

IBM ServerGuide Scripting Toolkit

Scripting Toolkit PRAID utility support for Windows Preinstallation Environment x86

A White Paper

April 5, 2007

Notes:

Visit www.ibm.com/pc/safecomputing periodically for the latest information on safe and effective computing.

Warranty Information: For a copy of applicable product warranties, write to: Warranty Information, P.O. Box 12195, RTP, NC 27709, Attn: Dept. JDJA/B203. IBM makes no representation or warranty regarding third-party products or services.

Before using this information and the product it supports, read the general information in "Notices," on page 63.

© Copyright International Business Machines Corporation 2006. All rights reserved.

U.S. Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

1	Preface.....	4
1.1	Intended Audience	4
1.2	Supported Operating Systems	4
1.3	Supported Software	4
1.4	Supported RAID Controllers.....	4
2	Scripting Toolkit PRAID utility support for Windows Preinstallation Environment x86	5
2.1	The WinPE_RAID_Configuration companion zip file	5
2.2	Unpacking the WinPE_RAID_Configuration companion zip file	5
2.3	Prerequisites	6
2.4	Creating the Windows PE bootable CDROM.....	7
2.4.1	Customizing the BuildWinPE Process	7
2.4.2	Building the Windows PE Image.....	10
2.4.3	Creating the ISO file.....	11
2.4.4	Creating the CDROM.....	11
2.4.5	Using the Windows PE bootable CDROM.....	11
2.5	Altiris Windows PE Support	12
2.5.1	Supported Windows PE sources for use with Altiris.....	12
2.5.2	Building the IBM Altiris Windows PE Image and Jobs.....	12
2.5.3	Using the new jobs within Altiris	13
2.6	Scripts and Utilities.....	15
2.6.1	Scripts	15
2.6.2	Toolkit Utilities.....	16
2.6.3	Third Party Utilities.....	56
3	Known Limitations & Bugs	61
4	Notices	63
4.1	Edition Notice	63
4.1.1	Trademarks	64

1 Preface

This white paper and companion zip file outlines performing the following tasks:

- Creating a bootable Windows PE x86 CDROM that supports RAID configuration and also network and mass storage devices for IBM System X and BladeCenter servers.
- Adding PRAID policy files to the Windows PE CDROM for custom RAID configurations.
- Adding the IBM Windows PE x86 driver set to Altiris Deployment Solution.
- Creating Windows PE based RAID configuration jobs within Altiris Deployment Solution.
- Creating Windows PE based imaging jobs within Altiris Deployment Solution.

1.1 Intended Audience

This document is intended for Systems Administrators who want to use a Windows Preinstallation Environment to configure RAID devices on IBM x86 architecture based servers. This guide covers intermediate and advanced systems administration tasks.

1.2 Supported Operating Systems

The white paper and companion zip file support the following Microsoft operating systems...

- Microsoft Windows XP
- Microsoft Windows Server 2003

Note: Microsoft Windows 2000 Professional and Microsoft Windows 2000 Sever are not supported by the BuildWinPE process.

1.3 Supported Software

This white paper and companion zip file only support Windows Preinstallation Environment version 2005. The Windows PE based Altiris Deployment Solution jobs are only supported with Altiris Deployment Solution 6.8 SP1 using either the Altiris supplied Windows PE version 2005.2.3.4 or greater or Microsoft Windows PE version 2005.

1.4 Supported RAID Controllers

The following RAID controllers are supported:

- ServeRAID adapter (SCSI)
- ServeRAID adapter (SATA)
- ServeRAID (Adaptec HostRAID) – (SCSI)
- ServeRAID (Adaptec HostRAID 7e) - (SATA)
- ServeRAID (Adaptec HostRAID 8e) – (SATA)

- ServeRAID (Adaptec HostRAID) – (SAS)
- ServeRAID adapter 8i (SAS)
- ServeRAID adapter 8k (SAS)
- ServeRAID adapter 8k-l (SAS)
- LSI-1020 & LSI-1030 chipsets
- LSI IDEal RAID
- LSI 1064/1064E (SAS)
- LSI 1068 (SAS)
- LSI MegaRAID 8480 (SAS)

2 Scripting Toolkit PRAID utility support for Windows Preinstallation Environment x86

The following section describes how to create a Windows Preinstallation Environment bootable CDROM using the BuildWinPE process provided in the WinPE_RAID_Configuration companion zip file.

2.1 The WinPE_RAID_Configuration companion zip file

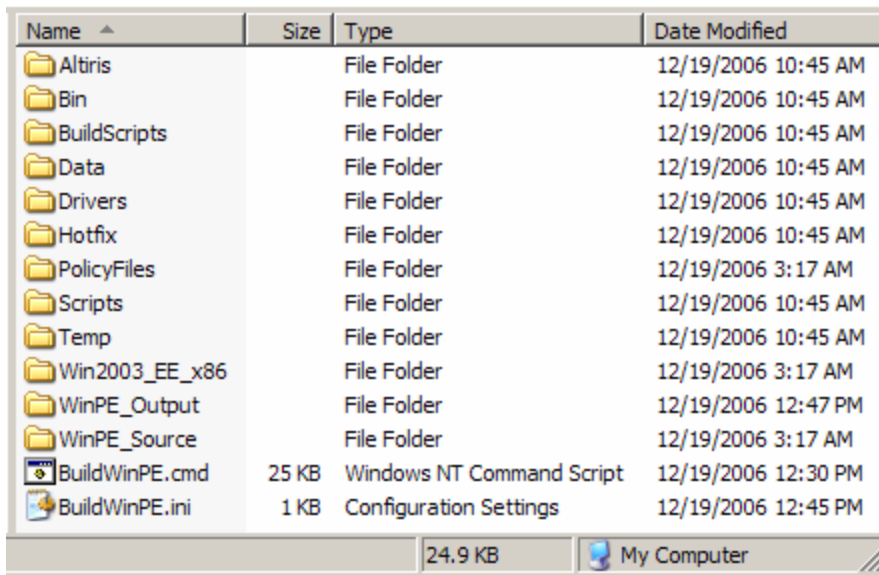
The WinPE_RAID_Configuration companion zip file contains all of the scripts, drivers, and utilities necessary to create a bootable Windows Preinstallation Environment CDROM that supports the IBM System X and BladeCenter servers and the PRAID utility. The companion zip file also contains scripts and jobs to create a bootable Windows Preinstallation Environment for Altiris 6.8 Service Pack 1 that supports both RAID configuration and imaging.

2.2 Unpacking the WinPE_RAID_Configuration companion zip file

The WinPE_RAID_Configuration zip file can be unpacked into any subdirectory on a system as long as no spaces are in the path. The Microsoft Windows PE mkimg.cmd command will not work with spaces in the path. The directory that the files are unpacked to is called the *WinPE_Base* directory.

Example: C:\WinPE_Base

The companion zip file contains the following directories and files...



Name	Size	Type	Date Modified
Altiris		File Folder	12/19/2006 10:45 AM
Bin		File Folder	12/19/2006 10:45 AM
BuildScripts		File Folder	12/19/2006 10:45 AM
Data		File Folder	12/19/2006 10:45 AM
Drivers		File Folder	12/19/2006 10:45 AM
Hotfix		File Folder	12/19/2006 10:45 AM
PolicyFiles		File Folder	12/19/2006 3:17 AM
Scripts		File Folder	12/19/2006 10:45 AM
Temp		File Folder	12/19/2006 10:45 AM
Win2003_EE_x86		File Folder	12/19/2006 3:17 AM
WinPE_Output		File Folder	12/19/2006 12:47 PM
WinPE_Source		File Folder	12/19/2006 3:17 AM
BuildWinPE.cmd	25 KB	Windows NT Command Script	12/19/2006 12:30 PM
BuildWinPE.ini	1 KB	Configuration Settings	12/19/2006 12:45 PM

Fig 1. Contents of the WinPE_RAID_Configuration companion zip file

The directories and files are defined as shown:

Directory	Description
Altiris	Scripts specific to integrating the Windows PE Image into Altiris
Bin	Toolkit utilities
BuildScripts	Scripts specific to building the Windows PE Image
Data	Data files used when building the Windows PE Image
Drivers	Driver files used when building the Windows PE Image
Hotfix	Hotfix files used when building the Windows PE Image
PolicyFiles	Policy files that are copied directly to the Windows PE Image
Scripts	Script files that are copied directly to the Windows PE Image
Temp	Temp directory used by the scripts
Win2003_EE_x86	Windows Server 2003 Enterprise Edition x86 w/ Service Pack 1 files
WinPE_Output	BuildWinPE process output directory
WinPE_Source	Windows Preinstallation Environment 2005 files
BuildWinPE.cmd	BuildWinPE main script file
BuildWinPE.ini	BuildWinPE settings file

2.3 Prerequisites

The following user provided files are required for the BuildWinPE process ...

- Microsoft Windows PE 2005
- Microsoft Windows Server 2003 Enterprise Edition with Service Pack 1 (32-bit)
- Microsoft Hotfix KB903825

Copy each of the requirements to the following locations in the unpacked WinPE_RAID_Configuration companion zip file:

Required files	Default directory
Microsoft Windows PE 2005	<i>WinPE_Base\WinPE_Source</i>
Microsoft Windows Server 2003 Enterprise Edition with Service Pack 1 (32-bit)	<i>WinPE_Base\Win2003_EE_x86</i>
Microsoft Hotfix KB903085 (setupdd.sys)	<i>WinPE_Base\Hotfix\x86\kb903085</i>

If the required files are already present on the system in a different directory, then the settings for the paths in the BuildWinPE.ini file can be changed to the location of the required files. See the section on Customizing BuildWinPE.

The following is also required to create a bootable Windows PE CDROM...

- User provided software to create a CDROM from an ISO file.

Note: The CDROM creation software is not required for Altiris users.

2.4 Creating the Windows PE bootable CDROM

Creating the Windows PE bootable CDROM requires using the BuildWinPE build process scripts provided in the WinPE_RAID_Configuration companion zip file. There are four steps to building the CDROM...

- Customize the BuildWinPE process.
- Build the Windows PE Image.
- Create the ISO file of the Windows PE Image.
- Create the CDROM from the ISO file.

2.4.1 Customizing the BuildWinPE Process

There are three areas that can be customized for the BuildWinPE process...

- BuildWinPE User Settings
- Adding PRAID policy files to the Windows PE Image
- Adding custom scripts to the BuildWinPE process

Note: The BuildWinPE User Settings will require customizing if the prerequisite files are not copied to the default locations.

2.4.1.1 BuildWinPE User Settings

The following user definable settings are available in the *WinPE_Base\BuildWinPE.ini* file...

Variable	Description
[BuildWinPE Settings]	Section
ReduceSize	Reduces the size of the Windows PE x86 image. Deletes files in the Windows PE x86 image specified by the <i>WinPE_Base\Data\x86\removelist.txt</i> file. Default: Yes

ForceWinPEBoot	Forces the Windows PE CDROM to always boot. Normally, if there is a file system present when Windows PE boots, it displays the message: "Press any key to boot from the CDROM..." and gives the user 8 seconds to press a key. The <i>ForceWinPEBoot</i> option disables this feature and forces the Windows PE CDROM to always boot. Default: Yes
ApplyKB903085	Applies Microsoft Hotfix KB903085. Default: Yes
CopyDrivers	Copies the IBM Windows PE Driver set to the Windows PE Image. Default: Yes
CopyToolkitFiles	Copies the following directories to the Windows PE Image... <i>WinPE_Base\Bin</i> <i>WinPE_Base\Scripts</i> <i>WinPE_Base\PolicyFiles</i> Default: Yes
EnablePRAIDConfiguration	Enable PRAID Configuration task on boot of the Windows PE CDROM. Default: Yes
CreateISO	Automatically create the ISO image. Default: No
CreateRamdiskISO	Automatically create the Ramdisk ISO image. Default: No
SuppressPrompts	Suppress interactive prompts during BuildWinPE script execution. Default: No
DisplayUserSettings	Display the user settings during BuildWinPE script execution. Default: No
[BuildWinPE Paths]	Section
WinPE_Source	Windows Preinstallation Environment 2005 source files. Default: %WinPE_Base%\WinPE_Source
Win2003_EE_x86_Source	Windows Server 2003 Enterprise Edition x86 SP1 (32-bit) files. Default: %WinPE_Base%\Win2003_EE_x86
KB903085	Microsoft Hotfix KB903085 files. Default: %WinPE_Base%\hotfix\x86\kb903085
WinPE_Output	Windows PE Build Output directory. Default: %WinPE_Base%\WinPE_Output
Custom_Script	Default custom script filename. Default: %WinPE_Base%\Custom\Custom.cmd
[BuildWinPE Filenames]	Section
WinPE_ISO_Filename	The Windows PE ISO filename

	Default: WinPE_x86.iso
WinPE_RAMDISK_ISO_Filename	The Windows PE Ramdisk ISO filename. Default: WinPE_x86_Ramdisk.iso
WinPE_Log_Filename	The Windows PE build process log filename. Default: WinPE_x86.log
[PRAID Settings]	Section
PolicyFile	Specifies the PRAID policy file to use for the configuration. This value should not contain any path information. The specified PRAID policy file must exist in the <i>WinPE_Base</i> \PolicyFiles directory. Default: <i>blank</i>
AutomaticallyRestart	Specifies to restart the system after the RAID configuration is complete. Default: No

Note: The %WinPE_Base% variable is automatically set by the BuildWinPE scripts to the directory where the BuildWinPE package was unzipped.

Example of the default BuildWinPE.ini file...

[BuildWinPE Settings]

ReduceSize=Yes
ForceWinPEBoot=Yes
ApplyKB903085=Yes
CopyDrivers=Yes
CopyToolkitFiles=Yes
EnablePRAIDConfiguration=Yes
CreateISO=No
CreateRamdiskISO=No
SuppressPrompts=No
DisplayUserSettings=No

[BuildWinPE Paths]

WinPE_Source=%WinPE_Base%\WinPE_Source
Win2003_EE_x86_Source=%WinPE_Base%\Win2003_EE_x86
KB903085=%WinPE_Base%\Hotfix\x86\kb903085
WinPE_Output=%WinPE_Base%\WinPE_Output
Custom_Script=%WinPE_Base%\Custom\Custom.cmd

[BuildWinPE Filenames]

WinPE_ISO_Filename=WinPE_x86.iso
WinPE_Ramdisk_ISO_Filename=WinPE_x86_Ramdisk.iso
WinPE_Log_Filename=WinPE_x86.log

[PRAID Settings]

PolicyFile=
AutomaticallyRestart=No

2.4.1.2 Adding PRAID policy files to the Windows PE Image

By default, RAID devices are configured with default settings. See the PRAID description in the Utilities section for more information. A custom policy file can be used instead by performing the following...

- Copy the custom PRAID policy file to the *WinPE_Base*\PolicyFiles directory.
- Specify the name of the custom policy file using the PolicyFile variable in the [PRAID] section of the BuildWinPE.ini file.

Note: Any policy files that exist in the *WinPE_Base*\PolicyFiles directory will be copied to the Windows PE Image, but only one can be activated at a time for automatic execution.

2.4.1.3 Adding custom scripts to the BuildWinPE process

Custom scripts can be added to the BuildWinPE process by creating a *WinPE_Base*\BuildScripts\Custom\Custom.cmd file. If the file exists, it is automatically executed by the BuildWinPE.cmd script file after the Windows PE Image is built (but prior to creating the optional ISO file). The custom script should set two environment variables upon completion...

- Set RC=*number*
- Set RMSG=*message*

The RC environment variable is the return code that can be set to inform the BuildWinPE.cmd script file that an error occurred during execution of the Custom.cmd script file. Zero indicates success and any other number indicates error. If it is set to a non-zero value, the RMSG environment variable should be set to an appropriate message that will be displayed to the console and written to the log file to indicate the error.

2.4.2 Building the Windows PE Image

The Windows PE Image is the collection of files and directories that will be placed on a CDROM. To create the Windows PE Image, run the BuildWinPE.cmd script file from the *WinPE_Base* directory at a command prompt.

Example:

```
C:\WinPE_Base>BuildWinPE.cmd
```

After this command completes, a Windows PE Image is created. The Windows PE Image can be optionally modified before the ISO file is generated for CDROM creation.

The default location for the generated *WinPE_Image* is *WinPE_Base*\WinPE_Output\WinPE_x86.

The layout of the Windows PE Image is shown below:

Directory	Description
<i>WinPE_Image\i386</i>	Windows PE files
<i>WinPE_Image\SGTK\Bin</i>	Toolkit Utilities (from <i>WinPE_Base\Bin\win32</i>)
<i>WinPE_Image\SGTK\PolicyFiles</i>	Policy files (from <i>WinPE_Base\PolicyFiles</i>)
<i>WinPE_Image\SGTK\Scripts</i>	Scripts (from <i>WinPE_Base\Scripts</i>)

2.4.3 Creating the ISO file

The ISO file of the Windows PE Image must be created before a CDROM can be created.

At a command prompt, run the BuildWinPE.cmd script with the /ISO switch from the *WinPE_Base* directory...

Example:

```
C:\WinPE_Base>BuildWinPE.cmd /ISO
```

This will create the *WinPE_Base\WinPE_Output\WinPE_x86.iso* file using the files located in the *WinPE_Base\WinPE_Output\WinPE_x86* directory.

Optional: To create the ISO file for a Windows PE bootable CDROM that can be removed after Windows PE is started, use the /RamdiskISO switch instead of /ISO.

Example:

```
C:\WinPE_Base>BuildWinPE.cmd /RamdiskISO
```

This will create the *WinPE_Base\WinPE_Output\WinPE_Ramdisk_x86.iso* file.

2.4.4 Creating the CDROM

Use the customer provided software to create the CDROM using the generated ISO file.

2.4.5 Using the Windows PE bootable CDROM

If the default User Settings are specified in the BuildWinPE.ini file, the PRAID configuration will run automatically when the Windows PE CDROM is booted on a system. After it completes, the user can interact with the environment via the command prompt window. To shutdown Windows PE, type Exit at the command prompt.

Other PRAID policy files created in the *WinPE_Base\PolicyFiles* directory (which are automatically copied to the *WinPE_Image*) can be ran manually by issuing the following command from the Windows PE command prompt window...

```
\SGTK\Scripts\ConfigureRAID.cmd \SGTK\PolicyFiles\nameofpolicyfile
```

Note: If the policy file parameter is not specified when running the ConfigureRAID.cmd script, then the default policy is in effect.

2.5 Altiris Windows PE Support

The WinPE_RAID_Configuration companion zip file contains the drivers, jobs, and utilities to support RAID configuration and Operating System imaging on IBM System X and BladeCenter servers in Altiris Deployment Solution 6.8 SP1.

The BuildWinPE script is used to add these components to the Altiris Deployment Solution server.

2.5.1 Supported Windows PE sources for use with Altiris

BuildWinPE requires one of the following versions of Windows PE for Altiris support:

- Altiris supplied Windows PE version 2005.2.3.4 or greater installed in the Altiris Deployment Solution server
- Microsoft supplied Windows PE 2005 source files

2.5.2 Building the IBM Altiris Windows PE Image and Jobs

At a command prompt, run the *WinPE_Base\BuildWinPE.cmd* script file with the */Altiris* switch to automatically build the Windows PE PXE image and create the default Windows PE jobs for RAID configuration and imaging.

Example:

```
C:\WinPE_Base>BuildWinPE.cmd /Altiris
```

In order for this command to run successfully the steps outlined in the Prerequisites section must have been performed. The Altiris Windows PE build process uses only the following settings from the BuildWinPE.ini file:

```
[BuildWinPE Settings]
ApplyKB903085=Yes
SuppressPrompts=No
```

```
[BuildWinPE Paths]
WinPE_Source = %WinPE_Base%\WinPE_Source
Win2003_EE_x86_Source = %WinPE_Base%\Win2003_EE_x86
KB903085 = %WinPE_Base%\hotfix\x86\kb903085
```

As long as the prerequisite files are copied to the default locations, then the BuildWinPE.ini file does not need to be changed to build an Altiris Windows PE image or add the jobs.

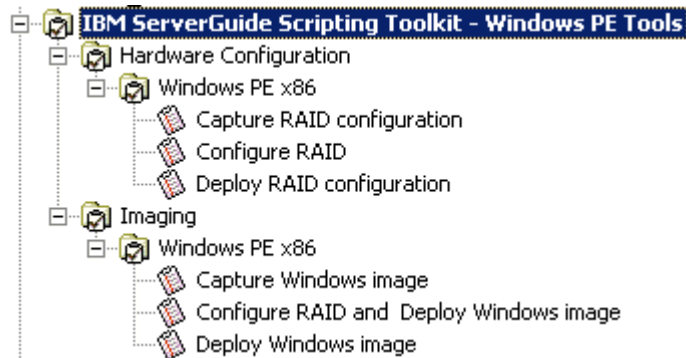
If Windows PE is already installed in the Altiris Deployment Solution server using the Altiris provided Windows PE installer, then the WinPE_Source variable is unused. The BuildWinPE.cmd script file will not attempt to rebuild the base Windows PE Image within Altiris Deployment Solution if it already exists.

Note: If the IBM_WinPE_x86 Altiris PXE image is already present, then it must be deleted prior to running the BuildWinPE script. The jobs associated with the PXE image must be removed from the Altiris console prior to deleting the PXE image. They can be optionally exported to a file to create a backup.

2.5.3 Using the new jobs within Altiris

The BuildWinPE.cmd script automatically adds six sample jobs to Altiris Deployment Solution.

The new jobs within Altiris are located under the following:



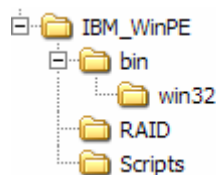
Each sample job executes using the Windows PE PXE image labeled IBM_WinPE_x86. This is a custom-built 32-bit Windows PE image that contains all the required drivers to support both networking and mass storage devices on IBM servers.

All associated scripts and policy files for the provided jobs are located in the IBM_WinPE directory within the "Deployment Server" directory of the Altiris Deployment Solution installation.

Example:

C:\Program Files\Altiris\Express\Deployment Solution\IBM_WinPE

This directory has the following structure:



The win32 directory contains all of the 32-bit utilities that are required for the provided jobs within Altiris.

The RAID directory is the location in which all RAID policy files are located. By default all RAID policy file captures are placed into an automatically generated directory within the RAID directory. The generated directory name is based on the machine type number.

The Scripts directory contains the scripts that are used by the sample jobs within Altiris.

The sample jobs can be modified from within the Altiris Deployment Solution console. Each sample job utilizes two user customizable variables named RD_PATH and RD_FILE. These variables can contain different values in each of the sample jobs.

Attention: If changes are made to an Altiris job or event, the changes are effective for all target servers that run that specific job or event. Do not make changes without creating a new job or

event, unless the desired behavior is to affect all target servers that are scheduled to run that job or event.

2.5.3.1 Configuring RAID

The RAID configuration within Windows PE is driven by the PRAID utility. In order to properly configure RAID, some variables within the sample jobs might require modification. The following variables can be modified within the job:

Variable Name	Description	Default
RD_PATH	The fully qualified path to the RAID Configuration file	%TK_PATH%\RAID
RD_FILE	The Name of the RAID configuration file.	blank

If the RD_FILE variable is left blank, then PRAID executes in default mode to configure the RAID in the target system. This configuration is based on the capabilities of the RAID controller(s) within the system and the number of available drives. Specifying a RAID policy file enables customization of the RAID configuration of the target system.

When configuring RAID on multiple systems or machine types, and each system has a unique policy file, you must do one of the following:

- Copy the existing job and modify the RD_FILE variable to point to the proper policy file in each job.
- Use the existing job and modify the RD_FILE variable prior to executing the job on each subsequent system.

If specified, each policy filename must be unique in the default RD_PATH directory.

2.5.3.2 Capturing and Deploying RAID configurations

The sample jobs for capturing RAID configurations to a policy file and deploying RAID configurations are customized for use with IBM System X and BladeCenter servers. The Capture Raid Configuration task determines how PRAID would configure the target RAID controller and creates a policy file to perform that configuration.

Once a RAID configuration has been captured to a policy file, it can then be deployed to a RAID controller or adapter.

The policy files must be located within the IBM_WinPE directory in order for the Windows PE Altiris tasks to utilize them. By default a RAID configuration is captured to the following directory:

IBM_WinPE\RAID*machtype*

Where *machtype* is the machine type of the captured system.

The target system must have the same number of hard disk drives attached to the same controller as the original system in order for the RAID deployment to be successful.

For capturing and deploying RAID configurations, the following variables can be modified within the provided sample jobs.

Variable Name	Description	Default
RD_PATH	The fully qualified path to the RAID Configuration file	%TK_PATH%\RAID\%MACHTYPE%
RD_FILE	The Name of the RAID configuration file.	raidclon.ini

For RAID capture and deploy jobs, neither variable can be blank.

2.5.3.3 Performing Image Capture and Installation

The imaging sample jobs that are provided only capture a Microsoft Windows operating-system image from a donor server and deploy that image to a target server.

Additional configurations, such as RAID configurations, on the target server might be required before an operating system can be deployed to it. If a RAID configuration is required prior to installing a captured image, use the Configure RAID and Deploy Windows Image sample job to configure RAID prior to deploying the image.

To perform an image capture, the Altiris Agent (AClient.exe) must be installed on the server from which you will clone the installation (donor server). The IBM ServerGuide Scripting Toolkit sample jobs for the scripted installation of operating systems install the corresponding Altiris Agent. If the Scripting Toolkit was not used to perform the initial installation, the applicable installation file must be copied to the target server. Then the file must be executed to install the agent.

The Windows Altiris Agent can also be deployed from the Altiris Deployment Server console. For further information, refer to the Altiris Deployment Server documentation.

Notes:

1. When a target server is deployed using the Image Install method, it will have the Altiris Agent installed, because this is a requirement of the donor server.
2. The default name of the captured windows image is winclone.img. If more than one image will be captured, the default name must be changed to prevent overwriting already captured images.

2.6 Scripts and Utilities

The companion zip file contains the following scripts and win32 utilities.

2.6.1 Scripts

2.6.1.1 BuildWinPE.cmd

Build Windows Preinstallation Environment Script

Usage...

```
BuildWinPE [/Image | /ISO | /RamdiskISO | /Altiris | /?]
```

Where...

/Image	Creates the Windows PE Image. Default.
/ISO	Creates the ISO of the Windows PE Image.
/RamdiskISO	Creates the Ramdisk ISO of the Windows PE Image.
/Altiris	Builds the IBM Altiris Windows PE Image and creates

/? the RAID and Imaging jobs.
 Displays the help.

2.6.2 Toolkit Utilities

2.6.2.1 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
 - 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.

Three 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) 2005. The 32-bit version was formerly named clini32.exe.

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.

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 against the current limits of 255 characters for the Windows version. A message is displayed 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
<i>Filename</i>	Defines the fully qualified path to the INI file to process
<i>filename2</i>	Defines the fully qualified path to an INI file to merge information into from <i>filename</i> . All values in <i>filename</i> are copied into <i>filename2</i> , replacing the value of any preexisting items in <i>filename2</i> .
/ES	Specifies to merge only the items or values in the empty section.
/A	Specifies to append values from items in <i>filename</i> to the items in <i>filename2</i> instead of replacing them. An optional delimiter can be specified using the /D: <i>delimiter</i> parameter.
/U	Specifies to uniquely append values from items in <i>filename</i> to the items in <i>filename2</i> instead of replacing them; only if the value doesn't already exist. An optional delimiter can be specified using the /D: <i>delimiter</i> parameter.
/P	Specifies that the data in <i>filename2</i> is persistent. If duplicate items are found, they are not replaced.
/S: <i>section</i>	Specifies the name of the section within the INI file to write or to read
/I: <i>item</i>	Specifies the name of the item within the INI file to write or to read
/V: <i>value</i>	Specifies the value to write to the INI file
/A: <i>value</i>	Specifies the value to append to the existing item in the INI file. The /I parameter is required to use the /A: <i>value</i> parameter.
/U: <i>value</i>	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: <i>value</i> 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: <i>variable</i>	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: <i>variable</i> 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.
/=: <i>string</i>	Verifies that the value of the item is equal to <i>string</i> , returning a value of 0 if true and 100 if false.
/C: <i>string</i>	Verifies that value of the item has <i>string</i> as a substring, returning a value of 0 if true and 100 if false.
/CT: <i>string</i>	Verifies that the value of the item has <i>string</i> 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: <i>delimiter</i> option is specified.
/B: <i>filename</i>	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.

<code>/D:delimiter</code>	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.
<code>/T:n</code>	Specifies the token in a delimited value to set as the specified environment variable, where <i>n</i> 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.
<code>/R</code>	Removes the specified section, item, or value from the INI file. Removing the last item in a section also removes the section.
<code>/CMT</code>	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.
<code>/UCMT</code>	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.
<code>/CC:character</code>	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.
<code>/AI</code>	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.
<code>/N</code>	Deletes an existing INI file and creates a new INI file. This parameter is not valid with the /E parameter.
<code>/NS</code>	Omits spaces around "=" when writing items into INI files. By default, the clini.exe utility concatenates spaces around "=" when writing items.
<code>/O</code>	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 version is limited to 255 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
<pre>clini info.ini /S:Hardware /I:Machine Type /V:8549 /N</pre>	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"
<pre>clini .\info.ini /S:Hardware /I:Machine Name /V:Server1</pre>	Adds the item Machine Name, with a value of Server1, to the existing Hardware section of the info.ini file
<pre>clini info.ini /S:Hardware /I:Machine Type /E:MachineType call CLIniSet.bat</pre>	Reads the Machine Type value from the info.ini file, and stores it as an environment variable called MachineType
<pre>clini info.ini /S:Hardware /I:Machine Type2 /V:%MachineType%</pre>	Writes the value of the environment variable MachineType to the INI file named info.ini, using section Hardware and item Machine Type2
<pre>clini info.ini /S:Hardware /I:Machine Type2 /E:MachineType2 /B:d:\EnvSet1.bat call d:\EnvSet1.bat</pre>	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
<pre>Clini info.ini /S:MySection /E Call cliniset.bat</pre>	This example creates environment variables for all the items found in section MySection.
<pre>Clini info.ini /AI /E /B:setthem.bat Call setthem.bat</pre>	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.
<pre>Clini info.ini /S:MySection /I:MyItem /E Call cliniset.bat</pre>	This example creates an environment variable called MyItem if it exists in the info.ini file.
<pre>Clini info.ini /S:MySection /I:My Item /E /NS Call cliniset.bat</pre>	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.
<p>After running the first five examples above, in sequence, the info.ini file would contain the following information:</p> <pre>[Hardware] Machine Type = 8549 Machine Type2 = 8549 Machine Name = Server1</pre> <p>Also, two new environment variables would be created as indicated below:</p> <pre>MachineType = 8549 MachineType2 = 8549</pre>	
<pre>clini info.ini /S:User /I:Name /V:Toolkit /N clini info.ini /S:User /I:Name /A: User</pre>	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:

<pre> or clini info.ini /S:User /I:Name /V:Toolkit /N clini info.ini /S:User /I:Name /A:User /D:" "</pre>	<pre> [User] Name = Toolkit User</pre>
<pre> 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:,</pre>	<p>Creates a new file named info.ini with a comma delimited list of values. The resulting info.ini file contains:</p> <pre> [Section] Item = Value1,Value2,Value3,Value2</pre>
<pre> 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:,</pre>	<p>Creates a new file named info.ini with a comma delimited list of unique values. The resulting info.ini file contains:</p> <pre> [Section] Item = Value1,Value2,Value3</pre>
<pre> clini info.ini /S:Section /I:Item /E:MyEVariable /T:2 or clini info.ini /S:Section /I:Item /E:MyEVariable /T:2 /D:,</pre>	<p>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:</p> <pre> Set MyEVariable=Value2</pre>
<p>Content of doit.bat:</p> <pre> @Echo off clini info.ini /S:Section /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</pre>	<p>This example creates a file called info.ini with the following content:</p> <pre> [Section] Item = Value1</pre> <p>Then it checks to see if the value of Item in [Section] is equal to Value1 and displays a message.</p> <p>After running doit.bat, the follow message is displayed:</p> <pre> It's true</pre>
<p>Content of doit.bat:</p> <pre> @Echo off clini info.ini /S:Section /I:Item /V:Value1 /N</pre>	<p>This example creates a file called info.ini with the following content:</p> <pre> [Section] Item = Value1</pre>

<pre> 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 </pre>	<p>Then it checks to see if the value of Item in [Section] contains substring alu and displays a message.</p> <p>After running doit.bat, the follow message is displayed:</p> <p>It's true</p>
<p>Content of doit.bat:</p> <pre> @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 </pre>	<p>This example creates a file called info.ini with the following contents:</p> <pre> [Section] Item = V1,V2,V3 </pre> <p>Then it checks to see if the value of Item in [Section] contains token v2 in a comma delimited list and displays a message.</p> <p>After running doit.bat, the follow message is displayed:</p> <p>It's true</p>
<pre> Clni info1.ini info2.ini </pre>	<p>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.</p>
<pre> Clni info1.ini info2.ini /P </pre>	<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/Values are copied over from info1.ini.</p>
<pre> Clni info1.ini info2.ini /S:MySection </pre>	<p>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 might already exist in the Section called MySection in info2.ini.</p>
<pre> Clni info1.ini info2.ini /S:MySection /I:MyItem </pre>	<p>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.</p>
<pre> Clni info1.ini info2.ini /ES </pre>	<p>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</p>

	section in info2.ini.
<code>Clini info1.ini info2.ini /A</code>	This example appends all the Values from the Sections and Items from info1.ini to info2.ini.
<code>Clini info1.ini info2.ini /U</code>	This example unique 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.
<code>Clini info1.ini info2.ini /U /D:</code>	This example unique appends all the Values from the Sections and Items from info1.ini to info2.ini using a comma as the delimiter if the Value doesn't already exist in info2.ini.
<code>Clini info.ini /V:My Ini Line /CMT</code>	This example comments out the line <i>My Ini Line</i> in the empty section in the info.ini file with a semicolon if the line exists.
<code>Clini info.ini /S:MySection /V:My Ini Line /UCMT</code>	This example uncomments the line <i>My Ini Line</i> in the <i>MySection</i> section of the info.ini if the line exists.
<code>Clini info.ini /I:MyItem /CMT</code>	This example comments out the line indicated by the item <i>MyItem</i> in the empty section of the info.ini file if the item exists.
<code>Clini info.ini /S:MySection /I:MyItem /CMT /CC:#</code>	This example comments out the line indicated by the item <i>MyItem</i> in the section <i>MySection</i> in the info.ini file with a # sign if the item exists.
<code>Clini info.ini /s:MySection /AI /V:My Value /CMT</code>	This example comments out the lines indicated by any item that has a value of <i>My Value</i> of all the items in the section <i>MySection</i> in the info.ini file if the item exists.
<code>Clini info.ini /s:MySection /CMT</code>	This example comments out the section header indicated by <i>MySection</i> in the info.ini file if the section exists.

2.6.2.2 DDCopy.exe

The Device Driver Copy (ddcopy.exe) utility can perform the following functions:

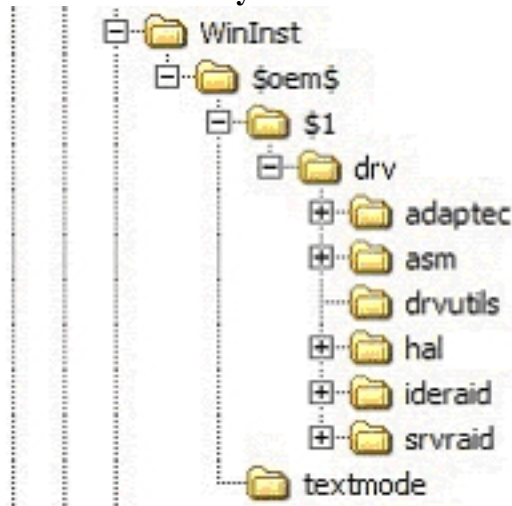
- Copy only those drivers in the driver set that support a specific machine.
- Copy network drivers required for Microsoft ADS.

Three 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) 2005 (32-bit)

The following graphic shows the Microsoft required directory structure for unattended installation:

Microsoft directory structure for unattended installation



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:\w2k_drv\%oem%\$1\drv` directory.

When you issue a `ddcopy` command against a directory that contains device-driver directories (for example, `c:\w2k_drv\%oem%\$1\drv`), `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 32-bit version of `ddcopy` does not run on DOS.

You can use Windows long file names with the 32-bit version of `ddcopy`.

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

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

Parameter	Description
<i>source_path</i>	Defines the fully qualified path to the directory that contains the device-driver directories and the drvset.ini file.
<i>destination_path</i>	Specifies the fully qualified path of the target directory for copying the device drivers.
<i>/M:machine_types</i>	Specifies machine types to limit the number of drivers that are copied. Multiple machine types are allowed when delimited by commas.
<i>/C:category</i>	<p>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:</p> <ul style="list-style-type: none"> • Network • Video • Management • Chipset • Mass Storage • Application • Tape • Hotfix. <p>This is only valid with driver sets from ServerGuide 7.4.12 or greater.</p>
<i>/ADS</i>	Copies only those drivers required for Microsoft Automated Deployment Services (ADS). For driver sets from ServerGuide 7.4.12 or higher, this function is also performed by the <i>/C:Network</i> parameter.
<i>/V:n</i>	<p>Specifies the verbose level used to report status during the deployment process. Valid values for n are:</p> <ul style="list-style-type: none"> • 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 not found
5	Destination is read-only
6	File not found

The following examples illustrate ddcopy.exe utility usage.

Example	Description
<code>ddcopy d:\drivers\%oem%\\$1\drv c:\wininst\%oem%\\$1\drv</code>	Copies all the drivers from the d:\drivers\%oem%\\$1\drv directory to the c:\wininst\%oem%\\$1\drv directory
<code>ddcopy d:\drivers\%oem%\\$1\drv c:\wininst\%oem%\\$1\drv /M:8832</code>	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.
<code>ddcopy d:\drivers\%oem%\\$1\drv c:\wininst\%oem%\\$1\drv /M:8832,8865</code>	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.
<code>ddcopy d:\drivers\%oem%\\$1\drv c:\mydrvs /m:8832 /c:network,video</code>	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.
<code>ddcopy d:\drivers\%oem%\\$1\drv "c:\program files\microsoft ads\nbs\repository\user\presystem" /C:Network</code>	Copies all the drivers with category = Network from the d:\drivers\%oem%\\$1\drv directory to the c:\program files\microsoft ads\nbs\repository\user\presystem directory.

2.6.2.3 DScan.exe

The Driver Scan utility can perform the following functions:

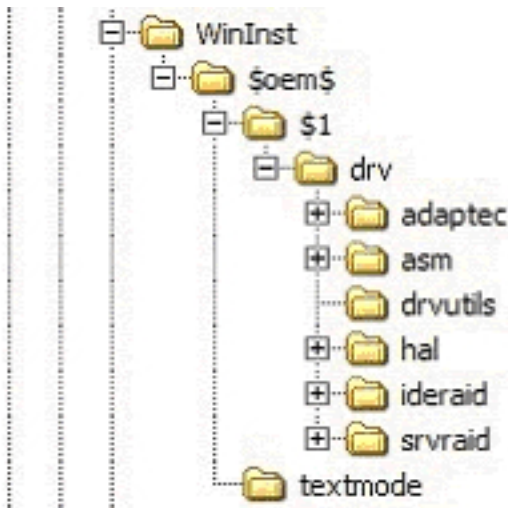
- Scan a device driver or set of device drivers to determine the installation mode (text mode, Plug and Play, or executable) and write this information to the drvinfo.ini file that is located in each device-driver directory. The drvinfo.ini file is used by the unattend.exe command during the installation 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.

One versions of the utility comes with this package:

- A 32-bit version for Windows 32-bit operating systems and for the Windows Preinstallation Environment (Windows PE) 2005 (32-bit)

The following graphic shows the Microsoft required directory structure for unattended installation:

Microsoft directory structure for unattended installation



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:\w2k_drv\%oem%\$1\drv` directory.

When you issue a `dscan` command against a directory that contains device-driver directories (for example, `c:\w2k_drv\%oem%\$1\drv`), `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 32-bit version of `dscan` does not run on DOS.

You can use Windows long-file names with the 32-bit version of dscan..

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 [UNATTEND.EXE](#) 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] [/H:filename  
[/OW] [/V:n] [/W:n] [/O:file_name] [/?]
```

Parameter	Description
<i>driver_path</i>	<p>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.</p> <p>If <i>driver_path</i> has \$oem\$ in the path, the Driver Scan utility creates the \$oem\$\textmode directory and merges the text mode device drivers.</p> <p>If the /SS parameter is used, the path is assumed to be the path to a single device driver.</p>
/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[: <i>path</i>]	<p>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.</p> <p>If <i>path</i> is specified, the text mode device drivers are merged to the specified path. Otherwise, the <i>driver_path</i> parameter must have \$oem\$ in the path so that the text mode device drivers are merged into the \$oem\$\textmode directory.</p>
/M: <i>machine_type</i>	Specifies a machine type, where <i>machine_type</i> is the machine

	type 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 the /M: <i>machine_type</i> parameter.
/H: <i>filename</i>	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 driver 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: <i>n</i>	Specifies the verbose level used to report status during the deployment process. Valid values for <i>n</i> are: <ul style="list-style-type: none"> • 0 - quiet mode • 3 - default • 5 - maximum information
/W: <i>n</i>	Specifies the version of Microsoft Windows for the device drivers: <ul style="list-style-type: none"> • 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: <i>file_name</i>	Combines the information in the DrvInfo.Ini files into a single file specified by the <i>file_name</i> 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

<code>dscan c:\drv\mydriver /SS</code>	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
<code>dscan c:\w2\%oem%\\$1\drv /T</code>	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
<code>dscan c:\drivers /T:c:\other\textmode</code>	Builds the text mode directory in c:\other\textmode using device drivers found in c:\drivers, but does not create drvinfo.ini files

2.6.2.4 HWDetect.exe (w/ hwdet.sys)

The hwdetect.exe utility performs basic hardware detection functions and more advanced PCI-device detection functions on the target server. You can store the information returned by the hwdetect.exe utility in an output file. You can also return values that set the errorlevel Windows environment variable.

The following table lists the operating systems supported by versions of hwdetect.exe.

Operating systems supported by the versions of hwdetect.exe

Version	Operating system
32-bit	The 32-bit version of Windows Preinstallation Environment 2005

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.

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: <ul style="list-style-type: none"> • 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

	<p>variable=value format. The return value is 0 when successful.</p> <p>You can use the > output-redirect option to save the output to an output file.</p>
/i	<p>Displays all hardware information for the target server in an INI-file format. The return value is 0 when successful.</p> <p>You can use the > output-redirect option to save the output to an output file.</p>
/m: <i>type</i>	<p>Compares the machine type of the target xSeries server to the specified machine type, <i>type</i>. The return values are:</p> <ul style="list-style-type: none"> • 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: <i>vendor_id</i>	Determines whether there is a PCI adapter in the target server that matches the specified vendor ID, where <i>vendor_id</i> is a hexadecimal value. The return value is the number of matching adapters in the target server, or 0 if there are no matches.
/did: <i>device_id</i>	Determines whether there is a PCI adapter in the target server that matches the specified device ID, where <i>device_id</i> 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: <i>sub_vendor_id</i>	Determines whether there is a PCI adapter in the target server that matches the specified sub-vendor ID, where <i>subvendor_id</i> 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: <i>sub_device_id</i>	Determines whether there is a PCI adapter in the target server that matches the specified sub-device ID, where <i>subdevice_id</i> 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: <i>bus_number</i>	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: <i>device_number</i>	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: <i>num</i>	Adds an integer number, <i>num</i> , 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 <i>num</i> .

The following examples illustrate hwdetect.exe utility usage.

Example	Description
<pre> 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 </pre>	<p>Determines if the target server is an IBM server or not, and branches accordingly to perform equipment-specific steps</p>
<pre> 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 </pre>	<p>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</p>
<pre> ----- myscript.sh ----- ----- #!/bin/sh ./hwdetect /m:8676 # Check the return code of the last command, # stored in '\$?' if ["\$?" -eq "1"]; then # Perform 8676 specific action here... echo "Inside machine type 8676 section" else ./hwdetect /m:8669 # Check the return code of the last command, </pre>	<p>This is a Linux shell script that determines if the target server is either a machine type 8676 or machine type 8669 server, and branches accordingly to call a system-specific action or to display a message of non-support for other machine types.</p>

<pre> # stored in '\$?' if ["\$?" -eq "1"]; then # Perform 8669 specific action here... echo "Inside machine type 8669 section" else # Requested machine type not found. echo "System not supported!" fi fi </pre>	
<pre> hwdetect /i>hwdetect.out clini hwdetect.out /S:CI /I:Vendor_ID.0 /E:Vendor CLIniSet.bat </pre>	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 an environment variable accordingly
<pre> hwdetect /vid:0x9005 /did:0x0250 if errorlevel 1 call 6Mstuff.bat </pre>	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
<pre> 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 </pre>	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
<pre> hwdetect /i>hwdetect.out </pre>	Displays hardware configuration information about the target server. The >hwdetect.out parameter is an 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.
8. The vendor, device, subvendor, and subdevice values are in hexadecimal notation.

2.6.2.5 LEcho.exe

The Logging Echo utility can perform 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.

Only one version of the Logging Echo utility is bundled with the IBM Windows PE Tools:

- A 32-bit version for use on Microsoft Windows 32-bit operating systems and the 32-bit version of Windows Preinstallation Environment (Windows PE) 2005.

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

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

When it is initialized, the LEcho.exe program 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.

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	Message to display to the screen/log file.																														
/F	<p>Formats the message using the following variables:</p> <table> <tr> <th>Variable</th><th>Definition</th></tr> <tr> <td>%d or %nd</td><td>System date where n indicates the format type.</td></tr> <tr> <td>0 =</td><td>Sun 12/31/2006 (Default)</td></tr> <tr> <td>1 =</td><td>Sunday, December 31, 2006</td></tr> <tr> <td>2 =</td><td>Sun, Dec 31, 2006</td></tr> <tr> <td>3 =</td><td>Dec 31, 2006</td></tr> <tr> <td>4 =</td><td>12-31-2006</td></tr> <tr> <td>5 =</td><td>12/31/2006</td></tr> <tr> <td>6 =</td><td>2006-12-31</td></tr> <tr> <td>7 =</td><td>2006-Dec-31</td></tr> <tr> <td>8 =</td><td>2006-December-31</td></tr> <tr> <td>9 =</td><td>20061231</td></tr> </table> <table> <tr> <td>%t or %nt</td><td>System time where n indicates the format type.</td></tr> <tr> <td>0 =</td><td>16:12:13 (Default)</td></tr> <tr> <td>1 =</td><td>04:12:13 PM</td></tr> </table>	Variable	Definition	%d or %nd	System date where n indicates the format type.	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	System time where n indicates the format type.	0 =	16:12:13 (Default)	1 =	04:12:13 PM
Variable	Definition																														
%d or %nd	System date where n indicates the format type.																														
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	System time where n indicates the format type.																														
0 =	16:12:13 (Default)																														
1 =	04:12:13 PM																														

Parameter	Description
	\a Alert (bell) \b Backspace \f Formfeed \n Newline \r Carriage return \t Horizontal tab
/R:n	Repeats the message n times.
/E:number	Display the error message and sets the system errorlevel to number.
/P	Pauses until a key is pressed.
/P:n	Pauses for n seconds 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	Create a new, blank log file even if it exists already.
/LO	Write the message to the log file only.
/DO	Write the message to the display only.
/?	Display a help message containing the application syntax.

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

Value	Description
0	Success or true
1	Syntax error
5	Log file cannot be written to.
100	False
255	Program error

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	Sends the text "Pausing for 30 seconds," and a countown

Example	Description
<code>seconds." /P:30</code>	timer beginning at 30. This countdown can be bypassed by pressing any key.
<code>LEcho "Running a 30 second timer." /T:30 /SC /SO</code>	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.

2.6.2.6 PRaid.exe

The ServerGuide Scripting Toolkit supports Policy-based RAID configuration and replication using the PRAID.EXE utility. Some features of PRAID.EXE include:

- A single user interface for configuring and replicating RAID controller settings.
- The ability to use the PRAID policies file to describe how your RAID controllers should be configured or replicated.
- Customizable logic to determine what configuration to use with which controllers. This logic can include the machine type of the server, the number of drives connected to the controller, and the RAID controller type.
- An AUTO mode to configure using default settings.
- The ability to configure all RAID controllers in a system with a single program call.
- Features to save useful information about each captured configuration, including machine type, date, and time of capture.
- The ability to restore all controllers to factory-default settings.

When used to configure RAID controllers, PRAID accepts a PRAID policies file as input and uses this information to execute the RAID utilities that are capable of configuring all supported RAID controllers with the specified parameters. Optionally, if you do not wish to use a policies file, you can instruct PRAID to configure the RAID controllers using default values for arrays and logical drives.

When used to capture RAID controller settings, PRAID creates or appends to a PRAID policies file that contains the RAID controller settings which can later be used to configure RAID controllers with the same hardware configuration.

PRAID has three modes of operation:

Configure mode

for scripted configuration of RAID controllers. Run the output-script file created by PRAID to configure the RAID controllers. See [Output-script file](#) for details.

Attention: When used in configure mode, PRAID restores all RAID controllers in the server to factory-default settings before configuring any of the RAID controllers. If you do not have a backup of data for all drives, it is recommended that you create backups before running PRAID on your server.

Capture mode

For replicating RAID controller settings.

Restore-defaults mode

For resetting RAID controllers to factory default settings only.

2.6.2.6.1 Environment requirements

The PRAID utility supports the following RAID controllers:

- ServeRAID-4H
- ServeRAID-4Mx
- ServeRAID-4Lx
- ServeRAID-5i
- ServeRAID-6M
- ServeRAID-6i/6i+
- ServeRAID-7k
- ServeRAID-7t
- ServeRAID-7e SATA
- ServeRAID-7e SCSI
- ServeRAID-8i
- ServeRAID-8k
- ServeRAID-8k-l
- ServeRAID-8e SATA
- ServeRAID-8e SAS
- LSI Integrated SCSI-RAID controller
- LSI Integrated SAS-RAID controller (1064/1064E/1068)
- LSI IDEal RAID controller
- LSI MegaRAID SAS controller

PRAID works by scripting the interfaces of other RAID-configuration utilities. In order to accomplish this, several utilities must be available to PRAID when it is running:

arcconf: ServeRAID-7t,8i,8k,8k-l

cfggen: LSI 1020/1030,LSI 1064x

hrconf: ServeRAID-7e,8e

hyperwin: IDE RAID

ipssend: ServeRAID-4H,4Mx,4Lx,5i,6M,6i,6i+,7k

megacli: LSI MegaRAID

These utilities must be in the system search path. If not, specify their location by using the /p parameter when invoking the PRAID.EXE utility. All of these utilities are included with the ServerGuide Scripting Toolkit.

2.6.2.6.2 Usage

Each mode of PRAID operation requires a different syntax, as shown:

Configure mode

```
PRAID.EXE /f:policies /d /o:outScript /p:path /t:temp /e1 /e2
/e3 /s:l,n /v:n /y /b
```

Capture mode

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

Restore defaults mode

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

In addition, you can supply the PRAID parameters in a parameters file, PRAID @parameters_file, instead of using the command line.

Configure mode parameters

Parameter	Description
/d	Configures all controllers in the system using default settings for arrays and logical drives instead of using a policies file. The default settings used are the same as the default settings for the policies file. See Default settings for RAID controllers for a list of default values for each RAID controller. You cannot use this parameter with the /f parameter.
/e1	Returns an error code of 1 if one or more controllers are not configured because there was no policy found to configure the controllers. This parameter cannot be used with the /a parameter.
/s:l,n	Calls the SAVESTAT program in the output-script file error checking. This call will cause the value n to be written to SAVESTAT location l anytime one of the RAID configuration utilities in the output-script file returns an error condition. To use this parameter, SAVESTAT must be located in the system search path.
/b	<p>Automatically create commands in the output-script file that will build the arrays if it is necessary before writing to the drives. PRAID makes the building process optional because it can take up to 45 minutes. If you do not supply the /b parameter, PRAID will prompt you to see if you want the array built as part of the output-script file. If you supply the /b parameter and the arrays do not need to be built before using them, then this parameter has no effect.</p> <p>RAID-1 arrays created on ServeRAID-7e SCSI controllers must be built before they can be used for data.</p>
/f:policies	<p>Required for configure (unless /d is used) and capture. The path and file name of the policies file:</p> <ul style="list-style-type: none"> In configure mode, this points to the policies for PRAID to use when configuring the RAID controllers. 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.
/y	Do not prompt 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.
<code>/p:path</code>	The full path to where the RAID configuration utilities are located (IPSSSEND, CFG1030, RAIDSEL, ACU, ACUSAS, ACUICHSV, and HYPERCFG). You do not need to specify this parameter if all of the RAID configuration utilities are already in the system search path, which is recommended.
<code>/e2</code>	Returns an error code of 2 if there are no supported RAID controllers found in the system. By default, PRAID does not return an error if no controllers are found in the system.
<code>/e3</code>	Returns an error code of 3 if at least one controller is found with no drives attached. By default, PRAID does not return an error if no drives are attached to a RAID controller. This parameter cannot be used with the <code>/a</code> parameter.
<code>/v:n</code>	The verbosity level, where n is 0 (quiet), 3 (default), or 5 (maximum).
<code>@parameters_file</code>	The path and file name of a file containing the command line parameters. Parameters should be separated by space characters. If you use this parameter, any other parameters that you specify on the command line will override the parameters that you supply in the params file.

Capture mode parameters

Parameter	Description
<code>/c:p</code>	<p>Indicates capture mode. The <code>:p</code> portion is optional. If you do not include the optional portion, then <code>:p</code> will assume the default value: <i>t,d</i>. You can use <code>:p</code> to provide a list of parameters describing the <i>AppliesTo</i> that should be created when capturing the parameters to a policy.</p> <p><code>:p</code> is a list containing any of the following:</p> <ul style="list-style-type: none"> • m - use the machine type of the system in the <i>AppliesTo.1</i> entry for the policy. • s - use the serial number of the system in the <i>AppliesTo.1</i> entry for the policy • t - use the type of the RAID controller in the <i>AppliesTo.1</i> entry for the policy. • c - use the controller number (scan order relative to all other RAID controllers in the system) in the <i>AppliesTo.1</i> entry for the policy. • d - use the number of drives connected to the RAID controller in the <i>AppliesTo.1</i> entry for the policy.
<code>/f:policies</code>	Required for configure (unless <code>/d</code> is used) and capture. The

	<p>path and file name of the policies file:</p> <ul style="list-style-type: none"> • In configure mode, this points to the policies for PRAID to use when configuring the RAID controllers. • 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.
<code>/y</code>	Do not prompt 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.
<code>/p:path</code>	The full path to where the RAID configuration utilities are located (IPSSSEND, CFG1030, RAIDSEL, ACU, ACUSAS, ACUICHSV, and HYPERCFG). You do not need to specify this parameter if all of the RAID configuration utilities are already in the system search path, which is recommended.
<code>/e2</code>	Returns an error code of 2 if there are no supported RAID controllers found in the system. By default, PRAID does not return an error if no controllers are found in the system.
<code>/e3</code>	Returns an error code of 3 if at least one controller is found with no drives attached. By default, PRAID does not return an error if no drives are attached to a RAID controller. This parameter cannot be used with the <code>/a</code> parameter.
<code>/v:n</code>	The verbosity level, where n is 0 (quiet), 3 (default), or 5 (maximum).
<code>@parameters_file</code>	The path and file name of a file containing the command line parameters. Parameters should be separated by space characters. If you use this parameter, any other parameters that you specify on the command line will override the parameters that you supply in the params file.

Restore defaults mode parameters

Parameter	Description
<code>/r</code>	Restores all RAID controllers to factory-default settings and then returns immediately. No RAID configuration is done if you use this parameter.
<code>/f:policies</code>	<p>Required for configure (unless <code>/d</code> is used) and capture. The path and file name of the policies file:</p> <ul style="list-style-type: none"> • In configure mode, this points to the policies for PRAID to use when configuring the RAID controllers. • 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.
/y	Do not prompt 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.
/p:path	The full path to where the RAID configuration utilities are located (IPSSSEND, CFG1030, RAIDSEL, ACU, ACUSAS, ACUICHSV, and HYPERCFG). You do not need to specify this parameter if all of the RAID configuration utilities are already in the system search path, which is recommended.
/e2	Returns an error code of 2 if there are no supported RAID controllers found in the system. By default, PRAID does not return an error if no controllers are found in the system.
/e3	Returns an error code of 3 if at least one controller is found with no drives attached. By default, PRAID does not return an error if no drives are attached to a RAID controller. This parameter cannot be used with the /a parameter.
/v:n	The verbosity level, where n is 0 (quiet), 3 (default), or 5 (maximum).
@parameters_file	The path and file name of a file containing the command line parameters. Parameters should be separated by space characters. If you use this parameter, any other parameters that you specify on the command line will override the parameters that you supply in the params file.

2.6.2.6.3 Usage examples

Configure mode examples

Example	Description
PRAID /d /y	<ul style="list-style-type: none"> Does not prompt before setting controllers to the factory-default settings. Performs drive synchronization, when required, without prompting. Configures the RAID controller automatically on exit. <p>This example is especially useful for unattended scripted installs.</p>
PRAID /f:policies.ini /v:5 /e1	<ul style="list-style-type: none"> Configures the RAID controllers in the system using the policies file <code>policies.ini</code>. Sets the verbosity to maximum. Returns an error code if there are no matching policies for one or more controllers. See the Return codes

	section for more information.
--	-------------------------------

Capture mode examples

Example	Description
PRAID /c /f:c:\mydata\policies.ini	Captures the configuration of all RAID controllers into the file c:\mydata\policies.ini. The c:\mydata\policies.ini file must exist before running this command.
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. Uses the \temp directory for temp space. The \temp directory must exist before running this command.

Restore defaults mode examples

Example	Description
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.

Running PRAID using a parameters file

Example	Description
PRAID @c:\sgtk\params.txt	Runs PRAID using the parameters located in the c:\sgtk\params.txt file.

2.6.2.6.4 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.
- **7** - Error resetting a controller to the default settings.
- **8** - Error gathering information about a controller.
- **9** - For all other errors.

2.6.2.6.5 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.

A policies file can be created using any of the following methods:

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. The example files are located in the \PolicyFiles directory.
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 using the format [Policy.name] where *name* is a unique user-assigned name that is used to identify 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.

2.6.2.6.5.1 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.

2.6.2.6.5.1.1 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 their 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. These example policies files are located in the \PolicyFiles directory:

- 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 policies file.

Policy file parameters

Keyword	Required?	Default	Description
<i>Policy.name</i>	Yes	None	This header designates the start of a new policy. See Policy.name for additional information.
<i>AppliesTo.n</i>	Yes	None	Use this parameter to describe when the current policy should be chosen to configure the RAID controllers. See AppliesTo.n for additional information.
<i>ReadAhead</i>	No	<ul style="list-style-type: none"> • ADAPTIVE (for ServeRAID 4H, 4MX, 4Lx, 5i, 6i, 6i+, 6M, 	Specifies the read ahead setting that should be applied to the RAID controller. See ReadAhead for additional information.

		<ul style="list-style-type: none"> and 7k) • ON (for ServeRAID -7t and 8i, 8k, and 8k-l) 	
RebuildRate	No	HIGH	Specifies the rebuild rate that should be applied to the RAID controller. See RebuildRate for additional information.
StripeSize	No	<ul style="list-style-type: none"> • 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 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 for additional information.
Array_Defaults	No	<ul style="list-style-type: none"> • 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 	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 for additional information.

		arrays has 4 or more physical drives <ul style="list-style-type: none"> • 0%:0 for all other cases 	
<code>Array.letter</code>	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 for additional information.
Hotspares	No	None	Defines a list of specific physical drives to designate as hot-spare drives. See Hotspares for additional information.
Logical_Mode	No	AUTO	Defines the logical-drive creation policy to use when creating logical drives. See Logical_Mode 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 for additional information.
<code>Logical.num</code>	No	None	Lets you specify how many logical drives are created and the specific parameters for each logical drive. See Logical.num for additional information.

2.6.2.6.5.1.1.1 Policy.name

Description

This 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.

Examples

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

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 following descriptive names :
 - SERVERAID-4H
 - SERVERAID-4Mx
 - SERVERAID-4Lx
 - SERVERAID-5i
 - SERVERAID-6i (for ServeRAID-6i and 6i+)
 - SERVERAID-6M
 - SERVERAID-7t
 - SERVERAID-7e-SCSI
 - SERVERAID-7e-SATA
 - SERVERAID-7k
 - SERVERAID-8i
 - SERVERAID-8k
 - SERVERAID-8k-1
 - ServeRAID-8e-SATA
 - ServeRAID-8e-SAS
 - LSI-SCSI-RAID
 - LSI-SCSI-RAID (1064/1064E/1068)
 - LSI-IDEal-RAID
 - LSI-SAS-RAID
 - LSI-MegaRAID
- *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
```

2.6.2.6.5.1.1.3 ReadAhead

Description

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](#) for the list of ReadAhead settings supported by PRAID for each RAID controller. Possible settings are:

- Adaptive
- On
- Off

Examples

```
ReadAhead = On
```

2.6.2.6.5.1.1.4 RebuildRate

Description

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](#) for the list of RebuildRate settings supported by PRAID for each RAID controller.

- High
- Medium
- Low

Examples

```
RebuildRate = High
```

2.6.2.6.5.1.1.5 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](#) 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
```

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
```

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 can 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
```

2.6.2.6.5.1.1.8 *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
```

2.6.2.6.5.1.1.9 *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
```

2.6.2.6.5.1.1.10 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
```

2.6.2.6.5.1.1.11 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
```

2.6.2.6.5.1.1.12 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](#).

Examples

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

2.6.2.6.6 Supported settings for RAID controllers

The following table 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.

Supported settings for each RAID controller when using PRAID

Bold settings are defaults.					
Controller	Rebuild Rate	Read Ahead	Stripe Size	RAID Levels ¹	Write-cache Mode
ServeRAID-4H	<ul style="list-style-type: none"> HIGH MEDIUM LOW 	<ul style="list-style-type: none"> ADAPTIVE ON OFF 	8,16,32,64	0,1,1E,5,5E AUTO (RAID-x0 not supported)	<ul style="list-style-type: none"> ON OFF AUTO
ServeRAID-4Mx	<ul style="list-style-type: none"> HIGH MEDIUM LOW 	<ul style="list-style-type: none"> ADAPTIVE ON OFF 	8,16,32,64	0,1,1E,5,5E,5EE AUTO (RAID-x0 not supported)	<ul style="list-style-type: none"> ON OFF AUTO
ServeRAID-4Lx					
ServeRAID-6i/6i+	<ul style="list-style-type: none"> HIGH MEDIUM LOW 	<ul style="list-style-type: none"> ADAPTIVE ON OFF 	8,16,32,64	0,1,1E,5,5EE AUTO (RAID-x0 not supported)	<ul style="list-style-type: none"> ON OFF AUTO
ServeRAID-6M					
ServeRAID-7k					
ServeRAID-5i	<ul style="list-style-type: none"> HIGH MEDIUM LOW 	<ul style="list-style-type: none"> ADAPTIVE ON OFF 	8,16,32,64	0,1,1E,5 AUTO (RAID-x0 not supported)	<ul style="list-style-type: none"> ON OFF AUTO
ServeRAID-7t	[n/a]	<ul style="list-style-type: none"> ON OFF 	16, 32, 64	0,1,5,10, VOLUME AUTO	<ul style="list-style-type: none"> ON OFF AUTO
ServeRAID-8i	[n/a]	<ul style="list-style-type: none"> ON OFF 	16, 32, 64 , 128, 256, 512	0,1,10,1E,5,50,5EE, 6,60, VOLUME	<ul style="list-style-type: none"> ON OFF AUTO
ServeRAID-8k	[n/a]	<ul style="list-style-type: none"> ON OFF 	16, 32, 64 , 128, 256, 512	0,1,1E,10,5,5EE, 6,60, VOLUME	<ul style="list-style-type: none"> ON OFF AUTO
ServeRAID-8k-l	[n/a]	<ul style="list-style-type: none"> ON OFF 	16, 32, 64 , 128, 256, 512	0,1,10,AUTO	[n/a]

ServeRAID-8e SAS	[n/a]	[n/a]	16, 32, 64	0, 1, 10, AUTO	[n/a]
ServeRAID-8e SATA	[n/a]	[n/a]	16,32, 64	0, 1, 10, VOLUME, AUTO	[n/a]
ServeRAID-7e SATA	[n/a]	[n/a]	16, 32, 64	0, 1, AUTO	[n/a]
ServeRAID-7e SCSI	[n/a]	[n/a]	16,32, 64	0, 1, 10, AUTO	[n/a]
LSI-1064x SAS	[n/a]	[n/a]	[n/a]	0,1,1E,AUTO	[n/a]
LSI 1020/1030 chipset	[n/a]	[n/a]	[n/a]	1, 1E ² , AUTO	[n/a]
LSI IDEal RAID	[n/a]	[n/a]	32, 64 ,128,256,512,1024,2048,4096	0, 1, AUTO	<ul style="list-style-type: none"> • ON • OFF • AUTO
LSI MegaRAID SAS	[n/a]	[n/a]	8, 16, 32 64, 128	0, 1, 5, 10, AUTO	[n/a]

1. RAID Levels 5E and 5EE only support one logical drive per array.
2. RAID level 1E is supported for the LSI 1030 only on the xSeries model 336.

Note:

The 7e and 8e RAID controllers support one logical drive per array, and ignore the logical drive size, using the FILL setting for the logical drive.

Default RAID levels are described in [Default RAID levels](#).

2.6.2.6.6.1 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*.

Default RAID levels

Number of drives in array	Controller types	RAID level
1	ServeRAID-4H, 4Mx, 4Lx, 5i, 6i, 6M, 7k, LSI-IDEal	0
1	ServeRAID-7t,8e-SATA, 8i, 8k, 8k-l	VOLUME
2	All	1
3	ServeRAID-8e-SATA, 8e-SAS, 8k-l	1

3	ServeRAID-7t	5
3 or more	ServeRAID-7e-SCSI	0
3 or more	ServeRAID-7e-SATA	1
3 or more	ServeRAID-4H, 4Mx, 4Lx, 5i, 6i, 6M, 7k, 8i, 8k, MegaRAID	5
4 or more	ServeRAID-7t, 8e-SATA, 8e-SAS, 8k-I	10

2.6.2.7 SaveStat.exe

The savestat.exe utility enables you to store and retrieve up to five values in CMOS, using persistent-state information, on the target server. This utility is useful to pass information to the deployment process after a restart (reboot) occurs, such as where in the deployment process to continue after the restart.

The 32-bit version of savestat.exe is for use on Microsoft Windows 2000 Server, Windows Server 2003, and the 32-bit version of Windows Preinstallation Environment 2005

Values are returned in the return code in Windows so that you can create a batch file to branch according to the value returned. The savestat.exe utility that comes with the ServerGuide Scripting Toolkit has the following command-line syntax:

```
savestat </setn=value|/getn|/reset>
```

Parameter	Description
/setn=value	Saves an integer value, <i>value</i> , to the <i>n</i> th location in persistent-storage memory, where <i>n</i> can be any number 1-5 and <i>value</i> can be any number 0-254. The return values are: 0 if successful 1 if not successful
/getn	Retrieves a value currently set in the <i>n</i> th location in persistent-storage memory, where <i>n</i> can be any number 1-5. The return value is the number stored, or 255 if not successful.
/reset	Resets all persistent-storage memory to zero values. The return values are: 0 if successful 1 if not successful

The following examples illustrate savestat.exe utility usage.

Example	Description
---------	-------------

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

2.6.2.8 Unattend.exe

2.6.3 Third Party Utilities

2.6.3.1 ArcConf.exe

The acudas.exe utility configures an IBM ServeRAID-7t, IBM ServeRAID-8i, IBM ServeRAID-8k, IBM ServeRAID-8k-l, controller. See the documentation that comes with the IBM ServeRAID controller for more information about the utility.

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

Parameter	Description
COPYBACK	toggles controller copy back mode
CREATE	creates a logical drive
DELETE	deletes one or more logical drives
DRIVERUPDATE	updates a windows device driver
FLASHCOPY	creates a copy of a logical drive
GETCONFIG	prints controller information
GETLOGS	gets controller log information
GETSTATUS	displays the status of running background tasks
GETVERSION	prints version information for all controllers
IDENTIFY	blinks LEDS on device(s) connected to a controller
KEY	installs a Feature Key onto a controller
MODIFY	performs RAID Level Migration or Online Capacity Expansion
RESCAN	checks for new or removed drives
ROMUPDATE	updates controller firmware
SETCONFIG	restores the default configuration
SETCACHE	adjusts physical or logical drive cache mode
SETNAME	renames a logical drive given its logical drive number
SETSTATE	manually sets the state of a physical drive
TASK	performs a task such as synchronize on a physical or logical drive

Example	Description
arcconf.exe create 0 logicaldrive MAX 5 0 0 0 1 0 2	Configures controller number 1 in the target server for RAID5 on the first three drives

2.6.3.2 CFGGen.exe

This program is called by PRAID to configure LSI 1020/1030, LSI 1064x SAS controllers.

The `cfggen.exe` utility configures the RAID controller in IBM eServer and xSeries servers that have an integrated SCSI controller with RAID capabilities based on the LSI 1064x chip set. Configuration information is obtained from a donor server and, after optional modifications, deployed onto one or more target servers.

See the documentation that comes with the server for more information about the utility.

The `cfggen.exe` utility that comes with the ServerGuide Scripting Toolkit has the following command-line syntax:

```
cfggen controller_number command parameters
```

Parameter	Description
<i>controller_number</i>	The controller number is an integer between 0 and 255.
<i>command</i>	The following commands are valid: CREATE Creates an IR volume. DEFAULTS Sets controller configuration to default settings. DISPLAY Displays controller, volume and physical device info. FORMAT Performs a low-level format of a SCSI hard disk drive. HOTSPARE Makes the drive a hot spare. STATUS Displays current volume status info. SETOFFLINE Takes a hard disk off line. VSIME Updates the IMEVolumeSettings parameter in Mfg Pg 4 or in the existing IME volume.
<i>parameters</i>	Issue a command using the following syntax to see parameter help for the command: <i>cfggen controller_number command</i>

The following examples show `cfggen.exe` utility usage:

Example	Description
<code>cfggen 1 create logicaldrive NEWARRAY 52071 1 0 1 0 2 qsync</code>	Creates a new mirrored logical drive of 52 GB on channel 0 and SCSI ID 1 and on channel 0 and SCSI ID2 on controller 1, using quick synchronization

cfggen 1 create logicaldrive NEWARRAY MAX 1 0 1 cfggen setstate 1 0 1 HSP	Creates a logical drive on controller 1, using all available space on the drive, sets RAID1 for channel 0 and SCSI ID 1; then, sets the state to hot spare
--	--

2.6.3.3 HRConf.exe

The hrconf.exe utility configures an IBM ServeRAID-7e-SATA, IBM ServeRAID-7e-SCSI, IBM ServeRAID-8e-SATA, IBM ServeRAID-8e-SAS controller. See the documentation that comes with the HostRAID controller for more information about the utility.

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

Parameter	Description
BACKUP	saves the current configuration to a file
CREATE	creates a logical drive
DELETE	deletes one or more logical drives
DRIVERUPDATE	updates a windows device driver
GETCONFIG	prints controller information
GETSTATUS	displays the status of running background tasks
GETVERSION	prints version information for all controllers
IDENTIFY	blinks LEDS on device(s) connected to a controller
RESCAN	checks for new or removed drives
RESTORE	restores a configuration from a saved file
ROMUPDATE	updates controller firmware
SETBOOT	marks or unmarks a logical drive bootable
SETCONFIG	restores the default configuration
SETSTATE	manually sets the state of a physical or logical drive
TASK	performs a task such as synchronize on a logical drive

Example	Description
hrconf.exe create 0 logicaldrive MAX 1 0 0 0 1	Configures controller number 1 in the target server for RAID1 on the first two drives

2.6.3.4 HyperWin.exe

This program is called by PRAID to configure the LSI-IDEal RAID controller where available on BladeCenter HS20 systems. The hyperwin.exe utility configures an LSI IDEal RAID controller in an IBM eServer BladeCenter HS20. The ServerGuide Scripting Toolkit uses this utility during the deployment process to configure the LSI IDEal RAID controller. See the IBM BladeCenter HS20 LSI IDEal RAID User's Guide for information about this utility

The hypercfg.exe utility that comes with the ServerGuide Scripting Toolkit has the following command-line syntax.

/A[m] [/!] [/2] - Automatically Configures arrays for Raid 0,1 & 10

- Where 'm' is RAID Mode for configuring.
- When 'm' = S[n], Arrays Configured for Raid 0 mode,'n'- Stripe size.
- Allowed Stripe Sizes are 32k,64k,128k,256k,512k,1024k,2048k,4096k.
- When /! option is used, drives are configured in special RAID 0 mode
- i.e., even when 2 drives are present only 1 drive is configured in
- RAID 0 mode. This option is valid only when 2 drives are present.
- When /2 option is used, 2MB size reservation is enabled
- When 'm' = M, Arrays configured for Raid 1 mode.
- When 'm' = R[n], Arrays configured for Raid 10 mode.'n'- Stripe size.
- Allowed Stripe Sizes are 32k,64k,128k,256k,512k,1024k,2048k,4096k.

- Default Option: RAID 0 Mode with 64K Stripe Size.

/O<oo> - Sets arrays with the Options.

- Where 'oo' is the option flag.
- When 'oo' = W, Switches ON Write Cache.
- When 'oo' = B, Switches ON Boot sector virus protection.
- When 'oo' = D, Switches ON DMA transfers.
- When 'oo' = P, Switches ON Silent mode.
- When 'oo' = M, Switches ON R0 to R1 promotion.
- When 'oo' = R[w/b/d/p/m], Specified flags are reset
- When R is used with no flags, all flags are reset

/C<n> - Selects card where the drive is present.

- Where 'n' is Card index starting from 0,1,2..and so on.

/D<cd> - Selects drive for R, I, L, E, P Options.

- Where 'c' is Channel ID taking values 0 or 1
- Where 'd' is Device ID taking values 0 or 1.

/R<n> [/C<n>] [/D<cd>] [/\$[FILENAME]] [/B[FILENAME]] - Reads & Displays 'n'th

- sector of specified drive.
- /\$[FILENAME] dumps a specified BIN file to the 'n'th sector
- /B[FILENAME] dumps the 'n'th sector to BIN file
- Default filename is HYPERCFG.BIN

/I[a] [/C<n>] [/D<cd>] [/B[filename]] - Displays Identify device packet

- of specified drive.
- /Ia displays ID Device Packet for all drives present.
- /B[filename] - specifies to dump the 512 byte identify data to filename.

- /B option shouldn't be specified with /Ia switch

Press ENTER to continue...

/L [/C<n>] [/D<cd>] [/b[filename]] - Displays the configuration sector of specified drive.

- using only /L displays the config. sector from first available
- Drive.
- Use /bfilename to dump the IRCD as bin file
- filename is not specified, HYPERCFG.BIN is default

/W[o] - Prints the sector dump for R, I, E options.

- Where 'o' specifies option for printing.
- When 'o' is v, Prints the configuration in Verbose mode.

/E<o> [/C<n>] [/D<cd>] - Erases & Displays the config. sector of specified drive

.

- Where 'o' specifies option for erasing.
- when 'o' = c, Erases config sector only.
- when 'o' = e, Erases Error log sector only.
- when 'o' = a, Erases both config & Error log sectors.

/F<fn> - Redirects the output to a file 'fn'.

- Default filename is 'hypercfg.cfg'.

/P[s] [/C<n>] [/D<cd>] - Switches the drive to specified Power state.

- Where 's' = S, the drive is powered on in Suspend state.
- Default: the drive is powered on in Active state.

/S - Executes in Silent mode.

/V - Displays the RAID bios version.

/X - Pause execution on encountering an error condition.

Press ENTER to continue...

/M[R/<MaxAddress>] [/C<n>] [/D<cd>] - This option sets the max. user

- accessible sector address for specified drive.
- Address is specified in HEX
- /Mr resets the Max address to native max address of specified drive

/@[tfn] /b[bfn] - Configures RAID from specified file .

- /@[tfn] - config from text file tfn Default File:hypercfg.cfg

- /@ /b[bf] - config from bin file bf Default File:hypercfg.bin
- /Z[e/d] - Enables/Disables SMART feature. Default is Enable
- getstatus n - Displays the Status of the Logical Drives in the given adapter if any of the drive is in rebuilding state.
- where 'n' is the adapter number (1,2,3,4).

Example	Description
hyperwin.exe /A /M /D01 /S	Configures controller number 1 in the target server for RAID1 on the first two drives

2.6.3.5 IPSSend.exe

The ipssend.exe utility is called by PRAID to configure the following IBM ServeRAID SCSI controllers:

- IBM ServeRAID 4H
- IBM ServeRAID 4Lx
- IBM ServeRAID 4Mx
- IBM ServeRAID 5i
- IBM ServeRAID 6i
- IBM ServeRAID 6i+
- IBM ServeRAID 6M
- IBM ServeRAID 7k

The ipssend.exe utility can restore a controller to factory-default settings, or create and configure a RAID array.

See the documentation that comes with the RAID controller for information about this utility. The following are examples of ipssend.exe utility usage.

Example	Description
ipssend create 1 logicaldrive NEWARRAY MAX 5 noprompt	Creates a new RAID5 array on the first controller, using the maximum available space
ipssend backup 1 c:\myraid.cfg noprompt	Creates a backup of the configuration settings on the first controller to a file named myraid.cfg
ipssend restore 1 c:\myraid.cfg noprompt	Uses a file named myraid.cfg to restore configuration settings to the first controller
ipssend setconfig 1 default	Resets the first controller to factory-default settings

3 Known Limitations & Bugs

The following section outlines known limitations and bugs when configuring RAID using Windows Preinstallation Environment.

Symptom:

Windows PRAID fails with error code 11 for LSI-SAS and LSI-SCSI controllers

Resolution:

The Windows CFGGEN utility fails with error code 11 during PRAID configuration for LSI SCSI-RAID controller and LSI SAS RAID controller during the cfggen create command when a previous cfggen delete command was issued without a reboot between the delete and create commands. To avoid this problem, delete any existing configuration for the affected controllers prior to running Windows PRAID to configure the controllers.

Symptom:

PXE booting Windows PE from Altiris is extremely slow on IBM BladeCenter servers.

Resolution:

Contact the Altiris Deployment Solution help center to obtain the Hotfix for this issue.

Symptom:

Windows PRAID fails with error code 11 when configuring RAID 5 for ServerRAID-7k

Resolution:

The Windows IPSSSEND utility fails with an error code of -1073741819 with a ServeRAID-7k controller when PRAID attempts to create a RAID 5 on the system. PRAID in default mode (praid /d) will attempt to create a RAID 5 when there are three or more drives present in the system. To use a different raid level, use a custom policy file.

Symptom:

Windows PRAID might fail to restore to default settings for LSI-SAS and LSI-SCSI controllers

Resolution:

The Windows CFGGEN delete command might return a successful return code without deleting the arrays present on the controllers. There is presently no workaround for this problem.

Symptom:

Windows PRAID does not capture the hotspare drives for LSI-SAS and LSI-SCSI controllers

Resolution:

The Windows CFGGEN utility does not capture the hotspare information but it does capture raid levels and all of the drive information.

If you want to capture a configuration and redeploy it to a different server, you must edit the captured policy file and insert the hotspare information into the policy file before redeployment.

Symptom:

PRAID fails with error code 11 when configuring RAID 10 for ServeRAID 8k-l

Resolution:

The Windows ARCCONF utility fails with error code -1073741819 when configuring ServerRAID 8k-l with a RAID level 10. PRAID in default mode (praid /d) will attempt to create a RAID 10 when there are four or more drives present in the system. To use a different raid level, use a custom policy file.

Symptom:

Windows PRAID fails with error code 8 for ServeRAID-7t, 8i, 8k, 8k-l and the message "join failed".

Resolution:

The Windows ARCCONF utility fails during the getconfig command after a setconfig DEFAULT command. This usually happens when the controller contains a hotspare with no RAID arrays present.

Restart the system and try PRAID again, the configuration will be updated so that hotspares are not present.

4 Notices

This information was developed for products and services offered in the U.S.A.

IBM might not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service might be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right might be used instead. However, it is the user responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM might have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement might not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM might make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product, and use of those Web sites is at your own risk.

IBM might use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

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

IBM makes no representations or warranties regarding third-party products or services.

4.1 Edition Notice

© COPYRIGHT INTERNATIONAL BUSINESS MACHINES CORPORATION, 2007.
All rights reserved.

Note to U.S. Government Users Restricted Rights — Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

4.1.1 Trademarks

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

- BladeCenter
- eServer
- IBM
- IBM (logo)
- ServerGuide
- ServerProven
- ServeRAID
- xSeries

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