`:8669`
`call 8669.bat`
`goto done`
 	
`:done`
	Determines if the target server is either a machine type 8676 or machine type 8669 server, and branches accordingly to call a system-specific batch file or displays a message of non-support for other machine types

Example	Description
`hwdetect /i>hwdetect.out` `clini hwdetect.out /S:CI /I:Vendor_ID.0 /E:Vendor CLIniSet.bat`	Creates an output file that lists the hardware configuration for the target server, so that the clini.exe utility can search for a specific PCI adapter from a vendor and set en environment variable accordingly
`hwdetect /vid:0x9005 /did:0x0250` `if errorlevel 1 call 6Mstuff.bat`	Determines if there is at least one IBM ServeRAID 6i/6i+/6M PCI adapter in the target server, and calls a batch file to process adapter-specific tasks

`hwdetect /vid:0x9005 /did:0x0250`
`if errorlevel 0 set TOTAL=0`
`if errorlevel 1 set TOTAL=1`
`if errorlevel 2 set TOTAL=2`
`if errorlevel 3 set TOTAL=3`
 	
`hwdetect /add:%TOTAL% /vid:0x1014 /did:0x01BD`
`if errorlevel 0 set TOTAL=0`
`if errorlevel 1 set TOTAL=1`
`if errorlevel 2 set TOTAL=2`
`if errorlevel 3 set TOTAL=3`
`if errorlevel 4 set TOTAL=4`
`if errorlevel 5 set TOTAL=5`
`if errorlevel 6 set TOTAL=6`
 	
`echo There are %TOTAL% IBM ServeRAID adapters in this system`
	Determines the total number of IBM ServeRAID adapters in the target server, assuming there are no more than three of each type: IBM ServeRAID 4, IBM ServeRAID 5, and IBM ServeRAID 6/6i/6M

`hwdetect /i>hwdetect.out`	Displays hardware configuration information about the target server. The >hwdetect.out parameter is a Windows output-redirect option that causes the output from the hwdetect.exe utility to be saved in the specified file.

Below is an example of the hwdetect.out file that the last example might create:

```
[System]
Machine_Type=8674
Model_Number=42X
Serial_Number=78Z9506
Product_Name=eserver xSeries 330
BIOS_version=1.04
BIOS_Build_Level=EME112A
BIOS_DATE=06/28/2002
BIOS_Manufacturer=IBM
BIOS_Language=US
Number_Of_Enclosures=1
Enclosure_Type.0=23
Processor_Slots=2
Active_Processors=1
Processor_Family.0=17
Processor_Speed_MHz.0=1400
Processor_X64 = TRUE
Total_Enabled_Memory_Mb=256
ROM_Diagnostics_Build_Level=EME112A
ISMP_Build_Level=BR8T30A
RSA_Build_Level=GEE834A
System_UUID = 8030E01060F010B010605090D0A020F0
Blade_Chassis_UUID = 0F020A0D0900F00F020A0D0900F00F02
Blade_Slot = 02

[PCI]
Total_Number_Devices=10
Bus_Number.0=0
Device_Number.0=1
Function_Number.0=0
```

```
Class_Code.0=0000
Revision.0=0
Header_Type.0=0
Vendor_ID.0=5333
Device_ID.0=8A22
Subvendor_ID.0=1014
Subdevice_ID.0=01C5
Bus_Number.1=0
Device_Number.1=2
Function_Number.1=0
Class_Code.1=0000
Revision.1=0
Header_Type.1=0
Vendor_ID.1=8086
Device_ID.1=1229
Subvendor_ID.1=1014
Subdevice_ID.1=105C
```

Running "hwdetect /p" produces the same output with the exception that the section names are tacked onto the beginning of each keyword:

```
System_Machine_Type = 8674
System_Model_Number = 42X
System_Serial_Number = 78Z9506
...
PCI_Bus_Number.0 = 0
PCI_Device_Number.0 = 1
...
```

Notes:

1. The BIOS_DATE value is listed in mm/dd/yyyy format.
2. The Enclosure_Type.0=23 is based on SMBIOS 2.3 spec. 23 = Main chassis.
3. There is an entry for Processor_Family and Processor_Speed_MHz for each microprocessor in the server.
4. The ROM_Diagnostics_Build_Level is empty for servers that do not support ROM diagnostics.
5. PCI devices are listed in the order they are scanned.
6. PCI devices are listed in the *Value.n* format, where *Value* is the variable name and *n* is the nth PCI device scanned.
7. The header_type field is not available for versions of hwdetect running on Windows 32 or 64-bit operating systems.
8. The vendor, device, subvendor, and subdevice values are in hexadecimal notation.

INVRAID.EXE

The table below provides the supported RAID adapter information by PRAID. PRAID works by parsing the output of other RAID configuration utilities. To accomplish this, the utilities must be in the system search path.

Table 7. Supported RAID adapter information

Adapter	Controller type	Utility
ServeRAID 7t	ServeRAID-7t	arcconf
ServeRAID 8i	ServeRAID-8i	
ServeRAID 8k	ServeRAID-8k	
ServeRAID 8k l	ServeRAID-8k-l	
ServeRAID 8s	ServeRAID-8s	
ServeRAID B5015	ServeRAID-B5015	brcli
LSI SAS 1078 IR	LSI-SAS-1078-IR	cfggen
LSI SAS (1064/1064E/1068/1078)	LSI-SAS-RAID	
LSI SCSI (1020/1030)	LSI-SCSI-RAID	
ServeRAID BR10i	ServeRAID-BR10i	
ServeRAID BR10il	ServeRAID-BR10il	
ServeRAID 7e SATA	ServeRAID-7e-SATA	hrconf
ServeRAID 7e SCSI	ServeRAID-7e-SCSI	
ServeRAID 8e SAS	ServeRAID-8e-SAS	
ServeRAID 8e SATA	ServeRAID-8e-SATA	
ServeRAID 6M	ServeRAID-6M	ipssend

Table 7. Supported RAID adapter information (continued)

Adapter	Controller type	Utility
LSI MegaRAID 8480	LSI-MegaRAID-8480	storcli
ServeRAID C105	ServeRAID-C105	
ServeRAID C100	ServeRAID-M100	
ServeRAID C100 R5	ServeRAID-M100-R5	
ServeRAID M1xxx Series	ServeRAID-M1xxx	
ServeRAID M1xxx Series R5	ServeRAID-M1xxx_R5	
ServeRAID M5014	ServeRAID-M5014	
ServeRAID M5014 R6/R60	ServeRAID-M5014-R6-R60	
ServeRAID M5015	ServeRAID-M5015	
ServeRAID M5015 R6/R60	ServeRAID-M5015-R6-R60	
ServeRAID M5025	ServeRAID-M5025	
ServeRAID-M5025-R6-R60	ServeRAID M5025 R6/R60	
ServeRAID M51xx Series	ServeRAID-M51xx	
ServeRAID M51xx Series R5	ServeRAID-M51xx_R5	
ServeRAID M51xx Series R5/R6	ServeRAID-M51xx_R5_R6	
ServeRAID M51xx Series R6	ServeRAID-M51xx_R6	
ServeRAID MR10i	ServeRAID-MR10i	
ServeRAID MR10ie	ServeRAID-MR10ie	
ServeRAID MR10il	ServeRAID-MR10il	
ServeRAID MR10is	ServeRAID-MR10is	
ServeRAID MR10k	ServeRAID-MR10k	
ServeRAID MR10M	ServeRAID-MR10M	
ServeRAID M5210	ServeRAID-M5210	
ServeRAID M5210 R5	ServeRAID-M5210_R5	
ServeRAID H1110/H1135	SAS2004	sas2ircu

Usage

```
invraid [/I | /P] /L /F
```

Table 8. INVRAID parameters

Parameter	Description
/I	Displays information about the all host adapters in the system in an INI-file format.
/P	Dumps information about all host adapters in a system in a keyword=value format.
/L	Specifies the light version of the RAID utility.
/F	Dumps information about all host adapters in the system to a file.

Return values

Table 9. Values returned by INVRAID

Return Value	Description
0	Success
1	Syntax Error
2	Program Error

Examples

To dump the information about all RAID controllers in a system to a file in INI file format with the name `myraid.ini`, use the /I parameter as shown here:

```
invraid.exe /i /f:myraid.ini
```

Returns:

```
[System]
Machine_Type = 7977
Serial_Number = KOKN689
Total_Number_Of_Controllers = 2

[RAIDController.1]
Model = ServeRAID-8k-l
BIOSVersion = 5.2-0 (15412)
FirmwareVersion = 5.2-0 (15412)
DriverVersion = 5.2-0 (15317)
RebuildRate = HIGH
StripeSize = 256
ReadAhead = ADAPTIVE
PCI = 9005:0286:FFFF:FFFF

[RAIDController.1.Array]
Total_Number_Of_Arrays = 1
ID.1 = A
Members.1 = 1,2,3,4

[RAIDController.1.Hotspares]
Total_Number_Of_Hotspares = 0

[RAIDController.1.Logical]
Total_Number_Of_Logicals = 1
Array.1 = A
Size.1 = 279800
Raid_Level.1 = 10
WriteCache.1 = AUTO

[RAIDController.1.Physical]
Total_Number_Of_Physicals = 4
Channel.1 = 1
ID.1 = 0
Size.1 = 140013
Type.1 = SAS
Serial_Number.1 = JDX2JN8K
Channel.2 = 1
ID.2 = 1
Size.2 = 140013
Type.2 = SAS
Serial_Number.2 = Q5902T4N
Channel.3 = 1
ID.3 = 2
Size.3 = 140013
Type.3 = SAS
```

```
Serial_Number.3 = Q5902TPA
Channel.4 = 1
ID.4 = 3
Size.4 = 140013
Type.4 = SAS
Serial_Number.4 = Q5902TS8

[RAIDController.2]
Model = ServeRAID-8s
BIOSVersion = 5.2-0 (15411)
FirmwareVersion = 5.2-0 (15411)
DriverVersion = 5.2-0 (15317)
PCI = 9005:0285:1014:034D

[RAIDController.2.Array]
Total_Number_Of_Arrays = 0

[RAIDController.2.Hotspares]
Total_Number_Of_Hotspares = 0

[RAIDController.2.Logical]
Total_Number_Of_Logicals = 0

[RAIDController.2.Physical]
Total_Number_Of_Physicals = 0
```

Using the /P parameter returns the same information, but the section title from the properties file is shown for each value:

```
invraid /p > myfile.ini
```

Returns:

```
System_Machine_Type = 7977
System_Serial_Number = KOKN689

RAIDController.1.Model = ServeRAID-8k-l
RAIDController.1.BIOSVersion = 5.2-0 (15412)
RAIDController.1.FirmwareVersion = 5.2-0 (15412)
RAIDController.1.DriverVersion = 5.2-0 (15317)
RAIDController.1.RebuildRate = HIGH
RAIDController.1.StripeSize = 256
RAIDController.1.ReadAhead = ADAPTIVE
RAIDController.1.PCI = 9005:0286:FFFF:FFFF
RAIDController.1.Array.ID.1 = A
RAIDController.1.Array.Members.1 = 1,2,3,4
RAIDController.1.Logical.Array.1 = A
RAIDController.1.Logical.Size.1 = 279800
RAIDController.1.Logical.Raid_Level.1 = 10
RAIDController.1.Logical.WriteCache.1 = AUTO
RAIDController.1.Physical.Channel.1 = 1
RAIDController.1.Physical.ID.1 = 0
RAIDController.1.Physical.Size.1 = 140013
RAIDController.1.Physical.Type.1 = SAS
RAIDController.1.Physical.Serial_Number.1 = JDX2JN8K
RAIDController.1.Physical.Channel.2 = 1
RAIDController.1.Physical.ID.2 = 1
RAIDController.1.Physical.Size.2 = 140013
RAIDController.1.Physical.Type.2 = SAS
RAIDController.1.Physical.Serial_Number.2 = Q5902T4N
RAIDController.1.Physical.Channel.3 = 1
RAIDController.1.Physical.ID.3 = 2
RAIDController.1.Physical.Size.3 = 140013
RAIDController.1.Physical.Type.3 = SAS
RAIDController.1.Physical.Serial_Number.3 = Q5902TPA
RAIDController.1.Physical.Channel.4 = 1
RAIDController.1.Physical.ID.4 = 3
```

```
RAIDController.1.Physical.Size.4 = 140013
RAIDController.1.Physical.Type.4 = SAS
RAIDController.1.Physical.Serial_Number.4 = Q5902TS8
RAIDController.2.Model = ServeRAID-8s
RAIDController.2.BIOSVersion = 5.2-0 (15411)
RAIDController.2.FirmwareVersion = 5.2-0 (15411)
RAIDController.2.DriverVersion = 5.2-0 (15317)
RAIDController.2.PCI = 9005:0285:1014:034D
```

LEcho.EXE

The Logging Echo utility (LEcho.exe) performs the following functions:

- Write a message to the display
- Write a message to a log file
- Set the system errorlevel with a specific code
- Display a message to the screen while pausing or running a timer for a discreet amount of time

The ServerGuide Scripting Toolkit provides 32 and 64 bit versions of LEcho.

In order for LEcho.exe to write a message to a log file, you must set the environment variable LECHO_LOG to a fully qualified path and filename using a command similar to the following:

```
set LECHO_LOG=C:\SGTKWinPE\Lecholog.txt
```

LEcho.exe checks the number of characters on the command line against the current command line limits of 8000 characters for the Windows environment. A message is displayed if the characters exceed the limit.

Usage

The LEcho.exe utility has the following command-line syntax:

```
The LEcho.exe utility has the following command-line syntax:

LEcho [message] [/F] [/R:n]
[/E:n] [/P] [/P:n] [/T:n]
[/SC] [/SN] [/N] [/LO] [/DO] [/?]
```

Parameter	Description
message	The message to display to the screen or log file.

Parameter	Description
/F	Formats the message using the following variables: • **%d or %nd** formats the system date. The format is indicated by *n*: – 0 = Sun 12/31/2006 (Default) – 1 = Sunday, December 31, 2006 – 2 = Sun, Dec 31, 2006 – 3 = Dec 31, 2006 – 4 = 12-31-2006 – 5 = 12/31/2006 – 6 = 2006-12-31 – 7 = 2006-Dec-31 – 8 = 2006-December-31 – 9 = 20061231 • %t or %nt formats the system time. The format is indicated by *n*. – 0 = 16:12:13 (Default) – 1 = 04:12:13 PM • \a sets an alert (bell) • \b - backspace • \f - form feed • \n - newline • \r - carriage return • \t - horizontal tab
/R:*n*	Repeats the message *n* times.
/E:*n*	Displays the error message and sets the system errorlevel to *n*.
/P	Pauses until a key is pressed.
/P:*n*	Pauses for *n* seconds or until a key is pressed
/T:*n*	Initiates a timer for *n* seconds. This timer cannot be ended prematurely.
/SC	Suppresses the output of the countdown timer.
/SN	Suppresses the newline character.
/N	Creates a new, blank log file. If the log file already exists, it is overwritten.
/LO	Writes the message to the log file only.
/DO	Writes the message only to the display.
/?	Display a help message containing the application syntax.

Return codes

The LEcho.exe utility returns the following values to indicate status:

Value	Description
0	Success or true
1	Syntax error
5	Cannot write to log file.
100	False
255	Program error

Examples

The following examples illustrate Logging Echo utility usage.

Example	Description
LEcho	Sends a blank line to the display, and the log file if LECHO_LOG is set.
LEcho “My Message”	Sends the text “My Message” to the display and to the log file if LECHO_LOG is set.
LEcho /T	Displays a message indicating the current system time to the display and to the log file if LECHO_LOG is set.
LEcho “My Message” /T	Sends the text “13:55:24 – My Message” to the display and the log file if LECHO_LOG is set, 13:55:24 indicating the current system time.
LEcho “My Message” /T /DO	Sends the text “13:55:24 – My Message” to the display only. 13:55:24 indicating the current system time.
LEcho “My Message” /E:200	Sends the text “My Message” to the display and the log file if LECHO_LOG is set, and then sets the system error level to 200.
LEcho /E:155	Sets the system error level to 155. No text is displayed or logged.
LEcho “New Log File” /N /LO	Creates a new log file from LECHO_LOG. If a log file already exists, it is deleted and a new one is created. It then sends the text “New Log File” to the new log file only. No text is displayed to the screen.
LEcho “%d\t%t – My Message” /F	Sends the text “Sun 12-31-2006 16:12:13 – My Message” to the screen and log file if LECHO_LOG is set.
LEcho “Pausing for 30 seconds.” /P:30	Sends the text “Pausing for 30 seconds,” and a countdown timer beginning at 30. This countdown can be bypassed by pressing any key.
LEcho “Running a 30 second timer.” /T:30 /SC /SO	Sends the text “Running a 30 second timer.” to the screen only, and returns control to the environment after 30 seconds. No timer is displayed.

PRAID.EXE

PRAID is a scriptable executable program that offers a single user interface for both configuring and replicating all RAID controllers supported by the ServerGuide Scripting Toolkit. PRAID works with both the 32- and 64-bit versions of the Windows Preinstallation Environment.

PRAID has three modes of operation:

- **Deploy mode** – for scripted configuration of RAID controllers.
- **Capture mode** – for replicating RAID controller settings.
- **Restore-defaults mode** – for resetting RAID controllers to factory-default settings only.

Deploy mode

Used in Deploy mode, PRAID offers the following features:

- Configures all RAID controllers in a server with a single call to the program.
- Automatically resets all RAID controllers to factory-default settings before configuring.
- Uses customizable logic to decide which configuration (policy) is applied to a server based on system hardware. The logic can involve:
  - Machine type of the server
  - Serial number of the server
  - Number of drives connected to the RAID controller
  - RAID controller type
  - Controller number (order) of the RAID controller
- Can be highly customized for specific RAID configurations, or highly generalized to handle many different RAID configurations.
- Provides a default or AUTO mode for automatically creating arrays and logical drives using default settings. This mode requires no knowledge of the number, size, or location of the drives connected to the RAID controllers.
- Automatically applies default values for any RAID configuration parameters that you do not supply. You supploy only the parameters that you want to change.
- Default values for each configuration parameter are equivalent to the default settings of the ServeRAID Manager express configuration method, where applicable.
- Allows up to 50 policies for configuring RAID controllers to be specified in a single policies file.

Note:

When using PRAID in Deploy mode, the **/r** parameter is required.

To delete RAID configuration on all controllers, specify **/r**. To delete RAID configuration on a specific controller, specify **/r#** where # is the controller number.

For example, `praid /f:policiy.ini /r /y`.

Capture mode

Used in Capture mode, PRAID offers the following features:

- Captures the RAID configurations of all supported controllers to a text file, the policies file, with a common format.
- Captured RAID configurations can be immediately used with PRAID in deploy mode to easily replicate the RAID configuration to many servers.
- Allows customizable logic when saving the captured parameters to determine when each captured configuration should be deployed.
- Saves useful information about each captured configuration, including the system machine type, date, and time when the configuration was captured.
- Allows you to edit any RAID configurations that you capture before deploying them to other systems.

Restore-defaults mode

Used in Restore-defaults mode, PRAID offers the following features:

- Deletes all arrays and logical drives on all RAID controllers.
- Sets other RAID controller settings back to factory defaults.

Environment requirements

Supported RAID adapter information by PRAID is shown below. PRAID works by parsing the output of other RAID-configuration utilities. To accomplish this, the utilities must be in the system search path. Refer to Table 7 on page 62 for supported RAID adapter information.

Usage

Each of the modes supported by PRAID requires a specific syntax, but they all share some common parameters, described in Table 10.

Table 10. PRAID parameters common to multiple modes

Parameter	Description	Usage

`/r:`*n*
 	
Restore-defaults mode
	Restores the RAID controller with the controller number specified by *n* to factory-default settings and then returns immediately. No RAID configuration is done if you use this parameter.
	
	If no value is specified for the controller number, all RAID controllers are reset to factory-default settings.
	
	Used alone, the parameter provides Restore-defaults mode. You must use this parameter in conjunction with Deploy mode parameters to reset controllers to the factory default settings before deploying a new configuration.
		`praid /r`
		
		Restores all controllers to factory-default settings.
		`praid /r:3`
		
		Restores controller three to factory-default settings. No other controllers are affected.
		`PRAID /f:policies.ini /r /v:5 /e1`
		
		Configures the RAID controllers in the system using the policies file `policies.ini`, sets the verbose mode to maximum, and returns an error code if there were no matching policies for any controllers.

Table 10. PRAID parameters common to multiple modes (continued)

Parameter	Description	Usage

`/f:policies_file`
 	
Specifies the policy file
	The policy file name. This parameter is required for capture mode, and for deploy mode unless the `/d` parameter is used.
	
	In deploy mode, this points to the policies that you would like PRAID to use when configuring the RAID controllers. You cannot use this parameter with the `/d` parameter.
	
	In capture mode, this points to the file where you would like the captured configurations to be written. If the file does not exist, PRAID will create it. If the file does exist, PRAID will append to the end of it.
	
	The `/f` parameter is valid in deploy and capture modes.
		`praid /f:myfile.ini`
		
		Uses the policies file, myfile.ini, to configure all RAID controllers.
		`praid /c /f:myfile.ini`
		
		Captures the RAID configuration of all controllers to the policy file, myfile.ini.

`/y` Suppresses prompting	Suppresses the confirmation prompt. This parameter is optional. If you select the `/y` paramter, PRAID does not prompt you before resetting controllers to factory-default settings. PRAID always resets all controllers to factory-default settings before configuring them. If you do not supply this parameter, PRAID will pause to warn you before resetting the RAID controllers to factory-default settings. The `/y` parameter is valid in deploy and restore-defaults modes. This parameter is optional.	`praid /f:myfile.ini /y` Uses the policies in myfile.ini to configure the RAID controllers and does not prompt before resetting all controllers to factory-default settings.
`/e2` Error code 2 if no supported controllers found	Returns an error code of 2 if there were no supported RAID controllers found in the system. By default, PRAID does not return an error if no controllers are found in the system. This parameter is valid in all modes. This parameter is optional.	`praid /c /f:c:\myfile.ini /e2` Captures the RAID configuration of all RAID controllers to c:\myfile.ini, and returns an error if no controllers are found in the system.

Table 10. PRAID parameters common to multiple modes (continued)

Parameter	Description	Usage
/e3 Error code 3 if no supported drives found	Returns an error code of 3 if at least one controller was found with no drives attached. By default, PRAID does not return an error if no drives are attached to a RAID controller. This parameter is valid in any mode. This parameter is optional.	praid /d /e3 Configures all RAID controllers with default settings and returns an error if one or more controllers has no drives attached.
/v:n Verbose level	Sets the verbosity level, where n is: • 0 - quiet • 3 - default • 5 - maximum This parameter is valid in any mode. This parameter is optional.to 'max'.	praid /d /v:5 Configures all RAID controllers with default settings, and sets the verbose level

Deploy mode

The syntax for Deploy mode is:

```
PRAID.EXE /f:policies /r /d /p:path /e1 /e2 /e3
/v:n /y /b
```

The parameters unique to Deploy mode are described below.

Table 11. PRAID Deploy mode parameters

Parameter	Description	Usage
/d Configure with defaults	Configure all controllers in the system using default settings instead of using a policies file. The default settings used are the same as the default settings for the policies file. You cannot use this parameter with the /f parameter. See Table 15 on page 84 for the default values that will be assigned for each RAID controller based on the number of drives attached to the controller. This parameter is required unless the /f parameter is specified.	praid /d /r Configures all RAID controllers in the system using default settings.
/e1 Error if no policy found	Returns an error code of 1 if one or more controllers are not configured due to the fact that there was no policy found to configure them This parameter is optional.	praid /f:policy.ini /r /e1 Configures all RAID controllers using the policies file, policy.ini, and returns an error if no matching policy was found.

Capture mode

The syntax for Capture mode is:

```
PRAID.EXE /c[:p] /f:policies /e2 /e3 /v:n
```

The parameters unique to Capture mode are described below.

Table 12. Capture mode parameters

Parameter	Description	Usage
`/c[:p]` Capture mode	Indicates capture mode. The :p portion is optional. If you do not include the optional portion, then :p will assume the default value: "t,d". You can use :p to provide a list of parameters describing the `AppliesTo` that should be created when capturing the parameters to a policy. See "AppliesTo.*n*" on page 77. :p is a list containing any of the following: • t – Use the type of the RAID controller in the AppliesTo.1 entry for the policy. • c – Use the controller number (scan order relative to all other RAID controllers in the system) in the AppliesTo.1 entry for the policy. • d – Use the number of drives connected to the RAID controller in the AppliesTo.1 entry for the policy. **Note:** You must specify the name of the policies file using the /f parameter when using the /c parameter. The policy or policies created are appended to the end of the file if the file exists. If the file does not exist, a new file is created. If there are multiple RAID controllers in the system, their configurations are placed in the file in scan order.	`praid /c:m,t /f:myfile.ini` Captures the configuration of all RAID controllers to myfile.ini using the machine type of the server and the RAID controller type as the AppliesTo.1 entry.

Restore-defaults mode

The syntax for Restore-defaults mode is:

```
PRAID /r:n /e2 /v:n /y
```

Usage examples

Deploy mode examples

```
PRAID /r /d /y
```

- Configures all RAID controllers in the system using default settings.
- Does not prompt before setting controllers to factory-default settings.

- Performs drive synchronization without prompting, when required.

This example is useful for unattended scripted installations.

```
PRAID /f:policies.ini /r /v:5 /e1
```

- Configures the RAID controllers in the system using the policies file: policies.ini.
- Sets the verbose mode to maximum.
- Returns an error code if there were no matching policies for one or more controllers

Capture mode examples

```
PRAID /c /f:c:\mydata\policies.ini
```

Captures the configuration of all RAID controllers into the file: `C:\mydata\policies.ini`.

```
PRAID /c:m,t /f:policies.ini
```

- Captures the configuration of all RAID controllers into the file: policies.ini.
- Uses the system machine type and RAID controller type as the AppliesTo.1 entry in the policies file for each captured configuration.

Restore-defaults mode examples

```
PRAID /r /v:0 /y
```

- Restores all RAID controllers to factory-default settings.
- Operates in silent mode, no messages are printed to the screen.
- Does not prompt the user before restoring factory-default settings.

Return codes

- **0** - Success.
- **1** - Execution was successful, but the */e1* parameter was supplied and at least one controller was not configured because there was no matching policy.
- **2** - Execution was successful, but the */e2* parameter was supplied and no controllers were found in the system.
- **3** - Execution was successful, but the */e3* parameter was supplied and at least one controller was not configured because no drives were attached.
- **4** - Syntax error on the command line.
- **5** - Syntax error in the policies file or the policy file could not be opened.
- **6** - Reserved
- **7** - Error resetting a controller to the default settings.
- **8** - Error gathering information about a controller.
- **9** - Error in the policy file.
- **10** - Error during processing.
- **11** - Error during deployment.

Policies file

When used in configure mode, the policies file directs how PRAID configures the RAID controllers in a system using keywords and values that can be customized by the user. In capture mode, PRAID creates or appends to the end of a policies file the parameters that can configure other RAID controllers identically to the ones in the current system.

Use any of the following methods to create a policies file:

1. Run PRAID in capture mode to create a policies file from an already-configured RAID controller.
2. Use one of the example policies files provided with the ServerGuide Scripting Toolkit, and customize it to configure your RAID controllers.
3. Use an ASCII text editor to create a new policies file.

The policies file is an ASCII text file that is organized in INI-file format. Each INI-file section name indicates the start of a new policy for configuring RAID controllers.

The policies file must contain one or more uniquely-named sections that use the format `[Policy.name]`, where *name* is a unique user-assigned name that identifies the policy. *Name* can be any combination of letters, numbers, underscores, periods, or dashes.

Some examples of legal section names are: `[Policy.1]`, `[Policy.mypolicy]`, and `[Policy.My-RAID5-config]`. Each section in the policies file represents a single policy for configuring RAID controllers. You can have up to 50 policies in a single policies file.

How PRAID selects a policy: Each section in the policies file represents a single policy for configuring the RAID controllers. In configure mode, each RAID controller is configured using a single policy, but a single policy can be used to configure multiple controllers. Each policy in a policies file contains one or more *AppliesTo.n* entries, where *n* is the number of the **`AppliesTo`** parameter within the policy. This entry is required in each section, so every section must contain at least an AppliesTo.1 entry. See "Policies file parameters" for a full description of the AppliesTo.n entry.

These entries are followed by a list of hardware parameters including machine type, number of drives connected to the RAID controller, and scan order, that are evaluated against the current system hardware. If all of the hardware parameters of an AppliesTo.n entry match the hardware being evaluated, this policy is used to configure the hardware. For each policy in the policies file, the AppliesTo.n entries for that policy are evaluated in order starting with AppliesTo.1. If none of the AppliesTo.n entries match the current hardware then the policy is not applied and the AppliesTo.n entries in the next policy are evaluated. This continues until either a match is found or no more policies exist in the file. If the end of the file is reached without a match then the controller is not configured. Because the policies are evaluated in order, you should place more specific policies at the beginning of the policies file.

Policies file parameters: This section describes the parameters used in the policies file. The Policy.*name* header and `AppliesTo.1` entry are the only parameters required. All values are case-insensitive.

If you do not specify a value for any of the other parameters, they will be assigned a default value when applicable. If a parameter is not valid for a RAID controller, it will be ignored.

In addition to this reference, the ServerGuide Scripting Toolkit also provides two example policies files that you can modify for your own use.

- RAID1-5.ini Creates a RAID-1 array using the first two drives, and a RAID-5 array using the remaining drives. Valid for ServeRAID-4H, 4Mx, 4Lx, 5i, 6i, 6M, 6i+, 7k, 8i

- RAID5HSP.ini Creates a single RAID-5 array with a single hot-spare drive using all available drives. Valid for ServeRAID-4H, 4Mx, 4Lx, 5i, 6i, 6i+, 6M, 7k, 7t, 8i.
- template.ini Provides a policies file template containing all parameters with details about each parameter.
- syntax.txt Provides a syntax specification for the polices file.

Table 13. Policy file parameters

Keyword	Required?	Default	Description
`Policy.name`	Yes	None	This header designates the start of a new policy. See "Policy.*name*" on page 77 for additional information.
`AppliesTo.n`	Yes	None	Use this parameter to describe when the current policy should be chosen to configure the RAID controllers. See "AppliesTo.*n*" on page 77 for additional information.
`ReadAhead`	No	• ADAPTIVE (for ServeRAID 4H, 4MX, 4Lx, 5i, 6i, 6i+, 6M, and 7k) • ON (for ServeRAID-7t and 8i, 8k, and 8k-l)	Specifies the read ahead setting that should be applied to the RAID controller. See "ReadAhead" on page 78 for additional information.
`RebuildRate`	No	HIGH	Specifies the rebuild rate that should be applied to the RAID controller. See "RebuildRate" on page 78 for additional information.
`StripeSize`	No	• 8 (for ServeRAID 4H, 4Mx, 4Lx, 5i, 6i, 6i+, 6M, and 7k) • 64 (for ServeRAID-7t, 8i, 8k, 8k-l, 7e-SCSI, 7e-SATA, 8e-SATA, 8e-SAS, and LIS-IDEal-RAID)	Specifies the stripe-unit size in KB that the controller should use for its arrays. See "StripeSize" on page 79 for additional information.
`Array_Mode`	No	AUTO	Defines the array-creation policy to use when selecting physical disk drives to include in an array. See "Array_Mode" on page 79 for additional information.

Table 13. Policy file parameters (continued)

Keyword	Required?	Default	Description
`Array_Defaults`	No	• 0%:1 for ServeRAID-8e-SATA and 8e-SAS, LSI-SCSI-RAID when at least 3 drives are available • 0%:1 for ServeRAID-4H, 4Mx, 4Lx, 5i, 6i, 6i+, 6M, and 7k, when one or more arrays has 4 or more physical drives • 0%:0 for all other cases	Defines the default values to use for the variance and number of hot-spare drives when `AUTO` is specified for `Array_Mode`. See "Array_Defaults" on page 79 for additional information.
`Array.`*`letter`*	No	None	Lets you specify exactly how many arrays are created and the exact physical drives that you would like in each array. See "Array.*letter*" on page 80 for additional information.
`Hotspares`	No	None	Defines a list of specific physical drives to designate as hot-spare drives. See "Hotspares" on page 81 for additional information.
`Logical_Mode`	No	AUTO	Defines the logical-drive creation policy to use when creating logical drives. See "Logical_Mode" on page 81 for additional information.
`Logical_Defaults`	No	FILL:AUTO:AUTO	Defines the default logical drive settings that should be used when creating logical drives. See "Logical_Defaults" on page 81 for additional information.
`Logical.`*`num`*	No	None	Lets you specify how many logical drives are created and the specific parameters for each logical drive. See "Logical.*num*" on page 82 for additional information.

Policy.name:
Description

The policy.*name* header designates the start of a new policy. You can specify *name* using any combination of letters, numbers, underscores, periods, or dashes. There is no maximum length for *name*, but the maximum length for a single line in the policies file is 256 characters. You can have up to 50 policies in a single policies file.

Example

```
[Policy.RAID-5-Hotspare]
```

AppliesTo.n:

Description

Use this parameter to describe when the current policy should be chosen to configure the RAID controllers. You can define up to 20 AppliesTo.n entries per policy. You must have an AppliesTo.1 entry for each policy, and AppliesTo.n is the only required parameter of a policy.

AppliesTo.n includes a comma delimited list containing one or more of the following parameters:

- m:*mtype*, where *mtype* is the four digit machine type of an IBM eServer or xSeries server.
- s:*serial*, where *serial* is the serial number of an IBM eServer or xSeries server.
- c:*contn*, where *contn* is the controller number (scan order) of the RAID controller with respect to all other RAID controllers in the system.

  The number assigned to a particular controller is dependent on the controller's physical PCI slot and the order in which your system scans its PCI slots.
- t:*ctype*, where *ctype* is the type of the controller. The type is not case-sensitive, and must be one of the controller types listed in the table of RAID adapters supported by PRAID.
- d:*drives*, where *drives* is an integer value specifying the number of drives connected to the controller. Only drives in a **Ready** state after resetting the controller to factory-default settings are counted.
- ALL. Indicates that this policy should be used for all RAID controllers. This parameter is good to use if you declare a default policy that is not covered by any of the other policies.

Examples

Example using the m,s,c,t, and d parameters:

```
AppliesTo.1 = m:8865,t:ServeRAID-7k
AppliesTo.2 = c:1,d:15,s:87R478U
```

Example using the ALL parameter:

```
AppliesTo.1 = ALL
```

ReadAhead:
Description

The **ReadAhead** parameter specifies the read ahead setting that should be applied to the RAID controller. If this parameter is not applicable for a RAID controller, it is ignored. See “Supported settings for RAID controllers” on page 82 for the list of ReadAhead settings supported by PRAID for each RAID controller. Possible settings are:

- `Adaptive`
- `On`
- `Off`

Example

```
ReadAhead = On
```

RebuildRate:

Description

The **RebuildRate** parameter specifies the rebuild rate that should be applied to the RAID controller. If this parameter is not applicable for a RAID controller, then it will be ignored. See "Supported settings for RAID controllers" on page 82 for the list of RebuildRate settings supported by PRAID for each RAID controller.

- `High`
- `Medium`
- `Low`

Example

```
RebuildRate = High
```

StripeSize:
Description

Specifies the stripe-unit size in KB that the controller should use for its arrays. If this parameter is not applicable for a RAID controller, then it will be ignored. See "Supported settings for RAID controllers" on page 82 for the list of StripeSize settings supported by PRAID for each RAID controller. Possible values are any stripe size supported by the controller.

Examples

```
StripeSize = 32
```

Array_Mode:
Description

Defines the array-creation policy to use when selecting physical disk drives to include in an array. Possible values are:

Auto Creates arrays using drives that have the same size in MB. This is the default. Each set of drives with same size on will be combined into a single array. The maximum number of drives allowed per array is determined by the limits of the RAID controller. Only drives in a **Ready** state after resetting the controller to factory-default settings are used in arrays. Hot-spare drives are created based on the rules supplied with the `Array_Defaults` parameter.

The `Array_Defaults` parameter allows you to modify the default behavior of the AUTO mode for arrays.

Custom Allows you to specify the exact physical disk drives to use in the array. If you specify this value, you must also specify the `Array.letter` parameter with a list of drives for each array that you want to create. If you want hot-spare drives to be created, you must use the `Hotspares` parameter to list the hot-spare drives.

Examples

```
Array_mode = CUSTOM
```

Array_Defaults:

Description

Defines the default values to use for the variance and number of hot-spare drives when `AUTO` is specified for `Array_Mode`. This parameter is not valid if `Array_Mode` is set to `CUSTOM`.

The value of `Array_Defaults` is expressed in the format: *`variance:hotspares`*, where:

variance specifies the percentage variance to use when selecting drives to add to the array. This parameter is useful when you are using drives that may vary slightly in size. Variance is based on a percentage of the drive s size in MB. Valid values are:

- `0%` - Only drives with equal size in MB will be combined into a single array.
- `5%` - All drives within 5% size in MB will be combined into a single array.
- `10%` - All drives within 10% size in MB will be combined into a single array.
- `100%` - All drives, regardless of size in MB, will be combined into a single array.

and

`hotspares` is an integer that specifies the total number of hot-spare drives to create. The largest drives are chosen as hot-spare drives first. If not enough drives are available to create hot-spare drives, then PRAID will not create any hot-spare drives.

Examples

```
Array_Defaults = 5%:1
```

Array.letter:
Description

Lets you specify exactly how many arrays are created and the exact physical drives that you would like in each array. You can specify the physical drives using any of the following methods:

- The channel number and SCSI ID (for SCSI) or bus number and target ID (for SATA/SAS) of each drive. The channel number or bus number is always 1-based. The SCSI ID or target ID is always 0-based.
- A list of integer values indicating that the *n*th drive should be included in the array
- The keyword `ALL` to indicate that all remaining drives attached to the controller that are not specified in previous arrays should be included in the current array.

The first array must be labeled `Array.A`. Additional arrays are labeled sequentially, `Array.B`, `Array.C`, and so on. The maximum number of arrays allowed per controller is determined by the limits of the specific RAID controller.

Examples

Example using channel number and SCSI ID:

```
Array.A = 1:1,1:2
Array.B = 1:3,1:4,1:5,2:1,2:2,2:3,2:4,2:5,2:6
Array.C = ALL
```

Example using integer values:

```
Array.A = 1,2,3
Array.B = ALL
```

Hotspares:
Description

Defines a list of specific physical drives to designate as hot-spare drives. You can specify the physical drives using any one of these methods:

- The channel number and SCSI ID (for SCSI) or bus number and target ID (for SATA/SAS) of each drive. The channel number or bus number is always 1-based. The SCSI ID or target ID is always 0-based.
- A list of integer values indicating that the *n*th drive should be included in the array
- The keyword `ALL` to indicate that all remaining drives attached to the controller that are not specified in previous arrays should be included in the current array.

Examples

Example using channel number and SCSI ID:

```
Hotspares = 1:12,2:14
```

Example using integer value:

```
Hotspares = 12, 13
```

Logical_Mode:
Description

Defines the logical-drive creation policy to use when creating logical drives. Possible values are:

`AUTO` Indicates that defaults should be used for all parameters. Default parameters are:
- One logical drive is created on each array using all available space.
- The RAID level is set using the AUTO (default) scheme
- Write-cache mode is set using the default value for the controller.

You can adjust these default values using the `Logical_Defaults` parameter.

`CUSTOM` Indicates that you want to specify all of the parameters for each logical drive that is created. If you specify `CUSTOM`, then you must specify the parameters for each logical drive using the `Logical.`*`num`* parameter.

Examples

```
Logical_Mode = CUSTOM
```

Logical_Defaults:
Description

Defines the default logical drive settings that should be used when creating logical drives. This parameter is only valid when `AUTO` is specified for `Logical_Mode`. Values for this parameter are expressed in the format: *`size:raidlevel:writecmode`*, where:

Size specifies the size of each logical drive. One logical drive will be created on each array using the given size. *Size* can be in any of the following formats:

- A positive integer – specifies the size in MB.

- A percentage – specifies that a percentage of the total space should be used.
- FILL – indicates that all available space on the array should be used.

Raidlevel specifies the RAID level for the logical drive. See "Supported settings for RAID controllers" for the list of RAID level settings supported by PRAID for each controller.

Writecmode is an optional parameter that specifies the write-cache mode for each logical drive. If the write-cache mode cannot be set for a specific configuration, then this parameter will be ignored. See "Supported settings for RAID controllers" for the list of write_cache mode settings supported by PRAID for each RAID controller.

Valid values are:

- ON
- OFF
- AUTO uses the default write-cache mode for the controller. (Recommended for most users.) This is the default value if writecmode is not specified.

Examples

```
Logical_Defaults = 50%:5EE:AUTO
```

Logical.num:
Description

Lets you specify how many logical drives are created and the specific parameters for each logical drive. You can set the array letter where the logical drive is located, logical drive size, RAID level, and write-caching mode for each logical drive. The first logical drive must be labeled Logical.1. Additional logical drives are numbered Logical.2, Logical.3, and so on. You must specify at least one logical drive for each array. The maximum number of drives allowed per array and the maximum total number of logical drives allowed is determined by the specific RAID controller.

Values for this parameter are expressed in the format:
array:size:raidlevel:writecmode where *array* specifies the array letter, and *size*, *raidlevel*, and *writecmode* are as described in "Logical_Defaults" on page 81.

Examples

```
Logical.1 = A:50%:0
Logical.2 = A:50%:5EE
Logical.3 = B:FILL:1:ON
Logical.4 = C:4096:AUTO:AUTO
```

Supported settings for RAID controllers: Table 14 lists the supported settings for each RAID controller when using PRAID.

In some cases, the list of supported settings when using PRAID might differ from the supported settings of the RAID controller. These known cases are indicated in the table.

Table 14. Supported settings for each RAID controller when using PRAID. Bold settings are defaults.

RAID adapters	Read policy	Write policy	RAID Levels[1]	Stripe Size (KB)
ServeRAID-B5015	• ON • **OFF**	[n/a]	R1, R5	4, 8, 16, 32, 64, **128**, 256, 512, 1024

Table 14. Supported settings for each RAID controller when using PRAID (continued). Bold settings are defaults.

RAID adapters	Read policy	Write policy	RAID Levels[1]	Stripe Size (KB)
LSI-IDEal-RAID	[n/a]	[n/a]	R0, R1	32, **64**, 128, 256, 512, 1024, 2048, 4096
LSI-MegaRAID-8480	[n/a]	[n/a]	R0, R1, R10, R5, R50	4, 8, 16, 32, **64**, 128
LSI-SAS-1078-IR	[n/a]	[n/a]	R0, R1	[n/a]
LSI-SAS-RAID	[n/a]	[n/a]	R0, R1, R1E	[n/a]
LSI-SCSI-RAID	[n/a]	[n/a]	R1	[n/a]
ServeRAID-7t	• ON • OFF • AUTO	• ON • OFF	RVOLUME, R0, R1, R10, R5	16, 32, **64**
ServeRAID-8i	• ON • OFF • AUTO	• ON • OFF	RVOLUME, R0, R1, R10, R1E, R5, R50, R5EE, R6, R60	16, 32, 64, 128, **256**, 512, 1024
ServeRAID-8k	• ON • OFF • AUTO	• ON • OFF	RVOLUME, R0, R1, R10, R1E, R5, R6	16, 32, 64, 128, **256**, 512, 1024
ServeRAID-8k-l	• ON • OFF • AUTO	• ON • OFF	RVOLUME, R0, R1, R10	16, 32, 64, 128, **256**, 512, 1024
ServeRAID-8s	• ON • OFF • AUTO	• ON • OFF	RVOLUME, R0, R1, R10, R1E, R5, R50, R6	16, 32, 64, 128, **256**, 512, 1024
ServeRAID-BR10ie	[n/a]	[n/a]	R0, R1, R1E	[n/a]
ServeRAID-BR10il	[n/a]	[n/a]	R0, R1, R1E	[n/a]
ServeRAID-M1015	[n/a]	[n/a]	R0, R1, R10	8, 16, 32, **64**
ServeRAID-M1015–R5	[n/a]	[n/a]	R0, R1, R10, R5, R50	8, 16, 32, **64**
ServeRAID-M1xxx	[n/a]	[n/a]	R0, R1, R10	8, 16, 32, **64**
ServeRAID-M1xxx_R5	[n/a]	[n/a]	R0, R1, R10, R5, R50	8, 16, 32, **64**
ServeRAID-M5014	• ON • OFF • **AUTO**	• **ON** • OFF	R0, R1, R10, R5, R50	8, 16, 32, **64**, 128
ServeRAID-M5014-R6-R60	• ON • OFF • AUTO	• **ON** • OFF	R0, R1, R10, R5, R50, R6, R60	8, 16, 32, **64**, 128
ServeRAID-M5015	• ON • OFF • **AUTO**	• **ON** • OFF	R0, R1, R10, R5, R50	8, 16, 32, **64**, 128
ServeRAID-M5015-R6-R60	• ON • OFF • **AUTO**	• **ON** • OFF	R0, R1, R10, R5, R50, R6, R60	8, 16, 32, **64**, 128
ServeRAID-M5025	• ON • OFF • **AUTO**	• **ON** • OFF	R0, R1, R10, R5, R50	8, 16, 32, **64**, 128, 256, 512, 1024
ServeRAID-M5025-R6-R60	• ON • OFF • **AUTO**	• **ON** • OFF	R0, R1, R10, R5, R50, R6, R60	8, 16, 32, **64**, 128, 256, 512, 1024
ServeRAID-M5xxx	• ON • OFF • **AUTO**	• **ON** • OFF	R0, R1, R10, R5, R50, R6, R60	8, 16, 32, **64**, 128, 256, 512, 1024
ServeRAID-M51xx	• ON • OFF • **AUTO**	• **ON** • OFF	R0, R1, R10	8, 16, 32, **64**, 128, 256, 512, 1024
ServeRAID-M51xx_R5	• ON • OFF • **AUTO**	• **ON** • OFF	R0, R1, R10, R5, R50	8, 16, 32, **64**, 128, 256, 512, 1024

Table 14. Supported settings for each RAID controller when using PRAID (continued). Bold settings are defaults.

RAID adapters	Read policy	Write policy	RAID Levels[1]	Stripe Size (KB)
ServeRAID-M51xx_R6	• ON • OFF • **AUTO**	• **ON** • OFF	R0, R1, R10, R6, R60	8, 16, 32, **64**, 128, 256, 512, 1024
ServeRAID-M51xx_R5_R6	• ON • OFF • **AUTO**	• **ON** • OFF	R0, R1, R10, R5, R50, R6, R60	8, 16, 32, **64**, 128, 256, 512, 1024
ServeRAID-M5210	• ON • OFF • **AUTO**	• **ON** • OFF	R0,R1,R10	8,16.32,**64**
ServeRAID-M5210-R5	• ON • OFF • **AUTO**	• **ON** • OFF	R0,R1,R10,R5,R50,R6,R60	8,16.32,**64**
ServeRAID-MR10i	• ON • OFF • **AUTO**	• **ON** • OFF	R0, R1, R10, R5, R50, R6, R60	8, 16, 32, **64**, 128
ServeRAID-MR10il	• ON • OFF • **AUTO**	• **ON** • OFF	R0, R1, R10, R5, R50, R6, R60	8, 16, 32, **64**, 128
ServeRAID-MR10is	• ON • OFF • **AUTO**	• **ON** • OFF	R0, R1, R10, R5, R50, R6, R60	8, 16, 32, **64**, 128, 256, 512, 1024
ServeRAID-MR10k	• ON • OFF • **AUTO**	• **ON** • OFF	R0, R1, R10, R5, R50, R6, R60	16, 32, **64**, 128, 256, 512, 1024
ServeRAID-MR10M	• ON • OFF • **AUTO**	• **ON** • OFF	R0, R1, R10, R5, R50, R6, R60	8, 16, 32, **64**, 128
ServeRAID-C100	[n/a]	[n/a]	R0, R1, R10	**64**
ServeRAID-C100-R5	[n/a]	[n/a]	R0, R1, R10, R5	**64**
ServeRAID-C105	[n/a]	[n/a]	R0, R1,R10	**64**
SAS2004	[n/a]	[n/a]	R0, R1, R10, R1E	[n/a]

1. RAID levels 5E and 5EE support only one logical drive per array.

Default RAID levels are described in "Default RAID levels."

Default RAID levels: The default RAID level that is applied to a logical drive depends on the number of drives in the array and the controller type. These default values are designed to match the default values of the express configuration method in ServeRAID Manager where applicable. The following table shows the default RAID values that PRAID will use when `AUTO` is specified for *raidlevel*.

Table 15. Default RAID levels

	Drives in array				
Controller	**1**	**2**	**3**	**4**	**5 or more**
ServeRAID-B5015	[n/a]	RAID 1	RAID 5	RAID 5+Hotspare	RAID 5+Hotspare
LSI-IDEal-RAID	[n/a]	RAID 1	[n/a]	[n/a]	[n/a]
LSI-MegaRAID-8480	RAID 0	RAID 0	RAID 0	RAID 0+Hotspare	RAID 0+Hotspare
LSI-SAS-1078-IR	[n/a]	RAID 1	RAID 1+Hotspare	RAID 1+Hotspare	RAID 1+Hotspare

Table 15. Default RAID levels (continued)

	Drives in array				
Controller	**1**	**2**	**3**	**4**	**5 or more**
LSI-SAS-RAID	[n/a]	RAID 1	RAID 1E+Hotspare	RAID 1E+Hotspare	RAID 1E+Hotspare
LSI-SCSI-RAID	[n/a]	RAID 1	RAID 1+Hotspare	RAID 1+Hotspare	RAID 1+Hotspare
ServeRAID-7t	RAID 0	RAID 1	RAID 5	RAID 5+Hotspare	RAID 5+Hotspare
ServeRAID-8i	VOLUME	RAID 1	RAID 5	RAID 5+Hotspare	RAID 5+Hotspare
ServeRAID-8k	VOLUME	RAID 1	RAID 5	RAID 5+Hotspare	RAID 5+Hotspare
ServeRAID-8k-l	VOLUME	RAID 1	RAID 1+Hotspare	RAID 10	RAID 10+Hotspare
ServeRAID-8s	VOLUME	RAID 1	RAID 5	RAID 5+Hotspare	RAID 5+Hotspare
ServeRAID-BR10ie	[n/a]	RAID 1	RAID IE	RAID IE + Hotspare	RAID IE + Hotspare
ServeRAID-BR10il	[n/a]	RAID 1	RAID IE	RAID IE + Hotspare	RAID IE + Hotspare
ServeRAID-M1015	RAID 0	RAID 0	RAID 0	RAID 0 + Hotspare	RAID 0 + Hotspare
ServeRAID-M1015–R5	RAID 0	RAID 0	RAID 0	RAID 0 + Hotspare	RAID 0 + Hotspare
ServeRAID-M1xxx	RAID 0	RAID 0	RAID 0	RAID 0 + Hotspare	RAID 0 + Hotspare
ServeRAID-M1xxx_R5	RAID 0	RAID 0	RAID 0	RAID 0 + Hotspare	RAID 0 + Hotspare
ServeRAID-M5014	RAID 0	RAID 0	RAID 0	RAID 0 + Hotspare	RAID 0 + Hotspare
ServeRAID-M5014-R6-R60	RAID 0	RAID 0	RAID 0	RAID 0 + Hotspare	RAID 0 + Hotspare
ServeRAID-M5015	RAID 0	RAID 0	RAID 0	RAID 0 + Hotspare	RAID 0 + Hotspare
ServeRAID-M5015-R6-R60	RAID 0	RAID 0	RAID 0	RAID 0 + Hotspare	RAID 0 + Hotspare
ServeRAID-M5025	RAID 0	RAID 0	RAID 0	RAID 0 + Hotspare	RAID 0 + Hotspare
ServeRAID-M5025-R6-R60	RAID 0	RAID 0	RAID 0	RAID 0 + Hotspare	RAID 0 + Hotspare
ServeRAID-M5xxx	RAID 0	RAID 0	RAID 0	RAID 0 + Hotspare	RAID 0 + Hotspare
ServeRAID-M51xx	RAID 0	RAID 0	RAID 0	RAID 0 + Hotspare	RAID 0 + Hotspare
ServeRAID-M51xx_R5	RAID 0	RAID 0	RAID 0	RAID 0 + Hotspare	RAID 0 + Hotspare
ServeRAID-M51xx_R6	RAID 0	RAID 0	RAID 0	RAID 0 + Hotspare	RAID 0 + Hotspare
ServeRAID-M51xx_R5_R6	RAID 0	RAID 0	RAID 0	RAID 0 + Hotspare	RAID 0 + Hotspare
ServeRAID-M5210	RAID 0	RAID 0	RAID 0	RAID 0 + Hotspare	RAID 0 + Hotspare
ServeRAID-M5210-R5	RAID 0	RAID 0	RAID 0	RAID 0 + Hotspare	RAID 0 + Hotspare
ServeRAID-MR10i	RAID 0	RAID 0	RAID 0	RAID 0+Hotspare	RAID 0+Hotspare
ServeRAID-MR10il	RAID 0	RAID 0	RAID 0	RAID 0+Hotspare	RAID 0+Hotspare
ServeRAID-MR10is	RAID 0	RAID 0	RAID 0	RAID 0+Hotspare	RAID 0+Hotspare
ServeRAID-MR10k	RAID 0	RAID 0	RAID 0	RAID 0+Hotspare	RAID 0+Hotspare
ServeRAID-MR10M	RAID 0	RAID 0	RAID 0	RAID 0+Hotspare	RAID 0+Hotspare
ServeRAID-C100	RAID 0	RAID 0	RAID 0	RAID 0+Hotspare	RAID 0+Hotspare
ServeRAID-C100-R5	RAID 0	RAID 0	RAID 0	RAID 0+Hotspare	RAID 0+Hotspare

Table 15. Default RAID levels (continued)

	Drives in array				
Controller	**1**	**2**	**3**	**4**	**5 or more**
ServeRAID-C105	RAID 0	RAID 0	RAID 0	RAID 0+Hotspare	RAID 0+Hotspare
SAS2004	[n/a]	RAID 0	RAID 0	RAID 10	RAID 10

SAVESTAT.CMD

The savestat utility allows you to store and retrieve up to twenty-one values to persistent storage. You can use **savestat.cmd** to return to your place in an installation, even when a reboot is required. This utility is designed to return values that set the *errorlevel* environment variable so that you can branch in a script or batch file based on the result of the utility's execution.

The utility runs in 32- and 64–bit versions of Windows Preinstallation Environment 2.1/3.0, Windows Server 2003, and Windows Server 2008.

Savestat.cmd uses the persistent storage capability of **ASU**. Therefore the following files must be available in order for the script to work:

- ASU.EXE
- device.cat
- ibm_mdis_server_os.inf
- savestat.vbs script
- savestat.def

Usage

The savestat utility that comes with the ServerGuide Scripting Toolkit has the following command-line syntax:

```
SAVESTAT [/q] /reset
SAVESTAT [/q] /set1=value [.../set2=value ... /set21=value]
SAVESTAT [/q] /getn
SAVESTAT [/q] /validate
SAVESTAT [/q] /signature
```

Parameter	Description	Usage
/set*n*=*value*	Saves an integer value, *value*, to the th location in persistent-storage memory, where *n* is an integer from 1–21 Return codes: • **0** if successful • **1** if not successful	savestat /set*n*=*value* Where: • *n* is an integer from 1–21 • *value* is an integer from 0–254
/get*n*	Retrieves the value currently set in the *n*th location in persistent-storage memory, where *n* is an integer from 1-21. Return codes: • The value stored at the location specified by *n*, if successful. • 255 if not successful.	savestat /get*n* Where *n* is an integer from 1–21

Parameter	Description	Usage
/reset	Resets all persistent-storage memory to zero values. Return codes: • **0** if successful • **1** if not successful	savestat /reset
/signature	Verifies that the persistent storage contains the **savestat** signature. Return codes: • **0** if storage contains the signature • **1** if storage does not contain the signature	savestat /signature
/validate	Verifies that the system is supported by **savestat**. Return codes: • **0** if the system is supported • **1** if the system is not supported	savestat /validate
/q	Invokes the quiet mode. This parameter is optional and can be used with any other **savestat** parameter.	savestat /q /set1=100

Examples

The following examples illustrate savestat utility usage.

Example	Description
savestat /set2=100	Stores the value 100 in the second persistent-storage memory location
savestat /get2 if errorlevel 100 goto end if errorlevel 1 goto level1 :level1 call level1.bat :end	Retrieves the value of the second persistent-storage memory location and branches in the batch file according to the value returned

TKSEARCH.EXE

The Toolkit Search utility (tksearch.exe) can perform the following functions:

- Search for the latest ServerGuide driver sets starting from a given path.
- Search for the latest ServerGuide driver set or sets that supports one or more specified machine types.
- Search for the latest ServerGuide driver set or sets that supports a specified Network Operating System.
- Determine the architecture and type of a Network Operating System.

Two versions of the Toolkit search utility come with the ServerGuide Scripting Toolkit:

- A 32-Bit version for Windows 32-Bit Operating Systems and for Windows Preinstallation Environment (Windows PE) 2.1/3.0 (32-Bit).
- An x64 version for Windows x64 Operating Systems and for Windows Preinstallation Environment (Windows PE) 2.1/3.0 (x64).

Usage

The syntax of the tksearch.exe command is:

```
tksearch driverpath [/W:n | /WP:nospath | WP:prodspec.ini] [/M:machtype/platform_ID] [/B:filename] [/?]
```

Parameter	Description
driverpath	The fully qualified directory path to start searching for driver sets. For example: `F:\sgdeploy\drvs`.
/W:*n*	The Windows version to prefer when searching for device drivers: • 0 = Windows 2000 Server • 1 = Windows 2003 Server (Default) • 2 = Windows 2000 Professional • 3 = Windows XP • 4 = Windows 2003 Server x64
/WP:*nospath*	The fully qualified path to directory to start searching for Windows prodspec.ini. For example, `F:\sgdeploy\os\w23_std`
/WP:prodspec.ini	The fully qualified path to Windows prodspec.ini file. For example, `F:\sgdeploy\os\w23_std\i386\prodspec.ini`
/I:*machinetype/platform_ID*	The machine types or platform IDs to limit the searching of drivers for. Multiple machine types or platform IDs can be specified using a comma as the delimiter, for example, /I:8853,8854.
/B:*filename*	The name of the batch file in which to place the resulted environment variables. The default is `.\DrvSet.bat`. Enviroment Variables are: • TK_NOS_Type - (Win2000,Win2003) • TK_NOS_Arch - (I386,AMD64) • TK_NOS_Arch_Type - (x86,x64) • TK_NOS_DDL_Type - (Win2000Server, Win2003Server, Win2003Serverx64) • TK_NOS_DDL_Path_# - (number of drivers sets found) • TK_NOS_DDL_Path_1 - (corresponding drivers set) • TK_NOS_DDL_Path_2 -(corresponding drivers set) TK_NOS_DDL_Path_3 (corresponding drivers set)

Return codes

The tksearch.exe utility returns the following values to indicate status:

Value	Indicates
0	Success: One or more driver libraries found.
1	Success: Zero driver libraries found.
2	Error: Command-line syntax error.
3	Error: Writing output batch file.
4	Error: Driver path not found.
5	Error: NOS not found in specified path.
6	Error: General application error.

Examples

The following are examples of Device Driver Search utility usage:

Example	Description
tksearch f:\sgdeploy\drvs	Search all ServerGuide drivers sets regarless of machine type or NOS type.
tksearch f:\sgdeploy\drvs /W:4	Search all ServerGuide drivers sets for Windows 2003 x64 regardless of machine type.
tksearch f:\sgdeploy\drvs /WP:f:\w2000\i386\prodspec.ini	Search all ServerGuide drivers sets for Windows 2000 Server regardless of machine type.
tksearch f:\sgdeploy\drvs /M:8853,7978	Search all ServerGuide drivers sets for systems with machine type 8853 and 7978 regardless the NOS type.
tksearch f:\sgdeploy\drvs /M:8853 /W:f:\sgdeploy\os\w23_std	Search all ServerGuide drivers sets for systems with machine type 8853 and Windows 2003 Standard.
tksearch f:\sgdeploy\drvs /M:8853,7978 /B:	Search all ServerGuide drivers sets for systems with machine type 8853 and 7978 regardless the NOS type. The search results are saved as environment variables in Batch file drvset.bat.
tksearch f:\sgdeploy\drvs /M:8853 /WP:F:\sgdeploy\os\w23_ee /B:drivers.bat	Search all ServerGuide drivers sets for systems with machine type 8853 and Windows 2003 The search results are saved as environment variables in Batch file drivers.bat.

UNATTEND.EXE

The unattend.exe utility adds device-driver specific information to the Microsoft Windows unattended installation answer file for a deployment scenario. The ServerGuide Scripting Toolkit uses the unattend.exe utility to dynamically add server-specific device-driver information to the answer file for an unattended installation. The device drivers on the *IBM ServerGuide Setup and Installation* CD are already configured for use with this utility.

Two versions of the utility come with the ServerGuide Scripting Toolkit:

- A 32-bit version for Windows 32-bit operating systems and for the Windows Preinstallation Environment (Windows PE) 2.1/3.0 (32-bit)
- A 64-bit version for Windows x64 operating systems and for Windows PE 2.1/3.0 (x64).

The unattend.exe utility processes three types of device drivers:

- Text mode device drivers
- Plug-and-play device drivers
- Executable device drivers

The unattend command adds the device-driver information to the answer file in one or more locations, depending on the type of device driver:

Text mode
Text mode device-driver information is added to the [MassStorageDevices] and [OemBootFiles] sections. Entries are not duplicated; existing entries are not changed.

Hardware abstraction layer (HAL)
HAL device-driver information is assigned to the ComputerType keyword in the [Unattended] section. Any value previously assigned to this keyword is overwritten.

Plug and Play
The OemPnPDriversPath keyword in the [Unattended] section is set to the path to the PnP device-driver directory. Any value previously assigned to this keyword is overwritten.

Executable
The executable device-driver information is added to the [GUIRunOnce] section. Existing entries are not changed.

Path to the oem directory
The OemFilesPath keyword in the [Unattended] section is set to the path to the oem directory. Any value previously assigned to this keyword is overwritten.

For text mode device drivers, the unattend.exe utility uses the information in the hwdetect.ini file and the txtsetup.oem file (located in the \oem\textmode directory of the target server) to add the text mode device driver information to the answer file. This utility also adds the Microsoft retail text mode device drivers using information specified in the txtsetup.sif file from the i386 directory of the Windows operating-system installation source files.

For plug-and-play and executable device drivers, the unattend.exe utility uses information from the hwdetect.ini file (the output of the hwdetect.exe utility) and the drvinfo.ini file in each device-driver directory in the target server to determine the device drivers to add to the answer file. Device drivers that are not supported on the target server are deleted from the device-drivers directory.

Usage

The unattend.exe utility has the following command-line syntax:

```
unattend [/?] <file_name|/U:file /D:path /H:file /I:path> [/U:file] [/D:path]
[/H:file_name] [/I:path] [/S:drive] [/C] [/T] [/P] [/E] [/V:n]
```

Parameter	Description
/?	Displays all parameters
file_name	Specifies a fully qualified path and file name for the unattend.ini file that contains command-line parameters for the unattend.exe utility. You can put parameters in this file instead of typing them all on the command line, which is useful for long command lines that exceed the 127-character limit. 1. Any settings for the /U, /D, /H, or /I parameters you place on the command line will override settings in the unattend.ini file. 2. If you do not specify a setting for *filename*, you must specify the /U, /D, /H, and /I parameters on the command line.
/U:*file*	Specifies a fully qualified path and file name for the answer file
/D:*path*	Specifies a fully qualified path to the device-drivers directory in the target server. If *path* does not include oem in the path, you must use the /T parameter to process text mode device drivers.
/H:*file*	Specifies a fully qualified path and file name for the hwdetect.ini file that was created by the hwdetect.exe utility
/I:*path*	Specifies a fully qualified path to the i386 directory in the target server.
/S:*drive*	Specifies the drive letter on the target server to which the operating system is being installed
/C	Creates a default unattend.ini file
/T	Causes the unattend.exe utility to add only the text mode device-driver information to the answer file
/P	Causes the unattend.exe utility to add only the plug-and-play device-driver information to the answer file
/E	Causes the unattend.exe utility to add only the executable device-driver information to the answer file
/V:*n*	Specifies the verbose level used to report status during the deployment process. Valid values for *n* are: **0** - quiet mode **3** - default **5** - maximum information

Return codes

The unattend.exe utility returns the following values to indicate status:

Value	Description
0	Success
1	Syntax error
2	Program error
3	Destination is read-only
4	No device-driver information files found

Examples

The following examples illustrate unattend.exe utility usage.

Example	Description
`unattend /U:c:\unattend.txt /D:c:\w2\$oem$\$1\drv /H:c:\hwdetect.ini /I:C:\i386`	Adds plug-and-play and executable device drivers from c:\w2\$oem$\$1\drv and the text mode device drivers from c:\w2\$oem$\textmode to the answer file, and deletes device drivers not specific to the target server
`unattend /U:C:\unattend.txt /D:c:\w2\$oem$\textmode /H:c:\hwdetect.ini /I:c:\i386 /T`	Adds only the text mode device drivers from c:\w2\oem\textmode directory to the answer file

UNATTEND.INI

The `unattend.ini` file contains all required parameters for the `unattend.exe` utility in a single file. Parameters specified on the command line will override settings in this file.

The `unattend.ini` file contains two sections, called [Unattend] and [GUIRunOnce]. The [Unattend] section contains variables that you can set instead of providing command-line parameters. The [GUIRunOnce] section enables you to specify a set of commands to run on the target server after the operating system is installed. These commands can run before the executable device drivers are installed, or after they are completed.

None of the commands can cause the server to restart (reboot).

Run commands that require user interaction after the executable device drivers are installed.

The `unattend.ini` file can contain the following valid variables:

Variable name	Description
[Unattend] section	
Drivers Path	Specifies a fully qualified path to the device-drivers directory in the target server. If `$oem$` is not in the path, you must use the /T parameter to process text mode device drivers.
Executable	Causes the `unattend.exe` utility to add only the executable device-driver information to the answer file. Valid values are `True` or `False`.
HWDetectIni	Specifies a fully qualified path and file name for the `hwdetect.ini` file that was created by the `hwdetect.exe` utility
I386 Path	Specifies a fully qualified path to the i386 directory in the target server.
PnP	Causes the `unattend.exe` utility to add only the plug-and-play device-driver information to the answer file. Valid values are `True` or `False`.
System Drive	Specifies the drive letter on the target server to which the operating system is being installed
Textmode	Causes the `unattend.exe` utility to add only the text mode device-driver information to the answer file. Valid values are `True` or `False`.
UnattendTxt	Specifies a fully qualified path and file name for the answer file
Verbose Level	Specifies the verbose level used to report status during the deployment process. Valid values are: **0** - quiet mode **3** - default **5** - maximum information

Variable name	Description
[GUIRunOnce] section	
*name*_Command	Specifies the *name* of the command to run
*name*_Supported_Systems	Specifies the servers on which to run the command, *name*. This value can be `All`, `None`, or a comma-delimited list of server machine types or platform IDs. You cannot use both *name*_Supported_Systems and *name*_Unsupported_Systems in the same `unattend.ini` file.
*name*_Unsupported_Systems	Specifies the servers on which not to run the command, *name*. This value must be a comma-delimited list of server machine types or platform IDs. You cannot use both *name*_Supported_Systems and *name*_Unsupported_Systems in the same `unattend.ini` file.
After Drivers	Specifies a comma-delimited list of commands to run after the executable device drivers are installed. Each command must have a *name*_Command variable and either a *name*_Supported_Systems or *name*_Unsupported_Systems variable defined.
Before Drivers	Specifies a comma-delimited list of commands to run before the executable device drivers are installed. Each command must have a *name*_Command variable and either a *name*_Supported_Systems or *name*_Unsupported_Systems variable defined.

The following are examples of `unattend.ini` file contents:

Example	Description

`[Unattend]`
`UnattendTxt=c:\unattend.txt`
`Drivers Path=c:\w2\$oem$\$1\drv`
`HWDetectIni=c:\hwdetect.ini`
`I386 Path=c:\i386`
`System Drive=`
`Textmode=`
`PnP=`
`Executable=`
`Verbose Level=`
	Adds plug-and-play device drivers from `c:\w2\$oem$\$1\drv` and text mode device drivers from `c:\w2\$oem$\textmode` to the answer file and deletes device drivers not specific to the target server

`[Unattend]`
`UnattendTxt=c:\unattend.txt`
`Drivers Path=c:\w2\$oem$\textmode`
`HWDetectIni=c:\hwdetect.ini`
`I386 Path=c:\i386`
`System Drive=`
`Textmode=True`
`PnP=`
`Executable=`
`Verbose Level=`
	Adds only the text mode device drivers from `c:\w2\$oem$\textmode` directory to the answer file

Example	Description
[Unattend] UnattendTxt=c:\unattend.txt Drivers Path=c:\w2\$oem$\$1\drv HWDetectIni=c:\hwdetect.ini I386 Path=c:\i386 System Drive= Textmode= PnP= Executable= Verbose Level= [GUIRunOnce] Before Drivers=LaunchIt,MoveIt After Drivers=DeleteIt,FinishIt LaunchIt_Command="CMD.EXE /C c:\RunMe.exe" LaunchIt_Supported_Systems=All MoveIt_Command="CMD.EXE /C Move c:\WinInst\Readme.htm c:\" MoveIt_Supported_Systems=8676,8870 DeleteIt_Command="CMD.EXE /C RMDIR c:\WinInst /q" DeleteIt_Unsupported_Systems=8870 FinishIt_Command="CMD.EXE /C c:\ShowMsg.exe" FinishIt_Supported_Systems=All	Adds the plug-and-play device drivers and executable device drivers from c:\w2\$oem$\$1\drv and the text mode device drivers from c:\w2\$oem$\textmode to the answer file, deletes device drivers not specific to the target server, and runs some specific commands both before and after executable device drivers are installed on specific servers

VALRAID

VALRAID is a utility program that can be used to validate policy files against inventory files generated by the INVRAID utility.

VALRAID has two modes of operation:

- **Simulation mode** - simulates the effect a policy file would have on a controller.
- **Check mode** - determines whether the policy file matches the configuration represented in the inventory file.

Simulation mode

Used in simulation mode, VALRAID will simulate the effect that a policy file would have on a RAID configuration if it were applied using the pRAID uitlity. This capability can be used when creating pRAID policy files. The policy files can be tested without running pRAID on the target system.

Check mode

Used in check mode, VALRAID determines whether the policy file specified matches the RAID configuration represented in the inventory file. This capability can be used in OS deployment scripts to skip the RAID configuration step if the controller is already configured with the required RAID configuration and thus avoiding an extra reboot before installing the OS. VALRAID will set the return code = 20 to indicate that the policy file does not match the configuration represented by the inventory file.

Usage

The two modes of operation share most parameters, but the syntax is mode-specific.

The simulation mode syntax is:

```
valraid /ini:input_inventory_file /inp:input_policy_file
/outi:output_inventory_file /outp:output_policy_file
/raid:/inifiles
```

The check mode syntax is:

```
valraid /c /ini:input_inventory_file /inp:input_policy_file /raid:/inifiles
```

Table 16. VALRAID parameters

Parameter	Description	Example
/ini:*input_inventory_file* Specifies the input inventory file.	Specifies the input inventory file. Generate the inventory file by running INVRAID against a target system.	valraid /ini:myfile.inv /inp:policy.ini /outi:newfile.inv /outp:newpolicy.ini /raid:/inifiles
/inp:*input_policy_file* Specifies the input policy file.	Specifies the input policy file.	valraid /ini:myfile.inv /inp:policy.ini /outi:newfile.inv /outp:newpolicy.ini /raid:/inifiles
/outi:*output_inventory_file* Specifies the filename of the output inventory file.	Specifies the filename for the output inventory file. This is an inventory file representing the RAID configuration that would result from using the PRAID utility to apply *input_policy_file* to the system described in *input_inventory_file*. This option is valid only for simulation mode.	valraid /ini:myfile.inv /inp:policy.ini /outi:newfile.inv /outp:newpolicy.ini /raid:/inifiles
/outp:*output_policy_file* Specifies the filename for the output policy file.	Specifies the filename for the output policy file. This file can be applied to a target system using the pRAID utility. This option is valid only for simulation mode.	valraid /ini:myfile.inv /inp:policy.ini /outi:newfile.inv /outp:newpolicy.ini /raid:/inifiles
/raid:*inifiles* Specifies the path to the RAID configuration ini files.	Specifies the directory that contains the RAID ini files. The default is /opt/ibm/sgtk/sgdeploy/sgtklinux/.data/valraid	valraid /ini:myfile.inv /inp:policy.ini /outi:newfile.inv /outp:newpolicy.ini /raid:/inifiles
/c Specifies check mode.	Specifies check mode. Check mode compares the configuration from *input_inventory_file* to the configuration represented in *input_policy_file*. The default is simulation mode.	valraid /c /ini:myfile.inv /inp:policy.ini /raid:inifiles

Return codes

VALRAID uses the following return codes:

- 0 – Success
- 1 – Error parsing input policy file
- 2 – Error parsing input inventory file
- 3 – Controller is not supported
- 4 – Raid level is not supported
- 5 – Stripesize is not supported
- 6 – Number of arrays not supported
- 7 – Number of drives in array not supported
- 8 – Number of logical volumes in array is not supported
- 9 – Not enough drives to create hotspare
- 10 – Not enough drives of the same size
- 11 – Error opening input policy file
- 12 – Error opening input inventory file
- 13 – Error opening output inventory file
- 14 – Error writing to output inventory file
- 15 – Error opening output policy file
- 16 – Error writing output policy file
- 17 – Partial drive sizing not supported
- 18 – Command line syntax error
- 19 – No policy match
- 20 – Controller not configured, does not match policy file

Appendix C. Incorporating the Scripting Toolkit with your existing process

To incorporate Scripting Toolkit procedures into an existing deployment process, use the HWDETECT.EXE utility to determine if the combined process is being executed on Scripting Toolkit supported hardware. You can then add appropriate branches in the batch files to use the existing process or the Scripting Toolkit process.

For example, you might use the /s option of HWDETECT.EXE to determine if the current system is an IBM eServer, xSeries, or BladeCenter server:

```
hwdetect.exe /s
if errorlevel 1 goto NONIBM
if errorlevel 0 goto IBM

:NONIBM
rem Perform non-IBM equipment specific processing here.

:IBM
rem Perform IBM eServer or xSeries equipment specific processing here.
```

Appendix D. Hints and tips

This section contains information on known problems and limitations, best practices, and hints and tips for using the Toolkit.

Using UXSPI to download updates

This section describes how to acquire firmware and driver updates for your IBM Servers using the graphical user interface (GUI) of the UpdateXpress System Pack Installer (UXSPI). For more information about using UXSPI, refer to the IBM UpdateXpress System Pack Installer User's Guide (ibm_utl_uxspi_x.xx_anyos_noarch), located in the `sgdeploy\SGTKWinPE\docs\uxspi\` folder in the location where you installed the IBM ServerGuide Scripting Toolkit Windows Edition.

To acquire driver and firmware updates for your system using UXSPI in the GUI mode, follow the instructions below:

1. Go to the location where you installed the ServerGuide Scripting Toolkit Windows Edition and navigate to the `sgdeploy\updates\uxsp` directory.
2. Start UXSPI by double-clicking the ibm_utl_uxspi_x.xx_winsrvr_32-64.exe executable. The main UXSPI window opens.
3. Click **Next** to proceed to the Select Command window.
4. Select **Create a repository of updates** and click **Next**.
5. Follow the on-screen instructions in the wizard to download the latest UXSPs or firmware updates.

Installing an operating system on a multi-adapter system

This section describes the special considerations for installing an operating system on a multi-adapter system.

When you perform a Windows installation the ServerGuide Scripting Toolkit, Windows edition attempts to install Windows to the first disk on the system presented by the diskpart command line utility. Due to limitations of the system, when multiple storage adapters are present on the system, the first disk presented by the diskpart utility is not always the first disk on the system.

You can control what disk the operating system is installed to using the TK_Partition_DiskNum variable in the Partitioning section of the SGTKWinPE.ini settings file. To determine the value for the disk you want to install to, complete RAID configuration and then run the utility GetDiskData.cmd to determine the available disks.

When a system contains a Fibre HBA that has been configured with a logical drive mapped to Logical Unit Number (LUN) 0, Toolkit uses this drive as the boot device. Therefore, Toolkit will fail if a different drive is selected for Windows installation. If you want to install to a different drive, disable the BIOS for the HBA before beginning your deployment.

Adding additional software components for installation post first Autologin

This section describes how to add additional software components for installation after the first Autologin.

By adding the software applications to the Scripting Toolkit and editing the custom post-OS installation script, you can add software components to your server as part of the post-OS installation stage. Follow these steps to add components:

1. Add the software application files to `sgdeploy\SGTKWinPE\Scripts\Custom`. All of the files in this directory will be copied to the installation image.
2. Add the commands to invoke the software application files to `sgdeploy\SGTKWinPE\Scripts\Custom\CustomPostInstall.cmd`.

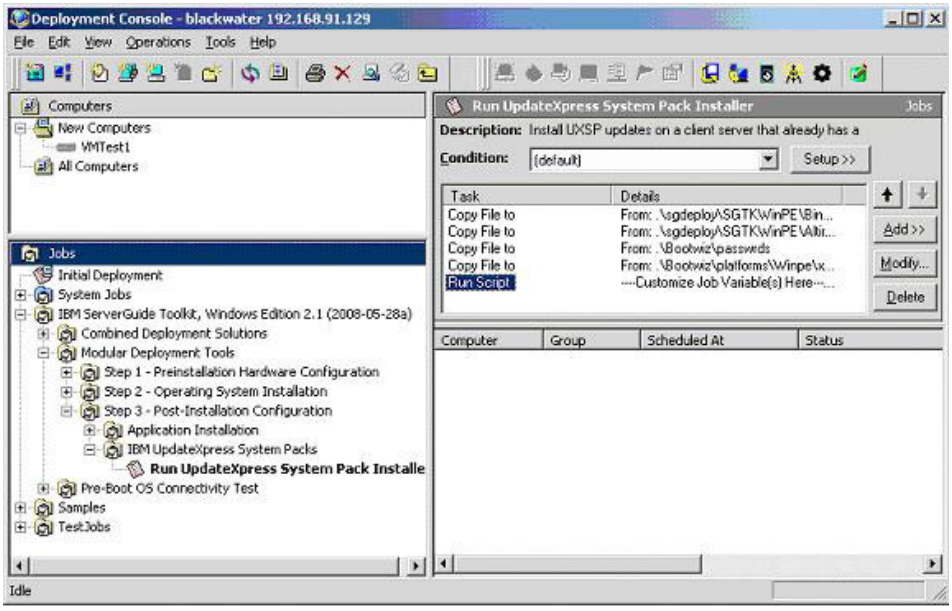
Note: This procedure is not suitable for use with interactive applications that require additional user input. Attempting to install applications that require user interaction can cause the post-OS installation stage to hang while waiting for user input.

Changing the security context of an Altiris task

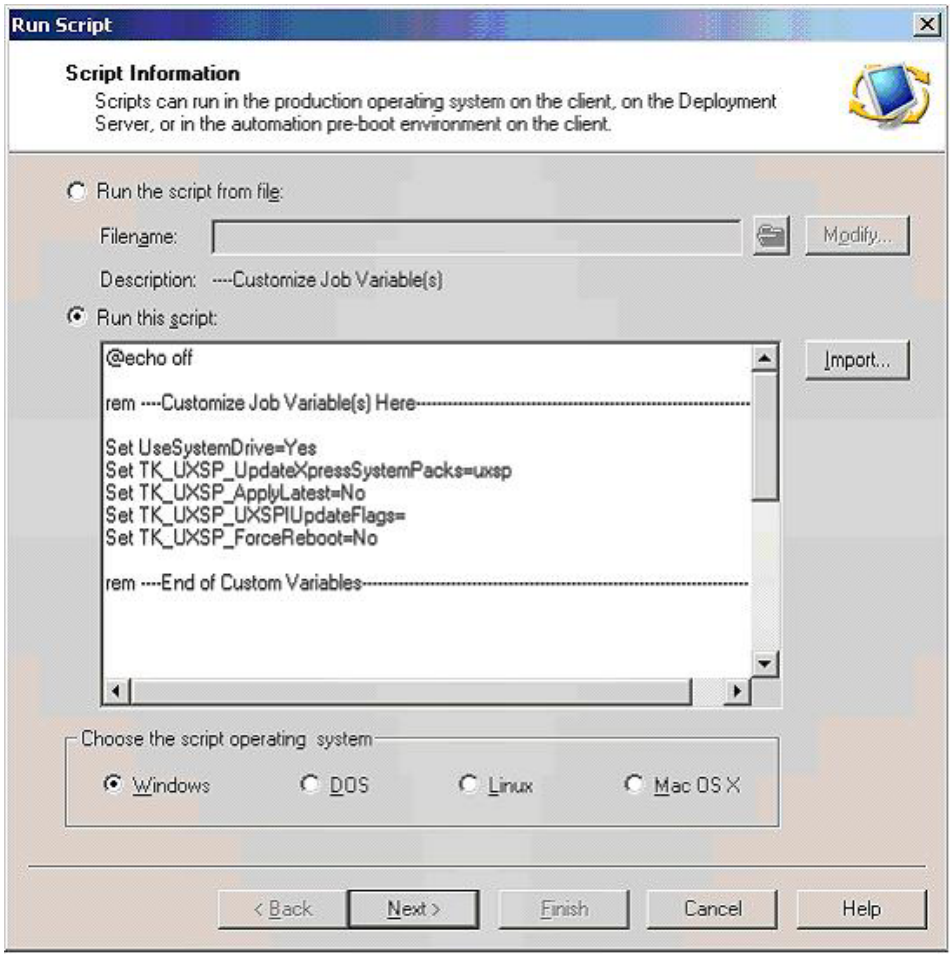
The IBM Director Agent and UpdateXpress Systemp Pack Post Installation jobs need Administrator privileges to run properly. This section describes the process for changing the security context of these jobs.

Follow these steps to change the security context of an Altiris job:

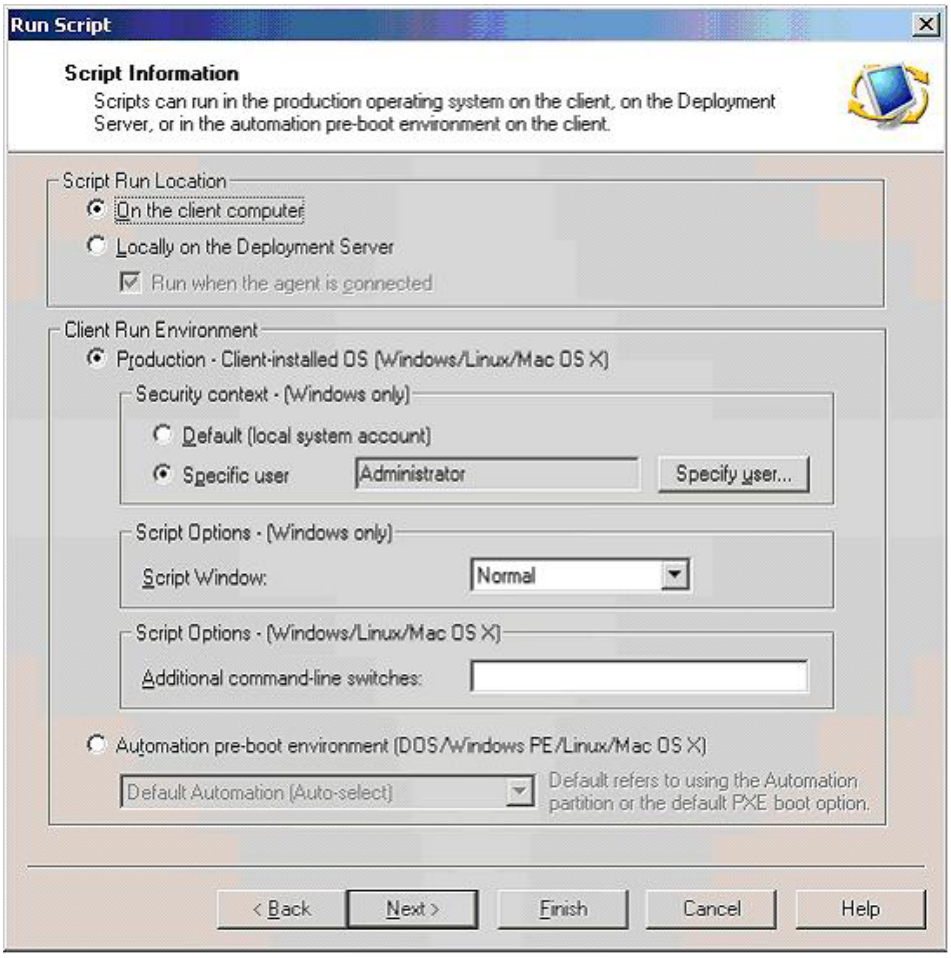
1. Start the Altiris Deployment Solution console, select and expand the **IBM ServerGuide Toolkit, Windows Edition** job folder.
2. Under **Modular Deployment Tools**, expand **Post-Installation Configuration**.
3. Under **Post-Installation Configuration**, expand the appropriate subfolder and select the job you want to modify. This example uses the **Run UpdateXpress System Pack Installer** job.
4. Click the **Run Script** to select it, as shown here:



5. With the **Run Script** task highlighted, click **Modify** to open the Run Script wizard as shown here:



6. Click **Next**.
7. Under Client Run Environment, select **Specify user** and enter Administrator, as shown here:



8. Click **Specify user** to open the Set user name and password dialog.
9. In the dialog, enter the Administrator password for the system, then click **OK**.
10. Click **Next** to proceed to the end of the Run Script wizard, then click **Finish** to exit the wizard.

Known problems and limitations

This section provides information and alternative solutions for known problems and limitations of the Toolkit.

The Toolkit will not work on a network share drive or any drive that is not formatted using NTFS

The Toolkit uses the Microsoft Windows Imagex.exe utility during the generation of the Windows PE ISO. The Imagex.exe utility will fail during the creation of the deployment scenario if the Toolkit is installed to a mapped network drive or a USB device if it is formatted with anything other than the NTFS file system.

Unknown HID SYS device in Windows Device Manager

When an IBM Remote Supervisor Adapter-II is installed in a server, an unknown HID SYS device may appear in the Windows Device Manager list if the adapter OS Type is set Linux. You can remove the unknown HID SYS device by completing the following steps after the OS installation:

1. Boot the machine and press **F1** to enter System Setup.
2. Navigate to **Advanced Setup**.
3. Navigate to **RSA II Setting**
4. Set the OS Type of the Remote Supervisor Adapter-II to **Other**.
5. Save and exit from System Setup.
6. Reboot the machine and start Windows.
7. Navigate to Device Manager.
8. Right-Click the unknown HID SYS device in the Windows Device Manager list.
9. Select **Update Driver** from the menu.
10. Select **No, not at this time** to connect to Windows Update.
11. Click **Next**.
12. Click **Next**.
13. Click **Finish**.

Altiris Agent Fails To Connect To Altiris Server

When you perform a Windows installation to an IBM blade within an BladeCenter S chassis, the installed Altiris Client Agent might not be able to connect to the Altiris Deployment Solution server after the deployment.

To resolve this issue, apply the following changes to the registry on the target system within the HKEY_LOCAL_MACHINE\Software\Altiris\Client Service folder:

Key	Type	Value
LastMCastDiscoveredTcpAddr	REG_SZ	*ServerIP*
LastMCastDiscoveredTcpPort	REG_SZ	402

Where *ServerIP* is the IP address of the Altiris Deployment Solution server.

After you have made the changes you must restart the Altiris Client Service for them to take effect.

Installing an operating system to a system with multiple storage adapters from Altiris

When you perform a Windows installation, the ServerGuide Scripting Toolkit, Windows edition attempts to install Windows to the first disk on the system presented by the diskpart command line utility. When multiple storage adapters are present on the system, the first disk presented by the diskpart utility is not always the first disk on the system.

To perform the installation to the first disk on a system with multiple storage adapters, you might need to modify the TK_Partition_DiskNum variable in the Altiris job. To locate the first disk, run the Capture Disk Data job by following these steps:

Note: Run the Capture Disk Data job after you have configured RAID.

1. Start the Altiris Deployment Solution console.

2. Select **IBM Scripting Toolkit, Windows Edition 9.51** > **Modular Deployment Tools** > **Step 1 - Preinstallation Hardware Configuration** > **Server Disk Analysis** .
3. Run the Capture Disk Data job against the system whose disk information you would like to retrieve.

This job will create a disk information file and store it on the source server. By default, it creates the file: `C:\Program Files\Altiris\eXpress\Deployment Server\temp\ID\disk_details.txt`.

From the output file, locate the disk number where you would like to install Windows. Then, in the Windows installation job in Altiris, modify the TK_Partition_DiskNum variable with the correct disk number and perform the Windows Installation.

Partitioning disks appears to hang at 0 percent complete

When using Scripting Toolkit to partition disks of 300 GB or larger, the process might appear to hang at 0 percent complete. The process is not hung, but it might take 15-20 minutes for the process to complete.

Windows ComputerName must be alphanumeric

The *ComputerName* variable used for Windows installations must be alphanumeric, and must contain at least one letter. Valid values of *ComputerName* must be 15 characters or less.

If *ComputerName* does not meet these criteria, you will receive an error during unattended Windows 2008 installations saying:

```
Windows could not parse or process the unattended answer file for pass
(specialize).
The settings specified in the answer file cannot be applied.
The error was detected while processing settings for component
[Microsoft-Windows-Shell-Setup].
```

The compname used by Altiris is used as the default value of *ComputerName*. To avoid this error, ensure that the compname in the Altiris console for the node meets the criteria for the Windows *ComputerName* variable. You can edit the answer file to assign a name to the computer, or you can use * to assign a random name to the computer. The following example shows how to edit the `windows2008.xml` file to change the computername.

```
<ComputerName>IBM1</ComputerName>
```

or

```
<ComputerName>*</ComputerName>
```

Limitations for RSA-II installations

You might encounter errors when using RSA-II to install Windows Server 2008 using an ISO image. These errors vary depending on the type of system to which you are installing. In order to avoid these errors, when using RSA-II to install Windows Server 2008, use a mounted physical CD or DVD instead of an ISO image.

Slow network installations on System x3850 and x3950

Network installations using onboard Ethernet communications on the System x3850 and x3950 are very slow. You can avoid this problem by using an external Ethernet adapter.

Error for non existent PS/2 mouse in Device Manager

Installations of Windows Server 2008 can result in an error displayed in Device Manager for a non existent PS/2 mouse.

Because the error refers to a device that is not present, it can be safely ignored.

To fix this error on a System x 3250 server update the BIOS to level 1.42a or higher.

Cannot deploy Windows Server 2012 on systems with active software RAID controller

You cannot use the Toolkit for Altiris to deploy Windows Server 2012 on systems with an active software RAID controller. As an alternative, you can use ServerGuide to deploy Windows Server 2012 on these systems.

Deploy System Settings Altiris job fails on System x3250 M2

When using Deploy System Settings to deploy CMOS settings to a System x3250 M2, the job fails with return code 5 and the following command line message:

```
Passwords cannot be set in "replicate" mode.  They can only be set in
"batch" or "set" mode.
```

This error occurs because the System x3250 M2 sets the CMOS values cmosSuperPassword and cmosUserPassword to "hidden", causing the error when the settings are deployed.

To avoid this problem, edit the settings INI file created by the Capture Settings job to remove the following settings:

```
cmosSuperPassword=<hidden>
cmosUserPassword=<hidden>
```

and then deploy the settings normally.

Savestat.cmd will not save to location 9 on xSeries 226 with BIOS PME170CUS

On the xSeries 226 with BIOS Level PME170CUS, **savestat.cmd** cannot save a value to byte nine in persistent storage.

BladeCenter HS22 unable to access bootable deployment media

When performing deployments to a BladeCenter HS22, type 1936 or 7870, the system might be unable to access the bootable media being used for the deployment. This error is caused by the system's inability to find the mounted media tray.

When this problem occurs, it is possible for all blades in the chassis to lose access to the media tray. To correct the problem, restart the BladeCenter Advanced Management Module (AMM) to restore access to the media tray.

To correct the problem, update the firmware for the system and the BladeCenter chassis to the latest level available.

Yellow exclamation point for Microsoft ISATAP adapter

A yellow exclamation point icon might be displayed by Device Manager for the Microsoft ISATAP adapter after installing Windows Server 2008. In most cases, the adapter is functioning properly and you can continue to use the device normally.

For more information on this situation, see http://support.microsoft.com/kb/932520.

Booting from SAN using Brocade Fibre Channel Adapters not supported

Booting from SAN using the Brocade Fibre Channel Adapters listed in this document is not supported.

Defualt Fibre configuration not supported on Emulex HBAs

The Target WWNN, Target WWPN and LUN number on the Fibre HBA Toolkit variables need to be set to configure the Primary, Alternate 1, Alternate 2 and Alternate 3 boot device settings. The default settings will NOT work on Emulex Fibre HBA adapters.

No x64 support for SCSI RAID controller configuration

The IBM Scripting Toolkit support only 32-bit RAID configuration of ServeRAID SCSI and LSI SCSI RAID controllers. If 64-bit RAID configuration is attempted, you might receive a "No controller found" message. To avoid this issue, use the x86 RAID configuration for the SCSI controllers.

The following controllers are affected by this issue:
- ServeRAID 4H
- ServeRAID 4Lx
- ServeRAID 4Mx
- ServeRAID 5i
- ServeRAID 6i/6i+
- ServeRAID 6M
- ServeRAID 7k
- LSI SCSI (1020/1030)

ServeRAID BR10i adapter not supported on iDataPlex® dx360 M2 with 12 Bay Storage Chassis (Machine type 7321)

IBM Systems Director 6.11 Platform Agent installation fails on Windows 2008

When installing the platform agent on Windows Server 2008, it might fail with return code 400. This indicates that the target system does not include Windows Installer 4.5.

To resolve this issue, follow the instructions in Microsoft Knowledge Base article 942288: http://support.microsoft.com/kb/942288.

Booting from SAN is not supported for the QLogic 10Gb Dual Port CNA for IBM System x (42C1800)\

When installing Windows using ServerGuide, disks show special ownership

When you install Windows 2008 using ServerGuide, ownership of the disk is assigned to **TrustedInstaller**. This is different from the default assignment of **Administrators** in a native Windows installation. To use the default assignment, edit the security property of the installation file to assign ownership to **Administrators**.

Failed to run IBM Director Agent For Window Install task in Altiris Console

When performing IBM Director Agent For Window install, you might encounter an error message "The system cannot find the drive specified", which might be caused by incorrect drives inventory setting.

To avoid this problem, change the drivers setting by following these steps:

1. In the Computers panel, right-click and select properties for your system.
2. In the left navigation panel, select **Drives** in left navigation.
3. If the drives are set as Local Disk(/), right-click and select **Advanced->Get Inventory**.
4. Specify the drives to Local Disk(C).

Appendix E. Getting help and technical assistance

If you need help, service, or technical assistance or just want more information about IBM products, you will find a wide variety of sources available from IBM to assist you. This appendix contains information about where to go for additional information about IBM and IBM products, what to do if you experience a problem with your xSeries or IntelliStation® system, and whom to call for service, if it is necessary.

Before you call

Before you call, make sure that you have taken these steps to try to solve the problem yourself:

- Check all cables to make sure that they are connected.
- Check the power switches to make sure that the system is turned on.
- Use the troubleshooting information in your system documentation, and use the diagnostic tools that come with your system. Information about diagnostic tools is in the *Hardware Maintenance Manual and Troubleshooting Guide* on the IBM *xSeries Documentation* CD or in the IntelliStation *Hardware Maintenance Manual* at the IBM Support Web site.
- Go to the IBM Support Web site at http://www.ibm.com/pc/support/ to check for technical information, hints, tips, and new device drivers or to submit a request for information.

You can solve many problems without outside assistance by following the troubleshooting procedures that IBM provides in the online help or in the publications that are provided with your system and software. The information that comes with your system also describes the diagnostic tests that you can perform. Most xSeries and IntelliStation systems, operating systems, and programs come with information that contains troubleshooting procedures and explanations of error messages and error codes. If you suspect a software problem, see the information for the operating system or program.

Using the documentation

Information about your IBM xSeries or IntelliStation system and preinstalled software, if any, is available in the documentation that comes with your systemin a variety of formats: books, online books, readme files, and help files.

See the troubleshooting information in your system documentation for instructions for using the diagnostic programs. The troubleshooting information or the diagnostic programs might tell you that you need additional or updated device drivers or other software. IBM maintains pages on the Web where you can get the latest technical information and download device drivers and updates. To access these pages, go to http://www.ibm.com/pc/support/ and follow the instructions. Also, you can order publications through the IBM Publications Ordering System at http://www.elink.ibmlink.ibm.com/public/applications/publications/cgibin/pbi.cgi.

Getting help and information on the Web

The IBM website has up-to-date information about IBM xSeries and IntelliStation products, services, and support. The address for IBM xSeries information is http://www.ibm.com/eserver/xseries/. The address for IBM IntelliStation information is http://www.ibm.com/pc/intellistation/.

You can find service information for your IBM products, including supported options, at http://www.ibm.com/pc/support/.

Software service and support

Through IBM Support Line, you can get telephone assistance, for a fee, with usage, configuration, and software problems with xSeries servers, IntelliStation workstations, and appliances. For information about products that are supported by Support Line in your country or region, go to http://www.ibm.com/services/sl/products/.

For more information about Support Line and other IBM services, go to http://www.ibm.com/services/, or go to http://www.ibm.com/planetwide/ for support telephone numbers. In the U.S. and Canada, call 1-800-IBM-SERV (1-800-426-7378).

Hardware service and support

You can receive hardware service through IBM Services or through your IBM reseller, if your reseller is authorized by IBM to provide warranty service. Go to http://www.ibm.com/planetwide/ for support telephone numbers, or in the U.S. and Canada, call 1-800-IBM-SERV (1-800-426-7378).

In the U.S. and Canada, hardware service and support is available 24 hours a day, 7 days a week. In the U.K., these services are available Monday through Friday, from 9 a.m. to 6 p.m.

Appendix F. Notices

This book contains the following notices designed to highlight key information:

- **Note:** These notices provide important tips, guidance, or advice.
- **Important:** These notices provide information or advice that might help you avoid inconvenient or difficult situations.
- **Attention:** These notices indicate possible damage to programs, devices, or data. An attention notice is placed just before the instruction or situation in which damage could occur.

Edition notice

Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both:

BladeCenter
e-business logo
eServer
IBM
IBM (logo)
TotalStorage
IntelliStation
ServeRAID
ServerGuide
ServerProven
xSeries

Intel and Pentium are trademarks of Intel Corporation in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a trademark of The Open Group in the United States, other countries, or both.

Other company, product, or service names might be trademarks or service marks of others.

Important notes

When referring to processor storage, real and virtual storage, or channel volume, KB stands for approximately 1000 bytes, MB stands for approximately 1,000,000 bytes, and GB stands for approximately 1,000,000,000 bytes.

When referring to hard disk drive capacity or communications volume, MB stands for 1,000,000 bytes, and GB stands for 1,000,000,000 bytes. Total user-accessible capacity might vary depending on operating environments.

IBM makes no representation or warranties regarding non-IBM products and services that are ServerProven, including but not limited to the implied warranties of merchantability and fitness for a particular purpose. These products are offered and warranted solely by third parties.

IBM makes no representations or warranties with respect to non-IBM products. Support (if any) for the non-IBM products is provided by the third party, not IBM.

Some software can differ from its retail version (if available), and may not include user manuals or all program functionality.

Index

Printed in USA