Hewlett Packard Enterprise

Scripting Tools for Windows PowerShell User Guide

BIOS Cmdlets v3.0.0.0

Abstract

This document contains instructions for using Scripting Tools for Windows PowerShell to manage the BIOS on HPE ProLiant servers. It is intended for system administrators who use the Scripting Tools for Windows PowerShell to manage their IT infrastructure. The BIOS cmdlets support both Legacy and UEFI (Unified Extensible Firmware Interface) boot modes.

Part Number: P00578-004
Published: May 2020
Edition: 1

Contents

Notices

The information contained herein is subject to change without notice. The only warranties for Hewlett Packard Enterprise products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. Hewlett Packard Enterprise shall not be liable for technical or editorial errors or omissions contained herein.

Confidential computer software. Valid license from Hewlett Packard Enterprise required for possession, use, or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Links to third-party websites take you outside the Hewlett Packard Enterprise website. Hewlett Packard Enterprise has no control over and is not responsible for information outside the Hewlett Packard Enterprise website.

Acknowledgments

Microsoft® and Windows® are trademarks of the Microsoft group of companies.

Introduction to Scripting Tools for Windows PowerShell

The Scripting Tools for Windows PowerShell provides a simplified and consistent infrastructure management experience. These sets of PowerShell utilities provide comprehensive Hewlett Packard Enterprise integration tools. These tools are designed for IT experts with experience in PowerShell scripting and configuring HPE ProLiant server hardware.

The Scripting Tools for Windows PowerShell includes sets of PowerShell cmdlets for configuring Hewlett Packard Enterprise ProLiant servers using familiar PowerShell syntax. Documentation describing how to apply these new tools to configure HPE ProLiant servers is also included.

This guide is intended for system administrators who use the Scripting Tools for Windows PowerShell to manage their IT infrastructure. Users must be familiar with Windows PowerShell and the system ROM of ProLiant servers. For more information about configuring the system ROM options, see the guide applicable to your ProLiant server:

For Gen9 and Gen10: **HPE UEFI System Utilities User Guide for your ProLiant server** and other related UEFI documents on the UEFI System Utilities information library (**http://www.hpe.com/info/ProLiantUEFI/docs**).

Windows PowerShell

Windows PowerShell is the Microsoft task automation framework, consisting of a command-line shell and associated scripting language built on a .NET Framework. Businesses face the need to configure large numbers of servers in a quick and reliable fashion. Scripting Tools for Windows PowerShell is a powerful set of utilities that can be used to perform various configuration tasks on Hewlett Packard Enterprise ProLiant servers. These cmdlets follow the standard PowerShell syntax and scripting model, making it easy for you to incorporate these functions into your administrative scripts.

Features

Scripting Tools for Windows PowerShell: BIOS Cmdlets v3.0.0.0 provides the following features:

- Provides interface to HPE BIOS RBSU\UEFI utilities to configure the BIOS settings.
- BIOS settings can be configurable on HPE ProLiant Gen9, Gen10, and Gen10 Plus servers remotely.
- Supports non-ProLiant servers such as StoreEasy, StoreVirtual, and StoreOnce.
- The connection object created using the HPEBIOSCmdlets module can be used across other modules such as HPEiLOCmdlets. This allows a single session to be established on a particular iLO. The same session is used to configure iLO/BIOS settings instead of creating multiple sessions to single iLO. (Version 3.0.0.0 onwards).
- Multithreading helps configure ‘n’ number of servers at once. For better performance, multithreading is used when one cmdlet sends data to multiple targets at the same time.
- You can establish a connection to multiple target servers by providing the range of IP addresses and the related username and password.
- Cloning is possible from one server to multiple servers.
- Provides a better error handling mechanism for BIOS feature supportability and its dependency.

- Provides a logging mechanism to diagnose a problem quickly.
- Provides information about cmdlets, parameters, and parameter value supportability on target servers (`Get-HPEBIOSCmdletInfo`).

Installation

Before installation, ensure that your system meets all requirements for supported operating systems, environments, and hardware. For more information, see the *Scripting Tools for Windows PowerShell Release Notes*.

Operating environment

Install the following before installing Scripting Tools for Windows PowerShell: BIOS Cmdlets. The following links provide access to the Microsoft download sites for these applications. Make sure that you read and understand the system requirements and other information provided.

1. Install Microsoft .NET Framework 4.7.1 or later.

   Microsoft .NET Framework 4.7.1.

   NOTE: Microsoft .NET Framework must be installed **before** installing Windows Management Framework.

2. Windows Management Framework 3.0 or later (which includes PowerShell 3.0 or later):
   - **Windows Management Framework 3.0**
   - **Windows Management Framework 4.0**
   - **Windows Management Framework 5.0**
   - **Windows Management Framework 5.1**

Supported operating systems

The BIOS cmdlets are supported on the following operating system versions:

- Microsoft Windows 7 SP1
- Microsoft Windows 8.1
- Microsoft Windows 10
- Microsoft Windows Server 2012 R2
- Microsoft Windows Server 2016
- Microsoft Windows Server 2019

Installing the BIOS cmdlets

Installing the BIOS cmdlets using MSI

1. Download the Scripting Tools for Windows PowerShell: BIOS Cmdlets installer from the following website: **http://www.hpe.com/servers/powershell**
2. Close all PowerShell windows before the installation.
3. Run the installer from an account with administrative privileges using any standard method of execution (command line or double-click).

It might be necessary to change the execution policy for PowerShell. In PowerShell, enter the following command to get more information and to help you to decide what to select:

```
help about_Execution_Policies
```

Use the following command to see your current execution policy settings:

```
Get-ExecutionPolicy -list
```

You can use the following PowerShell command until you determine if it meets your needs:

```
Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned
```

The installation will halt and not complete successfully if any of the following conditions are detected:

- Attempting to install without .NET 4.5 or above
- Attempting to install without PowerShell 3.0 or above

Installing the BIOS cmdlets from PowerShell Gallery

PowerShell Gallery is a market place where PowerShell module or scripts from vendors, users, and individuals are stored in a cloud environment. It is a central repository for PowerShell content.

You can choose to install the online version of BIOS cmdlets from the Microsoft gallery.

```
Install-Module -Name HPEBIOSCmdlets -Verbose
```

Visit the Microsoft gallery at **https://www.powershellgallery.com** and search for "Scripting Tools for Windows PowerShell: BIOS Cmdlets" for more details.

Uninstalling the BIOS cmdlets

Uninstalling the BIOS cmdlets from Add or Remove programs

1. Open Windows Control Panel
2. Under Programs, click **Uninstall a program**.
3. Select **Scripting Tools for Windows PowerShell: BIOS Cmdlets** from the list of programs.
4. Click **Uninstall**.

Uninstalling the BIOS cmdlets using the Uninstall-Module cmdlet

Use the Uninstall-Module cmdlet to remove the module from your system.

```
Uninstall-Module -Name HPEBIOSCmdlets -Verbose
```

Recommendations

Do not mix the sources of MSI and PowerShell Gallery for installation, upgrade, or uninstallation.

Repairing the BIOS cmdlets

Use the installer repair option for the following scenarios:

- The BIOS cmdlets module is installed, but PowerShell is not able to import the BIOS cmdlets module.
- BIOS cmdlets module files, dependent files, or registry entries are corrupted.

Procedure

1. Open Windows Control Panel.
2. Select **Programs and Features**.
3. Select **Scripting Tools for Windows PowerShell: BIOS Cmdlets**.
4. Click **Repair**.

Using the BIOS cmdlets

The following topics provide information about how to use the BIOS cmdlets.

Descriptions of the BIOS cmdlets

The following table provides a list and brief description of all the BIOS cmdlets.

Cmdlet help

Help is available with the BIOS cmdlets and is supported in the same way as other PowerShell cmdlets. To display a complete list of the BIOS cmdlets in PowerShell, type:

```
help *hpebios*
```

NOTE: You can also use the following command to display the BIOS cmdlets:

```
Get-Command –Module HPEBIOSCmdlets
```

To display complete help for a specific cmdlet, type:

```
help <cmdlet> -Full
```

where `<cmdlet>` is the name of the BIOS cmdlet.

The BIOS cmdlets support the PowerShell `Update-Help` feature. When you execute this command, it retrieves the most current help files from an HPE website and puts them in the correct location on your system.

Table 1: BIOS cmdlets

Cmdlet	Description
Add-HPEBIOSiSCSIBootAttempt	Adds the BIOS iSCSI boot attempt in the iSCSI boot source.
Clear-HPEBIOSLog	Clears any logs created by the BIOS cmdlets module.
Clear-HPEBIOSTLSCertificateInstallationQueue	Clears the BIOS TLS certificate installation queue.
Clear-HPEBIOSTLSCertificateUninstallationQueue	Clears the BIOS TLS certificate uninstallation queue.
Clear-HPEBIOSUserDefault	Clears the user default configuration.
Connect-HPEBIOS	Creates connections to one or multiple BIOS targets.
Disable-HPEBIOSEmbeddedLOMPort	Disables the network boot for the installed network interface card (NIC).
Disable-HPEBIOSLog	Disables BIOS cmdlets logging for the current PowerShell session.
Disable-HPEBIOSNVDIMMErase	Disables the BIOS NVDIMM erase.
Disable-HPEBIOSPCIDeviceOption	Sets the status of the embedded and add-in BIOS PCI devices for the target server.
Disable-HPEBIOSPCIeSlotNetworkBootOption	Disables the UEFI PXE boot status for the installed NIC in PCIe slots.
Disconnect-HPEBIOS	Closes the connection.
Edit-HPEBIOSiSCSIBootAttempt	Edits the existing BIOS iSCSI boot attempt in the iSCSI boot source array.
Enable-HPEBIOSEmbeddedLOMPort	Enables the network boot for the installed NIC.
Enable-HPEBIOSLog	Enables BIOS cmdlets logging for the current PowerShell session.
Enable-HPEBIOSNVDIMMErase	Enables the BIOS NVDIMM erase.
Enable-HPEBIOSPCIDeviceOption	Enables the embedded and add-in BIOS PCI devices for the target server.
Enable-HPEBIOSPCIeSlotNetworkBootOption	Enables the UEFI PXE boot status for the installed NIC in PCIe slots on the target server.
Get-HPEBIOSACPI_SLIT	Gets BIOS ACPI SLIT preferences information.
Get-HPEBIOSAdminInfo	Gets the reference information for the server administrator.

Table Continued

Cmdlet	Description
Get-HPEBIOSAdvancedDebugOption	Gets the BIOS advanced debug option.
Get-HPEBIOSAdvancedMemoryProtection	Gets BIOS advanced memory protection options information.
Get-HPEBIOSAdvancedPCIConfiguration	Gets the BIOS advanced PCI configuration.
Get-HPEBIOSAdvancedPerformanceTuningOption	Gets the BIOS advanced performance tuning options.
Get-HPEBIOSAdvancedSecurityOption	Gets the BIOS advanced security configuration.
Get-HPEBIOSAdvancedSystemROMOption	Gets BIOS advanced system ROM options.
Get-HPEBIOSBootBrowserConfiguration	Gets the BIOS boot browser settings.
Get-HPEBIOSBootMode	Gets the current Boot Mode for the systems that support UEFI.
Get-HPEBIOSBootOrderPolicy	Gets the current Boot Order Policy in UEFI systems.
Get-HPEBIOSBootTimeMemoryOptimization	Gets the BIOS boot time optimization settings.
Get-HPEBIOSCmdletInfo	Gets the list of cmdlets which are supported on the target server.
Get-HPEBIOSCurrentSecureBootState	Gets the BIOS current secure boot state.
Get-HPEBIOSCustomPostMessage	Gets BIOS custom post message.
Get-HPEBIOSDateTimeOption	Obtains the BIOS daylight-saving time, time format, and time zone settings.
Get-HPEBIOSEmbeddedDiagnostics	Obtains the embedded diagnostics settings for the BIOS.
Get-HPEBIOSEmbeddedLOMPort	Obtains the network boot status for the installed NIC.
Get-HPEBIOSEmbeddedUEFIShell	Gets BIOS Embedded UEFI Shell information.
Get-HPEBIOSEmbeddedUserPartition	Gets the Embedded User Partition settings.
Get-HPEBIOSEMSConsole	Gets the EMS console configuration.
Get-HPEBIOSFanOption	Gets the BIOS system fan installation and policy configurations.
Get-HPEBIOSIntelNICDMAChannel	Gets BIOS Intel DMA Channels information.
Get-HPEBIOSIntelTurboBoost	Gets BIOS Intel Turbo Boost information.

Table Continued

Cmdlet	Description
Get-HPEBIOSInternalSDCardSlot	Obtains the BIOS internal Secure Digital (SD) card slot configuration.
Get-HPEBIOSiSCSIBootAttempt	Gets the list of BIOS iSCSI boot attempts in the iSCSI boot source.
Get-HPEBIOSiSCSIInitiatorName	Gets the BIOS iSCSI initiator name.
Get-HPEBIOSiSCSINICSource	Gets the list of the BIOS iSCSI NIC sources.
Get-HPEBIOSLogConfig	Gets the log configuration settings.
Get-HPEBIOSMemoryConfiguration	Gets the BIOS memory configurations.
Get-HPEBIOSMemoryPower	Gets the current settings of memory-related power management.
Get-HPEBIOSMemoryProximityReportingForIO	Gets BIOS Memory Proximity Reporting for I/O information.
Get-HPEBIOSModuleVersion	Gets the module details for the BIOS cmdlets.
Get-HPEBIOSNetworkBootOption	Obtains the BIOS network boot and preboot network options.
Get-HPEBIOSNodeInterleaving	Gets BIOS Node Interleaving information.
Get-HPEBIOSNVDIMMConfiguration	Gets the BIOS NVDIMM configuration.
Get-HPEBIOSNVDIMMErase	Gets the BIOS NVDIMM-N sanitize/erase status.
Get-HPEBIOSPCIDeviceConfiguration	Gets the BIOS PCI device configuration.
Get-HPEBIOSPCIDeviceOption	Gets the status of the embedded and add-in PCI devices.
Get-HPEBIOSPCIePower	Gets the current PCIe related configuration, which may impact system power usage.
Get-HPEBIOSPCIeSlotNetworkBootOption	Obtains the UEFI PXE boot status for NIC installed in PCIe slots.
Get-HPEBIOSPersistentMemoryConfiguration	Gets the persistent memory configuration.
Get-HPEBIOSPowerCapping	Gets the status of Memory Power Capping and Dynamic Power Capping Functionality.
Get-HPEBIOSPowerProfile	Gets the level of power versus performance for the system.
Get-HPEBIOSPowerRegulator	Gets the current HPE Power Regulator.

Table Continued

Cmdlet	Description
Get-HPEBIOSPreferredIOOption	Gets the IO options configurations
Get-HPEBIOSPrefetcher	Gets BIOS HW prefetcher, Adjacent Sector prefetch, DCU prefetcher, and DCUIP prefetcher information.
Get-HPEBIOSProcessorOption	Gets the current settings of processor options.
Get-HPEBIOSProcessorPower	Gets the current settings of processor-related power management.
Get-HPEBIOSQPI	Gets BIOS QPI snoop configuration, QPI bandwidth optimization information.
Get-HPEBIOSQPILinkPower	Gets the current Intel QPI Link Power Management and Frequency.
Get-HPEBIOSRedundantPowerSupplyMode	Gets the current redundant power supply configuration.
Get-HPEBIOSRemovableFlashMediaBootSequence	Obtains the BIOS removable flash media boot sequence.
Get-HPEBIOSSATAControllerOption	Obtains the BIOS SATA controller configuration.
Get-HPEBIOSSecureBootState	Obtains the Secure Boot option settings.
Get-HPEBIOSSerialConsole	Gets the serial console configuration.
Get-HPEBIOSSerialPort	Gets the serial port configuration.
Get-HPEBIOSServerAvailability	Gets BIOS server availability information.
Get-HPEBIOSServerConfigLockSetting	Gets the BIOS Server Configuration Lock settings.
Get-HPEBIOSServerInfo	Gets the reference information for the server administrator.
Get-HPEBIOSServerSecurity	Gets BIOS server security information.
Get-HPEBIOSServiceContact	Gets the reference information for the server service.
Get-HPEBIOSSetting	Gets the base configuration, current settings, and pending settings of BIOS.
Get-HPEBIOSStorageOption	Gets the available storage options of the target server.
Get-HPEBIOSTPMChipInfo	Gets the TPM (Trusted Platform Module) information.
Get-HPEBIOSUPILinkPower	Gets the current Intel UPI link power configuration.

Table Continued

Cmdlet	Description
Get-HPEBIOSSystemInfo	Obtains the BIOS system information.
Get-HPEBIOSThermalOption	Obtains the fan cooling solution for the system.
Get-HPEBIOSTLSCertificate	Gets the list of BIOS installed TLS HTTPS certificates.
Get-HPEBIOSTLSCertificateInstallationQueue	Gets the list of BIOS certificates in the installation queue.
Get-HPEBIOSTLSCertificateUninstallationQueue	Gets the list of BIOS certificates in the uninstallation queue.
Get-HPEBIOSTLSConfiguration	Gets the BIOS TLS configuration.
Get-HPEBIOSTPMConfiguration	Obtains the TPM (Trusted Platform Module) configurations.
Get-HPEBIOSUEFIBootOrder	Gets the current advanced UEFI boot order list.
Get-HPEBIOSUEFIDevicePriority	Obtains the BIOS UEFI device priority.
Get-HPEBIOSUEFIOptimizedBoot	Gets the current UEFI Optimized Boot configuration.
Get-HPEBIOSUSBOption	Obtains BIOS USB options.
Get-HPEBIOSUserDefaultState	Gets the current user default state of the system.
Get-HPEBIOSUtilityLanguage	Gets BIOS system utility language.
Get-HPEBIOSVideoOption	Gets BIOS video options information.
Get-HPEBIOSVirtualization	Gets the hardware virtualization configuration.
Get-HPEBIOSVLANConfiguration	Obtains the details of global VLAN configuration for all enabled network interfaces.
Get-HPEBIOSWorkloadProfile	Gets the BIOS workload profile configuration.
Install-HPEBIOSTLSCertificate	Installs the BIOS TLS certificate into the installation queue.
Remove-HPEBIOSiSCSIBootAttempt	Removes the BIOS iSCSI boot attempt in the iSCSI source.
Reset-HPEBIOSAdminPassword	Resets the BIOS Administrator Password
Reset-HPEBIOSDefaultManufacturingSetting	Resets all BIOS configuration settings to default manufacturing values.
Reset-HPEBIOSPowerOnPassword	Resets the Power On Password.

Table Continued

Cmdlet	Description
Reset-HPEBIOSUserDefault	Resets all BIOS configuration settings to saved default user values.
Set-HPEBIOSACPI_SLIT	Sets BIOS ACPI SLIT preferences.
Set-HPEBIOSAdminInfo	Sets the reference information for the server administrator.
Set-HPEBIOSAdminPassword	Sets the BIOS Administrator Password.
Set-HPEBIOSAdvancedDebugOption	Sets the BIOS advanced debug option.
Set-HPEBIOSAdvancedMemoryProtection	Sets the BIOS advanced memory protection method.
Set-HPEBIOSAdvancedPCIConfiguration	Sets the BIOS advanced PCI configuration.
Set-HPEBIOSAdvancedPerformanceTuningOption	Sets the BIOS advanced performance tuning options.
Set-HPEBIOSAdvancedSecurityOption	Sets the BIOS advanced security configuration.
Set-HPEBIOSAdvancedSystemROMOption	Sets BIOS system ROM options.
Set-HPEBIOSBootBrowserConfiguration	Sets the BIOS boot browser settings.
Set-HPEBIOSBootMode	Sets the Boot Mode for the systems that support UEFI.
Set-HPEBIOSBootOrderPolicy	Sets the Boot Order Policy in UEFI systems.
Set-HPEBIOSBootTimeMemoryOptimization	Sets the boot time optimization options.
Set-HPEBIOSCustomPostMessage	Sets a custom post message.
Set-HPEBIOSDateTimeOption	Sets the BIOS daylight-saving time, time format, and time zone settings.
Set-HPEBIOSEmbeddedDiagnostics	Sets the BIOS Embedded Diagnostics settings.
Set-HPEBIOSEmbeddedUEFIShell	Sets BIOS Embedded UEFI Shell.
Set-HPEBIOSEmbeddedUserPartition	Sets the Embedded User Partition information.
Set-HPEBIOSEMSConsole	Sets the EMS console configuration.
Set-HPEBIOSFanOption	Sets the BIOS system fan installation and policy configurations.
Set-HPEBIOSIntelNICDMAChannel	Sets BIOS Intel DMA Channels.

Table Continued

Cmdlet	Description
Set-HPEBIOSIntelTurboBoost	Sets Intel Turbo Boost.
Set-HPEBIOSInternalSDCardSlot	Sets the BIOS internal SD card slot configuration.
Set-HPEBIOSiSCSIInitiatorName	Sets the BIOS iSCSI initiator name.
Set-HPEBIOSLogConfig	Sets the log configuration settings.
Set-HPEBIOSMemoryConfiguration	Sets the BIOS memory configurations.
Set-HPEBIOSMemoryPower	Sets the memory-related power management.
Set-HPEBIOSMemoryProximityReportingForIO	Sets BIOS Memory Proximity Reporting for I/O information.
Set-HPEBIOSNetworkBootOption	Sets BIOS network boot and preboot network options.
Set-HPEBIOSNodeInterleaving	Sets BIOS Node Interleaving.
Set-HPEBIOSNVDIMMConfiguration	Sets the BIOS NVDIMM configuration.
Set-HPEBIOSPCIDeviceConfiguration	Sets the BIOS PCI device configuration.
Set-HPEBIOSPCIePower	Sets the current PCIe related configuration, which may impact system power usage.
Set-HPEBIOSPersistentMemoryConfiguration	Sets the persistent memory configuration.
Set-HPEBIOSPowerCapping	Sets the status of Memory Power Capping and Dynamic Power Capping Functionality.
Set-HPEBIOSPowerOnPassword	Sets the BIOS Power On Password.
Set-HPEBIOSPowerProfile	Sets the level of power versus performance for the system.
Set-HPEBIOSPowerRegulator	Sets the Power Regulator.
Set-HPEBIOSPreferredIOOption	Sets the IO options.
Set-HPEBIOSPrefetcher	Sets BIOS HW, Adjacent Sector, DCU, and DCUIP prefetcher.
Set-HPEBIOSProcessorOption	Sets the current processor options settings.
Set-HPEBIOSProcessorPower	Sets the processor-related power management.

Table Continued

Cmdlet	Description
Set-HPEBIOSQPI	Sets BIOS QPI snoop configuration and QPI bandwidth optimization.
Set-HPEBIOSQPILinkPower	Sets the Intel QPI Link Power Management and Frequency.
Set-HPEBIOSRedundantPowerSupplyMode	Sets the Redundant Power Supply Mode.
Set-HPEBIOSRemovableFlashMediaBootSequence	Sets the BIOS removable flash media boot sequence.
Set-HPEBIOSSATAControllerOption	Sets the BIOS SATA controller configuration.
Set-HPEBIOSSecureBootState	Sets the Secure Boot option settings.
Set-HPEBIOSSerialConsole	Sets the serial console configuration.
Set-HPEBIOSSerialPort	Sets the serial port configuration.
Set-HPEBIOSServerAvailability	Sets BIOS server availability.
Set-HPEBIOSServerInfo	Sets the reference information for the server administrator.
Set-HPEBIOSServerSecurity	Sets BIOS server security options.
Set-HPEBIOSServiceContact	Sets the reference information for the server service.
Set-HPEBIOSStorageOption	Sets the storage options of the target server.
Set-HPEBIOSUPILinkPower	Sets the Intel UPI link power configuration.
Set-HPEBIOSSystemInfo	Sets the BIOS system information.
Set-HPEBIOSThermalOption	Sets the fan cooling solution for the system.
Set-HPEBIOSTLSConfiguration	Sets the BIOS TLS configuration.
Set-HPEBIOSTPMConfiguration	Sets the TPM (Trusted Platform Module) configurations.
Set-HPEBIOSUEFIBootOrder	Sets the UEFI Boot Order configuration.
Set-HPEBIOSUEFIOptimizedBoot	Sets the UEFI Optimized Boot state.
Set-HPEBIOSUSBOption	Sets the BIOS USB options.
Set-HPEBIOSUserDefault	Sets the BIOS user default configuration.

Table Continued

Cmdlet	Description
Set-HPEBIOSUtilityLanguage	Selects the language for the system.
Set-HPEBIOSVideoOption	Sets BIOS video options.
Set-HPEBIOSVirtualization	Sets the hardware virtualization configuration.
Set-HPEBIOSVLANConfiguration	Sets the details of global VLAN configuration for all enabled network interfaces.
Set-HPEBIOSWorkloadProfile	Sets the BIOS workload profile configuration.
Test-HPEBIOSConnection	Checks if the connection to the target is still valid.
Uninstall-HPEBIOSTLSCertificate	Uninstalls the installed certificate.

Establishing a BIOS connection

Scripting Tools for Windows PowerShell: BIOS Cmdlets supports the following generations of HPE ProLiant servers. Each generation of servers has a different connection mechanism.

- Gen9 servers
- Gen10 servers
- Gen10 Plus servers
- Non-ProLiant servers such as StoreEasy, StoreVirtual, and StoreOnce

NOTE:

- You can establish a BIOS connection to ProLiant servers and non-ProLiant (StoreEasy, StoreVirtual, and StoreOnce) servers using the same procedure. See **Establishing a connection to a Gen9 server** for Gen9-based non-ProLiant servers. See **Establishing a connection to a Gen10 or Gen10 Plus server** for Gen10/Gen10 Plus-based non-ProLiant servers.
- Use the iLO IP address to establish a connection on non-ProLiant servers (StoreEasy, StoreVirtual, and StoreOnce). The appliance address is not valid.

Establishing a connection to a Gen9 server

Use this mode of connection for Gen9 servers. It uses the iLO IP address and iLO user credentials.

Prerequisites

- Install BIOS cmdlets on the Windows management client.
- Make sure the target server iLO IP address is able to ping from the management client where the BIOS cmdlets are installed.

- If the administrator password has been set for BIOS, the administrator password must be provided through a connection using the `AdminPassword` parameter. The set cmdlet will not execute if the password has been set but not provided during connection.
- The target iLO must have a valid server certificate. If a valid certificate is not available, use the `DisableCertificateAuthentication` switch parameter to establish the connection.

Procedure

1. Execute Connect-HPEBIOS with the iLO IP address of the Gen9 target server.

   A successful connection will return the session object.

2. Use the session object from the previous step to run BIOS cmdlets.

Example: Connecting to a Gen9 server that does not have a valid server certificate

```
PS C:\> $connection = Connect-HPEBIOS -IP 10.20.30.40 -Username admin -
Password admin123 -DisableCertificateAuthentication

PS C:\> $connection
IP                                : 10.20.30.40
Hostname                          :
Timeout                           : 30
DisableCertificate Authentication : True
TargetInfo                        : @{ProductName=ProLiant ML350 Gen9;ServerFamily=ProLiant;
                                    iLOFirmwareVersion=2.62;SystemROM=P92 v2.72 (03/25/2019);
                                    ProcessorName=Intel(R) Xeon(R)CPU E5-2603 v3 @ 1.60GHz}
ExtendedInfo                      : @{UserSuppliedAddress=10.20.30.40;
                                    HttpConnectAddress=10.20.30.40;UserName=admin;
                                    Modifier1=vyUuq0Oh/XOc1n+Vv4w1YcFF+FvdoVk/FUEH2cdAD/4=;
                                    Modifier2=4eA+yvA4kyRK8WYwIeT7zA==}
ConnectionInfo                    : @{RIBCL=; REST=}

PS C:\> $connection.TargetInfo
ProductName                       : ProLiant ML350 Gen9
ServerFamily                      : ProLiant
ServerGeneration                  : Gen9
ServerModel                       : ML350
iLOGeneration                     : iLO4
iLOFirmwareVersion                : 2.62
SystemROM                         : P92 v2.72 (03/25/2019)
ProcessorName                     : Intel(R) Xeon(R) CPU E5-2603 v3 @ 1.60GHz

PS C:\> $connection.ExtendedInfo

UserSuppliedAddress               : 10.20.30.40
HttpConnectAddress                : 10.20.30.40
UserName                          : admin
Modifier1                         : vyUuq0Oh/XOc1n+Vv4w1YcFF+FvdoVk/FUEH2cdAD/4=
Modifier2                         : 4eA+yvA4kyRK8WYwIeT7zA==

PS C:\> $connection.ConnectionInfo | fl

RIBCL                             : @{AuthToken=ciNRS6xqkr9VO6JV7f1LQQ==; iLOResetWaitTimeout=0}
REST                              : @{RootUrl=https://10.20.30.40/rest/v1;
                                    XAuthToken=iOIck9yMeVbdoNJaD4Zh1xUHMa1CsiQpP1
                                    MHZls5BpfkG1yMFYgjEAEdZFfUm0Ng;
                                    Location=https://10.20.30.40/rest/v1/
                                    SessionService/Sessions/admin5e385e3ea6a7ef9e;
                                    BIOSAdminPassword=; BIOSAttributeRegistryName
                                    =HpBiosAttributeRegistryP92.1.1.72}
```

Establishing a connection to a Gen10 or Gen10 Plus server

Use this method to connect with Gen10 servers. It uses the iLO IP and iLO user credentials to connect.

Prerequisites

- Install BIOS cmdlets on the Windows management client.
- Make sure the target server iLO IP address is able to ping from the management client where the BIOS cmdlets are installed.
- The target iLO must have a valid server certificate. If a valid certificate is not available, use the `DisableCertificateAuthentication` switch parameter to establish the connection.
- The iLO user must have the "Host BIOS" privilege to change the BIOS features using the BIOS cmdlets.

Procedure

1. Execute Connect-HPEBIOS with the iLO IP address of the Gen10 target server.

   A successful connection will return the session object.

2. Use the session object from the previous step to run BIOS cmdlets.

Example: Connecting to a Gen10 server that has a valid certificate

```
PS C:\> $connection = Connect-HPEBIOS -IP 10.20.30.40-Username admin
                -Password admin123

PS C:\> $connection

IP : 10.20.30.40
Hostname                          : xyz.abcd.com
Timeout                           : 30
DisableCertificateAuthentication : True
TargetInfo                        : @{ProductName=ProLiant DL580 Gen10;ServerFamily=ProLiant;
                                    ServerGeneration=Gen10;ServerModel=DL580;
                                    iLOGeneration=iLO5;iLOFirmwareVersion=2.1;
                                    SystemROM=U34 v2.16 (09/12/2019);
                                    ProcessorName=Intel(R) Genuine processor}
ExtendedInfo                      : @{UserSuppliedAddress=10.20.30.40;
                                    HttpConnectAddress=10.20.30.40;UserName=admin;
                                    Modifier1=b3JJfwxwbZSKKyjpgKmn3QPj16vDZ7Q3oq9ejPZCOPk=;
                                    Modifier2=cqEhobrR75LcegppbRmPWQ==}
ConnectionInfo                    : @{Redfish=}

PS C:\> $connection.TargetInfo

ProductName                       : ProLiant DL580 Gen10
ServerFamily                      : ProLiant
ServerGeneration                  : Gen10
ServerModel                       : DL580
iLOGeneration                     : iLO5
iLOFirmwareVersion                : 2.1
SystemROM                         : U34 v2.16 (09/12/2019)
ProcessorName                     : Intel(R) Genuine processor

PS C:\> $connection.ExtendedInfo

UserSuppliedAddress               : 10.20.30.40
HttpConnectAddress                : xyz.abcd.com
UserName                          : admin
Modifier1                         : b3JJfwxwbZSKKyjpgKmn3QPj16vDZ7Q3oq9ejPZCOPk=
Modifier2                         : cqEhobrR75LcegppbRmPWQ==
```

```
PS C:\> $connection.ConnectionInfoRedfish


-------

@{ResourceDirectoryJSON={"@odata.context":"/redfish/v1/
$metadata#HpeiLOResourceDirectory.HpeiLOResourceDirectory",
"@odata.etag":"W/\"9B6CB9C6\"", "@odata.id":"/...
HttpConnectAddress : xyz.abcd.com
```

IPv6 support

Consider the following when using IPv6.

- IPv6 is supported and enabled on the system.
- IPv6 is supported along with IPv4 for network addresses on all cmdlets that have an IP address parameter. The double colon zero subnet format for IPv6 addresses is supported.

  For example:

  `1a00::1fe8`

  equates to:

  `1a00:0000:0000:0000:0000:0000:0000:1fe8`
- Address ranges are supported with the dash character.

  For example:

  `1a00::1fe8-1fef` resolves to eight addresses,

  from:

  `1a00::1fe8`

  through:

  `1a00::1fef`
- Sets of addresses are supported with the comma character.

  For example:

  `1a00,1b00::1fe8`

  resolves to two addresses:

  `1a00::1fe8`

  and:
  `1b00::1fe8`

Example: Connect-HPEBIOS using IPv6 to Gen9 target server

```
PS C:\> $connection = Connect-HPEBIOS -IP FE80::9618:82FF:FE03:3A70
        -Usernameadmin -Password aduser1234 -DisableCertificateAuthentication

PS C:\> $connection

IP                                : fe80:0:0:0:9618:82ff:fe03:3a70
Hostname                          :
Timeout                           : 30
DisableCertificateAuthentication : TrueTargetInfo : @{ProductName=ProLiant ML350 Gen9;
                                    ServerFamily=ProLiant;ServerGeneration=Gen9;
```

```
                                  ServerModel=ML350; iLOGeneration=iLO4;
                                  iLOFirmwareVersion=2.62;SystemROM=P92v2.72 (03/25/2019);
                                  ProcessorName=Intel(R) Xeon(R)CPU E5-2603 v3 @ 1.60GHz}
ExtendedInfo                    : @{UserSuppliedAddress=fe80:0:0:0:9618:82ff:fe03:3a70;
                                  HttpConnectAddress=fe80:0:0:0:9618:82ff:fe03:3a70;UserName=admin;
                                  Modifier1=vyUuq0Oh/XOc1n+Vv4w1YcFF+FvdoVk/FUEH2cdAD/4=;
                                  Modifier2=4eA+yvA4kyRK8WYwIeT7zA==}
ConnectionInfo                  : @{RIBCL=; REST=}

PS C:\> $connection.TargetInfo

ProductName                     : ProLiant ML350 Gen9
ServerFamily                    : ProLiant
ServerGeneration                : Gen9
ServerModel                     : ML350
iLOGeneration                   : iLO4
iLOFirmwareVersion              : 2.62
SystemROM                       : P92 v2.72 (03/25/2019)
ProcessorName                   : Intel(R) Xeon(R) CPU E5-2603 v3 @ 1.60GHz

PS C:\> $connection.ExtendedInfo

UserSuppliedAddress             : fe80:0:0:0:9618:82ff:fe03:3a70
HttpConnectAddress              : fe80:0:0:0:9618:82ff:fe03:3a70
UserName                        : admin
Modifier1                       : vyUuq0Oh/XOc1n+Vv4w1YcFF+FvdoVk/FUEH2cdAD/4=
Modifier2                       : 4eA+yvA4kyRK8WYwIeT7zA==

PS C:\> $connection.ConnectionInfo | fl

RIBCL                           : @{AuthToken=ciNRS6xqkr9VO6JV7f1LQQ==; iLOResetWaitTimeout=0}
REST                            : @{RootUrl=https://fe80:0:0:0:9618:82ff:fe03:3a70/rest/v1;
                                  XAuthToken=iOIck9yMeVbdoNJaD4Zh1xUHMa1CsiQpP1
                                  MHZls5BpfkG1yMFYgjEAEdZFfUm0Ng;
                                  Location=https://fe80:0:0:0:9618:82ff:fe03:3a70/
                                  rest/v1/SessionService/Sessions/
                                  admin5e385e3ea6a7ef9e; BIOSAdminPassword=;
                                  BIOSAttributeRegistryName=HpBiosAttributeRegistryP92.1.1.72}
```

NOTE: Other examples in this document use IPv4, but could use IPv6 instead, if supported on the network. Both IPv4 or IPv6 addresses can be used to establish the connection.

For more information on IPv6, see the following website or the references it links to **http://en.wikipedia.org/wiki/IPv6**.

XAuthToken support

Example: Connect-HPEBIOS using the XAuthToken parameter (SSO token)

```
PS C:\> $connection = Connect-HPEBIOS -IP 192.168.10.34
                -XAuthToken f8febab0d8c218372b9aa74e97d17af7 -DisableCertificateAuthentication

PS C:\> $connection

IP                              : 10.20.30.40:100
Hostname                        : dl360gen10-01.abcd.com:100
Timeout                         : 30
DisableCertificateAuthentication : True
TargetInfo                      : @{ProductName=ProLiant DL580 Gen10;ServerFamily=ProLiant;
                                  ServerGeneration=Gen10;ServerModel=DL580;
                                  iLOGeneration=iLO5;iLOFirmwareVersion=2.1;
                                  SystemROM=U34 v2.16 09/12/2019);
                                  ProcessorName=Intel(R)Genuine processor}
ExtendedInfo                    : @{UserSuppliedAddress=dl360gen10-01.abcd.com:100;
                                  HttpConnectAddress=dl360gen10-01.abcd.com:100;
                                  UserName=admin;
                                  Modifier1=b3JJfwxwbZSKKyjpgKmn3QPj16vDZ7Q3oq9ejPZCOPk=;
                                  Modifier2=cqEhobrR75LcegppbRmPWQ==}
ConnectionInfo                  : @{Redfish=}
```

```
PS C:\> $connection.TargetInfo

ProductName                       : ProLiant DL580 Gen10
ServerFamily                      : ProLiant
ServerGeneration                  : Gen10
ServerModel                       : DL580
iLOGeneration                     : iLO5
iLOFirmwareVersion                : 2.1
SystemROM                         : U34 v2.16 (09/12/2019)
ProcessorName                     : Intel(R) Genuine processor

PS C:\> $connection.ExtendedInfoUserSuppliedAddress : dl360gen10-01.abcd.com:100
        HttpConnectAddress : dl360gen10-01.abcd.com:100

UserName                          : admin
Modifier1                         : b3JJfwxwbZSKKyjpgKmn3QPj16vDZ7Q3oq9ejPZCOPk=
Modifier2                         : cqEhobrR75LcegppbRmPWQ==

PS C:\> $connection.ConnectionInfoRedfish

@{ResourceDirectoryJSON={"@odata.context":"/redfish/v1/
$metadata#HpeiLOResourceDirectory.HpeiLOResourceDirectory",
"@odata.etag":"W/\"9B6CB9C6\"","@odata.id":"/...
```

Example: Connecting to iLO 5 servers using XAuthToken (through HPE OneView)

XAuthToken can be generated from Redfish API for the given iLO IP or through HPE OneView using OV cmdlets as follows:

```
PS C:\>Connect-HPOVMgmt -Hostname <IP> -UserName <username>
       -Password <password>
PS C:\>$result = Get-HPOVServer
PS C:\>$remoteConsoleUrl = "$($result[1].uri)/remoteConsoleUrl"
PS C:\>$resp = Send-HPOVRequest $remoteConsoleUrl
PS C:\>$URL,$session = $resp.remoteConsoleUrl.Split("&")
PS C:\>$http, $iLOIP = $URL.split("=")
PS C:\>$sName,$xAuthToken = $session.split("=")
```

Using the `Get-HPOVIloSso` cmdlet available from the HPOneView version 4.x module:

```
$connection = Connect-HPEiLO -IP $iLOIP -XAuthToken $xAuthToken

PS C:\> $connection

IP                                : 10.20.30.40
Hostname                          : dl360gen10-01.abcd.com
Timeout                           : 30
DisableCertificateAuthentication  : True
TargetInfo                        : @{ProductName=ProLiantDL580 Gen10;ServerFamily=ProLiant;
                                    ServerGeneration=Gen10;ServerModel=DL580;
                                    iLOGeneration=iLO5;iLOFirmwareVersion=2.1;
                                    SystemROM=U34 v2.16 (09/12/2019);ProcessorName=Intel(R)
                                    Genuine processor}
ExtendedInfo                      : @{UserSuppliedAddress=10.20.30.40;
                                    HttpConnectAddress=10.20.30.40;UserName=admin;
                                    Modifier1=b3JJfwxwbZSKKyjpgKmn3QPj16vDZ7Q3oq9ejPZCOPk=;
                                    Modifier2=cqEhobrR75LcegppbRmPWQ==}
ConnectionInfo                    : @{Redfish=}
```

Connecting to multiple target servers

For better performance, multithreading is used when one cmdlet sends data to multiple targets. Most cmdlets that support multiple targets use the multithreading feature in the cmdlets. The higher value between the following is considered for maximum threads launching:

Up to 64 threads or the total number of cores present on the client machine where the cmdlets are executed. This number is chosen after measuring response times and observing greatly diminishing returns by using more.

Performance of the cmdlets depends on factors such as:

- Current system load
- Available memory
- Number of processors
- Network configuration
- Other systems in the network
- Other network traffic

To take advantage of multithreading, a single cmdlet is used. However, it is directed to multiple targets in a single invocation by passing parameter values as an array.

Establishing a connection to multiple target servers

A connection can be established to the multiple target servers by providing the range of IP addresses and the related username and password. Connect-HPEBIOS returns a list of session objects which can be used to execute any BIOS cmdlets.

Example

In this example, the count of "3" shows that three session objects, each of which corresponds to a server IP address, are returned by the Connect-HPEBIOS cmdlet.

```
PS C:\> $connection = Connect-HPEBIOS -IP 10.20.30.1,10.20.30.2,10.20.30.3
        -Username admin -Password admin123
PS C:\> $connection.Count
3
```

Executing a cmdlet using multiple connections

Example 1

The following example executes Get-HPEBIOSBootMode to multiple servers.

```
PS C:\> Get-HPEBIOSBootMode -Connection $connection

IP              :    10.20.30.1
Hostname        :    abc1.domain.com
Status          :    OK
BootMode        :    UEFIMode

IP              :    10.20.30.3
Hostname        :    abc2.domain.com
Status          :    OK
BootMode        :    LegacyBIOSMode
```

```
IP              :    10.20.30.5
Hostname        :    abc3.domain.com
Status          :    OK
BootMode        :    UEFIMode
```

Example 2

The following example executes Set-HPEBIOSBootMode on multiple servers. The boot mode will be changed correspondingly on each server.

```
PS C:\> $result = Set-HPEBIOSBootMode -Connection $connection –BootMode
                  @(“LegacyBIOSMode”,” UEFIMode”,” LegacyBIOSMode”)
```

Executing a cmdlet with multiple connections, passing the connection through a pipeline input and other parameters using a named parameter

When there are multiple connections being passed as pipeline input, use the comma operator (`,`) to pass the list of connections as an array. If the comma operator is not used, then the parameter value at index 0 will be set for all the target servers.

Example 1

In this example, `$connections` has three connections that are passed as pipeline input to the Set-HPEBIOSBootMode cmdlet using the comma operator (`,`). BootMode will be changed in each of the corresponding target servers.

```
PS C:\> ,$connections | Set-HPEBIOSBootMode
        –BootMode @(“ LegacyBIOSMode”,” UEFIMode”,” LegacyBIOSMode”)
```

Example 2

In this example, `$connections` has three connections that are passed as pipeline input to the Set-HPEBIOSBootMode cmdlet without the comma operator (`,`). BootMode at index 0 (LegacyBIOSMode) will be set for all the target servers.

```
PS C:\> $connections | Set-HPEBIOSBootMode –BootMode
        @(“ LegacyBIOSMode”,” UEFIMode”,” LegacyBIOSMode”)

        WARNING: Some values of the command-line parameter
        'BootMode' have been ignored for the pipeline
        object(s) at index: [0],[1],[2].
```

NOTE: You must reboot the server to apply the configuration changes to BIOS.

Piping output from one command to another

A useful feature of PowerShell is the ability to pipe output from one command to another. The following example shows piping output from Connect-HPEBIOS to Get-HPEBIOSPowerProfile to produce the power profile information of those connected servers. The `–Verbose` parameter is used to view more information.

PowerShell script:

```
PS C:\> $PowerProfile= Connect-HPEBIOS -IP 192.168.243.100-102 -Username "username"
 -Password "password" -DisableCertitificateAuthentication | Get-HPEBIOSPowerProfile
```

Script output:

```
PS C:\> $PowerProfile

IP              : 192.168.243.100
Hostname        : server1.Company.net
Status          : OK
PowerProfile    : MaximumPerformance


IP              : 192.168.243.101
Hostname        : server2.Company.net
Status          : OK
PowerProfile    : BalancedPowerAndPerformance

IP              : 192.168.243.102
Hostname        : server3.Company.net
Status          : OK
PowerProfile    : Custom
```

The verbose output indicates that three threads are being used for Get-HPEBIOSPowerProfile. This threading enables multiple commands to multiple servers to be sent at the same time. Connect-HPEBIOS makes the connection object array of the three servers. Those are in turn passed through to Get-HPEBIOSPowerProfile, which uses those connections and requests the power profile information from each server. The final results are the power profile information for the three servers connected.

Using the Get-HPEBIOSModuleVersion cmdlet

The Get-HPEBIOSModuleVersion cmdlet is used to determine the current version of the BIOS cmdlets module installed.

The Get-HPEBIOSModuleVersion cmdlet has no parameters. It accesses the installed module file and help files and displays information about them including version numbers. The following is typical Get-HPEBIOSModuleVersion cmdlet output.

```
PS C:\> Get-HPEBIOSModuleVersion

Name                              : HPEBIOSCmdlets
Path                              : C:\Program Files\Hewlett Packard Enterprise\PowerShell
                                    \Modules\HPEBIOSCmdlets\HPEBIOSCmdlets.dll
Description                       : Scripting Tools for Windows PowerShell : BIOS Cmdlets creates
                                    an interface to HPE BIOS ROM-Based Setup
                                    Utilty (RBSU) or UEFI System Utilities.
                                    These cmdlets can be used to configure the BIOS
                                    settings on HPE ProLiant servers.
GUID                              : b37fea1c-1be5-42a6-bbc9-a453018b2a0f
CurrentUICultureName              : en-US
CurrentUICultureVersion           :
CurrentModuleVersion              : 3.0.0.0
LatestAvailableModuleVersion      : 3.0.0.0
ProductModuleDownloadURL          : https://www.hpe.com/servers/powershell
PowerShellGalleryModuleDownloadURL : www.powershellgallery.com/packages/HPEBIOSCmdlets/3.0.0.0
DotNetVersion                     : 4.7
DotNetFrameworkDescription        : 4.7.02053
PSVersion                         : @{PSVersion=5.0.10586.117;PSCompatibleVersions=System.
                                    Version[];BuildVersion=10.0.10586.117;
                                    CLRVersion=4.0.30319.42000;WSManStackVersion=3.0;
                                    PSRemotingProtocolVersion=2.3;
                                    SerializationVersion=1.1.0.1}
OSVersion                         : @{Caption=Microsoft Windows 8.1 Enterprise;
                                    CSDVersion=; Version=6.3.9600; BuildNumber=9600}
```

```
CCGVersion                          : 2.0.0.0
AvailableUICulture                  : {}
Status                              : OK
StatusInfo                          :
```

Managing BIOS using BIOS cmdlets

Examples in this section assume that a connection has been made with Connect-HPEBIOS and is in the `$connection` variable. The `$connection` variable contains a single connection object.

After a connection to the target server is established, you can use cmdlets to manage the BIOS. This section contains examples of how these cmdlets are used.

- **Get and Set BIOS cmdlets**
- **Enable and Disable BIOS cmdlets**
- **Set cmdlets for features that have dependencies**
- **Set parameter that is not supported on target server**
- **Set parameter value that is not supported on the target server**
- **Executing a cmdlet that is not supported on the target server**
- **Executing Get BIOS cmdlets OutputType as raw text**

Get and Set BIOS cmdlets

Example: retrieve thermal-related configuration

The `Get- HPEBIOSThermalOption` cmdlet can be used to retrieve thermal-related configuration as shown in the following example.

```
PS C:\>Get-HPEBIOSThermalOption -Connection $connection

IP                                  : 10.20.30.15
Hostname                            : abc.domain.com
Status                              : OK
ThermalConfiguration                : MaximumCooling
ThermalShutdown                     : Enabled
ExtendedAmbientTemperatureSupport   : ASHRAE3
```

Example: change the BIOS thermal-related configuration

Other cmdlets can be used to set or update BIOS settings. For example, the `Set-HPEBIOSThermalOption` cmdlet can be used to change the BIOS thermal-related configuration as shown in the following example.

```
PS C:\> $setResult = $connection | Set-HPEBIOSThermalOption
        -ThermalConfiguration IncreasedCooling -ThermalShutdown
        Disabled
```

If the set cmdlet is successful, it returns `null`. The following example shows a check of a successful cmdlet execution.

```
PS C:\> $setResult -eq $null
True
```

Enable and Disable BIOS cmdlets

The Enable and Disable cmdlets in BIOS have a special implementation in which parameter values represent the feature, and the parameters themselves are the feature values from the RBSU context. This reduces the number of cmdlets and avoids redundant types of cmdlets.

Example: get status of the LOM port

In the following example, the `Get-HPEEmbeddedLOMPort` cmdlet gets the status of the LOM port which is enabled or disabled. The `Enable-HPEBIOSEmbeddedLOMPort` cmdlet is used to enable the LOM port and `Disable-HPEBIOSEmbeddedLOMPort` cmdlet is used to disable the LOM port.

Get the current EmbeddedLOMPort configuration

```
PS C:\> $returnObj = Get-HPEBIOSEmbeddedLOMPort -Connection $connection
PS C:\> $returnObj
        IP                  : 10.20.30.20
        Hostname            : abc.domain.com
        Status              : Warning
        StatusInfo          : @{Category=PropertySupportability;
                              Message=The property or properties listed
                              in AffectedAttribute are not supported on
                              the target server ProLiant DL380 Gen10. For
                              more details about supported properties,
                              see the cmdlet help or the Troubleshooting
                              chapter of BIOS Cmdlets user guide.;
                              AffectedAttribute=System.Collections.Generic.List
                              `1[System.String]}
        EnableNetworkBoot   : {NICBoot1, NICBoot2, NICBoot3, NICBoot4}
        DisableNetworkBoot  : {}

PS C:\> $returnObj.EnableNetworkBoot
NICBoot1
NICBoot2
NICBoot3
NICBoot4
```

Disable embedded LOM Port

```
PS C:\> Disable-HPEBIOSEmbeddedLOMPort -Connection $connection
        -DisableNetworkBoot
        @(,@("NICBoot1","NICBoot2","NICBoot3","NICBoot4"))
```

Enable embedded LOM Port

```
PS C:\> Enable-HPEBIOSEmbeddedLOMPort -Connection $connection
        -EnableNetworkBoot
        @(,@("NICBoot1","NICBoot2","NICBoot3","NICBoot4"))
```

Set cmdlets for features that have dependencies

While executing some of the Set cmdlets for features that have dependencies, the values of dependent features might be modified or an error will be returned.

Example 1

This example illustrates a Power profile on a Gen9 server that has dependent features whose values will change as listed under `AffectedAttribute`.

```
PS C:\> $result = $connection | Set-HPEBIOSPowerProfile -PowerProfile
        BalancedPowerAndPerformance
PS C:\> $result
        IP                  : 10.20.30.15
        Hostname            : abc.domain.com
        Status              : Warning
        StatusInfo          : @{Category=Dependency; Message=The feature(s)
                              listed in AffectedAttribute might get
                              modified to specified values due to
                              dependency on PowerProfile.;
                              AffectedAttribute=System.Collections.Generic.Dictionary
                              `2[System.String,System.Object]}

PS C:\> $result.StatusInfo
        Category            : Dependency
        Message             : The feature(s) listed in AffectedAttribute might get
                              modified to specified values due to dependency on
                              PowerProfile.
        AffectedAttribute   : : {[PowerRegulator, DynamicPowerSavingsMode],
                              [IntelQPILinkPowerManagement, Enabled],
                              [IntelQPILinkFrequency, Auto],
                              [MinimumProcessorIdlePowerCoreState, C6State]...}


PS C:\> $result.StatusInfo.AffectedAttribute
Key                                                 Value
---                                                 -----
PowerRegulator                                      DynamicPowerSavingsMode
IntelQPILinkPowerManagement                         Enabled
IntelQPILinkFrequency                               Auto
MinimumProcessorIdlePowerCoreState                  C6State
MinimumProcessorIdlePowerPackageState               PackageC6State
EnergyPerformanceBias                               BalancedPerformance
MaximumMemoryBusFrequency                           Auto
ChannelInterleaving                                 Enabled
PCIExpressSupport                                   MaximumSupported
```

Example 2

The following example illustrates what happens when a user attempts to set UEFIOptimizedBoot when the boot mode is "LegacyBIOSMode". This generates an error because UEFIOptimizedBoot is supported only when boot mode is "UEFIMode".

```
PS C:\> $connection | Get-HPEBIOSBootMode
        IP          : 10.20.30.15
        Hostname    : abc.domain.com
        Status      : OK
        BootMode    : LegacyBIOSMode

PS C:\> $result = $connection | Set-HPEBIOSUEFIOptimizedBoot
        -UEFIOptimizedBoot Enabled

PS C:\> $result
        IP          : 10.20.30.15
```

```
        Hostname   : abc.domain.com
        Status     : Error
        StatusInfo : @{Category=Dependency; Message=Cannot modify
                     the listed parameter in AffectedAttribute
                     when BootMode is LegacyBIOSMode.;
                     AffectedAttribute=System.Collections
                     .Generic.List`1[System.String]}

PS C:\Windows\s> $result.StatusInfo
         Category          : Dependency
         Message           : Cannot modify the listed parameter
                             in AffectedAttribute when BootMode
                             is LegacyBIOSMode.
         AffectedAttribute : {UEFIOptimizedBoot}
```

Set parameter that is not supported on target server

If parameters are not supported, then a cmdlet will return PSObject with "StatusInfo" which contains an error message and other related details.

In this example, the parameter "HPOptionROMPrompting" is not supported on a Gen10 target server. To set "HPOptionROMPrompting" on Gen10 servers, the returned value will have a "StatusInfo".

```
PS C:\> $result = $connection | Set-HPEBIOSAdvancedSystemROMOption
        -HPOptionROMPrompting Enabled

PS C:\> $result
        IP                : 10.20.30.15
        Hostname          : abc.domain.com
        Status            : Error
        StatusInfo        : @{Category=ParameterSupportability;
                            Message=The parameter(s) listed in
                            AffectedAttribute are not supported on the
                            target server ProLiant DL360 Gen10. For
                            more details about supported parameters,
                            see the cmdlet help or the Troubleshooting
                            chapter of BIOS Cmdlets user guide.;
                            AffectedAttribute=System.Collections
                            .Generic.List`1[System.String]}

PS C:\> $result.StatusInfo
        Category          : ParameterSupportability
        Message           : The parameter(s) listed in AffectedAttribute
                            are not supported on the target server ProLiant
                            DL360 Gen10. For more details about supported
                            parameters, see the cmdlet help or the
                            Troubleshooting chapter of BIOS Cmdlets user
                            guide.
        AffectedAttribute    :    {HPOptionROMPrompting}
```

Set parameter value that is not supported on the target server

If a parameter value is not supported, the cmdlet will return PSObject with "StatusInfo" which contains an error message and other related details.

EMSConsole values COM1 and COM2 are not supported on Gen10. If a user attempts to set the EMSConsole value to COM1 on a Gen10 target server, returned value will have a "StatusInfo".

```
PS C:\> $result = $connection | Set-HPEBIOSEMSConsole
        -EMSConsole COM1

PS C:\> $result.StatusInfo
```

```
Category          : ParameterValueSupportability
Message           : The parameter value(s) listed in
                    AffectedAttribute are not supported
                    on the target server ProLiant DL360
                    Gen10. For more details about
                    supported parameter values, see
                    the cmdlet help or the Troubleshooting
                    chapter of BIOS Cmdlets user guide.
AffectedAttribute : {[EMSConsole, COM1]}
```

Executing a cmdlet that is not supported on the target server

If a user attempts to execute a cmdlet that is not supported on the target server, PSObject with "StatusInfo" will be returned.

`Get-HPEBIOSAdvancedDebugOption` is only supported on Gen10 servers. When executed on different target servers, it will return PSObject with "StatusInfo".

```
Get-HPEBIOSAdvancedDebugOption is only supported on Gen 10 servers. When executed on different target
servers, it will return PSObject with "StatusInfo".

PS C:\> $result = $connection |Get-HPEBIOSAdvancedDebugOption

PS C:\> $result

IP         : 10.20.30.15
Hostname   : abc.domain.com
Status     : Error
StatusInfo : @{Category=CmdletSupportability; Message=This cmdlet is not supported on the target
             server ProLiant DL380 Gen9. For more details about using this cmdlet, see the help.}

PS C:\ > $result.StatusInfo

Category   : CmdletSupportability
Message    : This cmdlet is not supported on the target server
             ProLiant DL380 Gen9. For more details about using
             this cmdlet, see the help.
```

Executing Get BIOS cmdlets OutputType as raw text

You can select the RawText output type to display detailed information returned from the BIOS cmdlets. If the target server is Gen9 or Gen10, the RawText format will be JSON. The following examples use the Get-HPEBIOSThermalOption cmdlet.

RawText output from Gen9 server

```
PS C:\> $connection | Get-HPEBIOSThermalOption -OutputType RawText

{
  "ThermalConfiguration": "IncreasedCooling",
  "ThermalShutdown": "Enabled",
  "ExtendedAmbientTemperatureSupport": "Disabled"
}
```

NOTE: If an error or warning occurs, the Get cmdlet will return PSObject regardless of output type.

Using the Disconnect-HPEBIOS cmdlet

Use the Disconnect-HPEBIOS cmdlet to disconnect the connection object when you are finished using the BIOS settings. This is necessary when the connection is for WinPE. If there is a connection to WinPE that has not disconnected, another connection cannot be made to the WinPE because only one VSP session is allowed at a time.

For a connection to iLO G9 or G10 servers: if you do not use Disconnect-HPEBIOS, there will be an open session that will not be disconnected until a session inactivity timer expires. Connection expire time is based on iLO session time-out.

```
PS C:\> Disconnect-HPEBIOS -Connection $connection

PS C:\>
```

If the cmdlet is successful, no other message is displayed. If an error occurs, an appropriate output message is displayed.

Typical BIOS cmdlets with examples

The following sections contain examples of ways to use BIOS cmdlets.

Examples in this section assume that a connection has been made with Connect-HPEBIOS and is in `$connection`.

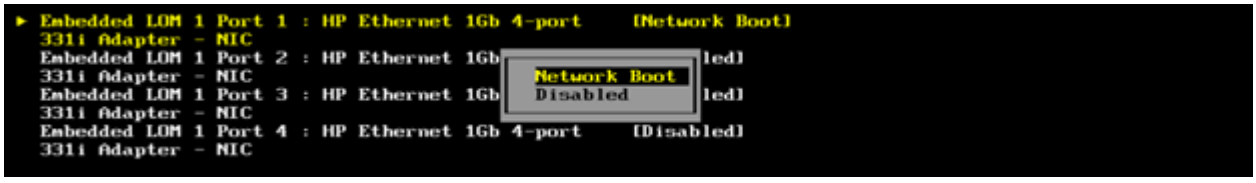
`$connection` contains one connection object.

- **Enable-HPEBIOSEmbeddedLOMPort and Disable-HPEBIOSEmbeddedLOMPort**
- **Set and Get PCI Device configuration for Gen10 servers**

Enable-HPEBIOSEmbeddedLOMPort and Disable-HPEBIOSEmbeddedLOMPort

IMPORTANT: All of the cmdlets that use Enable and Disable as a verb have the following unique characteristic: in context to RBSU, Enable and Disable cmdlets use features as parameter values and feature values as parameter names.

The following illustration shows RBSU EmbeddedLOMPort 1-4. All of the EmbeddedLOMPorts have two values: "Network Boot" and "Disabled".



The same feature can be configured using the cmdlets Enable-HPEBIOSEmbeddedLOMPort and Disable-HPEBIOSEmbeddedLOMPort.

Example A: Get the status of EmbeddedLOMPorts

```
PS C:\> $result = $connection | Get-HPEBIOSEmbeddedLOMPort

PS C:\> $result


IP                                                                    :
  10.20.30.40
        Hostname              : abc.domain.com
        Status                : Warning
```

```
        StatusInfo          : @{Category=PropertySupportability;
                              Message=The property or properties
                              listed in AffectedAttribute are not
                              supported on the target server
                              ProLiant DL380 Gen10. For more
                              details about supported properties,
                              see the cmdlet help or the
                              Troubleshooting chapter of BIOS
                              Cmdlets user guide.;
                              AffectedAttribute=System.Collections
                              .Generic.List`1[System.String]}
                              EnableNetworkBoot  : {NICBoot1}
                              DisableNetworkBoot : {NICBoot2,
                              NICBoot3, NICBoot4}

PS C:\> $result.EnableNetworkBoot
NICBoot1

PS C:\> $result.DisableNetworkBoot
NICBoot2
NICBoot3
NICBoot4
```

Example B: Enable the EmbeddedLOMPorts

```
PS C:\> Enable-HPEBIOSEmbeddedLOMPort -EnableNetworkBoot
 @(,@("NICBoot2", "NICBoot3", "NICBoot4"))
-Connection $connection
```

Example C: Verify the enabled EmbeddedLOMPort

```
PS C:\> $result = $connection | Get-HPEBIOSEmbeddedLOMPort

PS C:\> $result

        IP    :    10.20.30.40
        Hostname            : abc.domain.com
        Status              : Warning
        StatusInfo          : @{Category=PropertySupportability;
                              Message=The property or properties
                              listed in AffectedAttribute are
                              not supported on the target server
                              ProLiant DL380 Gen10. For more
                              details about supported properties,
                              see the cmdlet help or the
                              Troubleshooting chapter of BIOS
                              Cmdlets user guide.;
                              AffectedAttribute=System.Collections
                              .Generic.List`1[System.String]}
                              EnableNetworkBoot  : { NICBoot1,
                              NICBoot2, NICBoot3, NICBoot4}
                              DisableNetworkBoot : {}

    PS C:\> $result.EnableNetworkBoot
    NICBoot1
```

```
NICBoot2
NICBoot3
NICBoot4
```

Set and Get PCI Device configuration for Gen10 servers

The Set and Get PCI Device cmdlets are used to configure PCI devices and their related features. Each PCI device has following features:

- PCIeLinkSpeed
- PCIePowerManagement
- PCIeOptionROM
- Bifurcation

In RBSU, all the PCI devices are listed under PCI device configuration and each PCI device has related features. The following cmdlets are provided to get and set the PCI device configuration.

- Get-HPEBIOSPCIDeviceConfiguration
- Set-HPEBIOSPCIDeviceConfiguration

Examples in this section assume that a connection has been made with Connect-HPEBIOS and is in `$connection`.

`$connection` contains one connection object.

Example A: Get PCI device configuration

```
PS C:\> $result = $connection | Get-HPEBIOSPCIDeviceConfiguration
PS C:\> $result

IP         : 10.20.30.40
Hostname   : abc.domain.com
Status     : Warning
StatusInfo: @{Category=PropertySupportability; Message=The property or
            properties listed in AffectedAttribute are not supported on
            the target server ProLiant BL460c Gen10. For more details
            about supported properties, see the cmdlet help or the
            Troubleshooting chapter of BIOS Cmdlets user guide.;
            AffectedAttribute=System.Collections.Generic.Dictionary`2
            [System.String,System.Object]}
PCIDevice : {@{Name=EmbeddedSAS1; Status=Enabled; Type=EmbebdedSAS;
            PCIeLinkSpeed=Auto; PCIePowerManagement=Disabled;
            PCIeOptionROM=Disabled}, @{Name=EmbeddedNIC;
            Status=Enabled; Type=EmbeddedNIC; PCIeLinkSpeed=Auto;
            PCIePowerManagement=Disabled; PCIeOptionROM=Enabled},
            @{Name=FlexibleLOM1; Status=Enabled; Type=FlexibleLOM;
            PCIeLinkSpeed=Auto; PCIePowerManagement=Disabled;
            PCIeOptionROM=Enabled}, @{Name=EmbeddedSATA1;
            Status=Enabled; Type=EmbeddedSATA;
            PCIeOptionROM=Enabled}...}

PS C:\> $result.PCIDevice
Name                : EmbeddedSAS1
Status              : Enabled
Type                : EmbebdedSAS
```

```
PCIeLinkSpeed      : Auto
PCIePowerManagement: Disabled
PCIeOptionROM      : Disabled

Name               : FlexibleLOM1
Status             : Enabled
Type               : FlexibleLOM
PCIeLinkSpeed      : Auto
PCIePowerManagement: Disabled
PCIeOptionROM      : Enabled

Name               : EmbeddedSATA1
Status             : Enabled
Type               : EmbeddedSATA
PCIeOptionROM      : Enabled
Name               : EmbeddedSATA2
Status             : Enabled
Type               : EmbeddedSATA
PCIeOptionROM      : Enabled

Name               : PCISlot3
Status             : Enabled
Type               : PCI
PCIeLinkSpeed      : Auto
PCIePowerManagement: Disabled
```

Example B: Enable the PCI devices

Use the EnablePCIDevice switch parameter to enable the PCI device listed under PCIDevice.

```
PS C:\> Set-HPEBIOSPCIDeviceConfiguration -Connection $connection -PCIDevice
 @(,@("EmbeddedSAS1","EmbeddedNIC","EmbeddedSATA1","EmbeddedSATA2"))
```

You can get the list of supported PCI devices for Gen10 servers by using Enum:

```
[HPE.BIOS.PCIDeviceForGen10]
```

Example C: Disable PCI Devices

Use the DisablePCIDevice switch parameter to enable the PCI device listed under PCIDevice.

```
PS C:\> Set-HPEBIOSPCIDeviceConfiguration -Connection $connection -PCIDevice
 @(,@("EmbeddedSAS1","EmbeddedNIC","EmbeddedSATA1","EmbeddedSATA2"))
```

Example D: Configure PCI device-related features

The following cmdlet will configure the PCIeLinkSpeed and PCIePowerManagement for "EmbeddedSATA1" and "EmbeddedSATA2".

```
PS C:\> Set-HPEBIOSPCIDeviceConfiguration -Connection $connection -PCIDevice
 @(,@("EmbeddedSATA1","EmbeddedSATA2"))  -PCIeLinkSpeed @(,@
("Auto","PCIeGeneration1")) -PCIePowerManagement Disabled
```

NOTE: Input rules for the parameters (PCIeLinkSpeed, PCIePowerManagement, PCIeOptionROM, and Bifurcation):

- If you provide an equal number of values for a parameter and PCI devices, one-to-one mapping will be successful. Each PCI device will be set to the specified values.
- If any of the above mentioned parameters provides only one value, all the PCI devices will be set to that specific value.
- If you provide more values for a parameter than the number of PCI devices you provide, extra values will be discarded.
- If you provide fewer values for any of the above parameters than the number of PCI devices, there will be one-to-one mapping between the PCIDevice values and the other parameter values. The excess PCI devices will be ignored because there is nothing to set.

Enum Provided by BIOS cmdlets module

The BIOS cmdlets module provides `Enum` to get the list of supported values for the cmdlets where IntelliSense does show the values.

The following table lists the `Enums` provided. The namespace for `Enums` is under "[HPE.BIOS]".

Cmdlets where `Enum` can be used	**`Enum` provided by BIOS cmdlets**
Enable/Disable-HPEBIOSEmbeddedLOMPort	[HPE.BIOS.EmbeddedLOMPort]
Enable/Disable-HPEBIOSNVDIMMErase	[HPE.BIOS.NVDIMMErase]
Enable/Disable-HPEBIOSPCIDeviceOption	[HPE.BIOS.PCIDeviceForGen9]
Enable/Disable-HPEBIOSPCIeSlotNetworkBootOption	[HPE.BIOS.PCIeSlotNetworkBoot]
Set-HPEBIOSPCIDeviceConfiguration	[HPE.BIOS.PCIDeviceForGen10]
	[HPE.BIOS.PCIeLinkSpeed]
	[HPE.BIOS.PCIeOptionROM]
	[HPE.BIOS.PCIePowerManagement]
	[HPE.BIOS.Bifurcation]

Example: Enable the PCI device on a Gen9 target server using `Enum`

```
PS C:\> $connection | Enable-HPEBIOSPCIDeviceOption -EnablePCIDevice
@(,@([HPE.BIOS.PCIDeviceForGen9]::EmbeddedFlexibleLOM1.ToString(),
[HPE.BIOS.PCIDeviceForGen9]::EmbeddedSATAController1.ToString()))
```

Cmdlet, parameter, and parameter value supportability on target servers

All the BIOS cmdlets are not supported on all generations of servers. When a cmdlet that is not supported is used, an error message is returned. Familiarize yourself with the cmdlets that are supported on the target server before executing any cmdlets. BIOS cmdlets provide the Get-HPEBIOSCmdletInfo cmdlet, which provides a list of supported cmdlets on the target servers, along with supported parameters and their supported values. This cmdlet will help to write the production script.

Example

For connections, refer to connection Gen9 and Gen10 sections.

```
PS C:\> $supportedCmdlets = Get-HPEBIOSCmdletInfo -Connection $c

PS C:\> $supportedCmdlets

IP          : 10.20.30.20
HostName    : abc.domain.com
Status      : OK
CmdletInfo  : {[Get-HPEBIOSACPI_SLIT, System.Collections.Generic.Dictionary`2
              [System.String,System.Object]],
              [Set-HPEBIOSACPI_SLIT,System.Collections.Generic.Dictionary`2
              [System.String,System.Object]],
              [Get-HPEBIOSAdminInfo,System.Collections.Generic.Dictionary`2
              [System.String,System.Object]],
              [Set-HPEBIOSAdminInfo, System.Collections.Generic.Dictionary`2
              [System.String,System.Object]]...}
```

Check the cmdlet name using "CmdletInfo". If it contains the cmdlet name, the cmdlet is supported on the target server.

The following example shows how to check whether a cmdlet is supported. If the `CmdletInfo.ContainsKey()` command returns “true”, the cmdlet is supported.

```
PS C:\> $ret.CmdletInfo.ContainsKey("Set-HPEBIOSAdvancedMemoryProtection")
True
```

The following example illustrates checking supported parameters for the Set-HPEBIOSAdvancedMemoryProtection cmdlet.

```
PS C:\> $result = $supportedCmdlets.CmdletInfo
        ["Set-HPEBIOSAdvancedMemoryProtection"]
PS C:\> $result
Key   : Connection
Value : {}

Key   : AdvancedMemoryProtection
Value : {FastFaultTolerantA3DCSupport, AdvancedECCSupport,
        OnlineSpareAdvancedECCSupport,
        MirroredMemoryAdvancedECCSupport}
```

NOTE: The parameter name is stored as a key and its supported values are stored as values.

The following example illustrates checking supported parameter values.

```
PS C:\> $result[“AdvancedMemoryProtection”]
FastFaultTolerantA3DCSupport
AdvancedECCSupport
OnlineSpareAdvancedECCSupport
MirroredMemoryAdvancedECCSupport
```

NOTE:

- CmdletInfo is a dictionary where cmdlets and their supported parameters are stored as key Value pairs.
- Each key of cmdletInfo is a supported cmdlet and the value of corresponding key is Dictionary of supported parameters and their supported values. "CmdletInfo.Keys" contains all the cmdlets that are supported on the target server.
- If the parameter value does not support validate set, then supported value will be empty list.

Get BIOS base configuration, current configuration, and pending configuration

The cmdlet Get-HPEBIOSSetting returns all the BIOS configuration information for the target server. This information includes base configuration, current configuration, and pending configuration of the target server. You can also use this cmdlet to get the base configuration from one server and apply it to other servers. Get-HPEBIOSSetting is supported only on Gen9 and Gen10 servers.

Example

For connection information, see Gen9 and Gen10 in the connection section.

```
PS C:\> $BIOSSettings =  $connection | Get-HPEBIOSSetting
PS C:\> $BIOSSettings.BaseConfig
PS C:\> $BIOSSettings.CurrentBIOSSettings
PS C:\> $BIOSSettings.PendingBIOSSettings
PS C:\> $BIOSSettings.CurrentBIOSBootSettings
PS C:\> $BIOSSettings.PendingBIOSBootSettings
```

In the previous example:

- `$BIOSSettings.BaseConfig` is the base configuration of BIOS settings (factory default settings)
- `$BIOSSettings.CurrentBIOSSettings` is the current BIOS settings (applied on the system)
- `$BIOSSettings.PendingBIOSSettings` is the pending BIOS setting (that will apply after reboot)
- `$BIOSSettings.CurrentBIOSBootSettings` is the current boot setting of system (that is applied)
- `$BIOSSettings.PendingBIOSBootSettings` is the pending boot setting (that will apply after reboot)

Using parameters from a file

In some situations, it may be more convenient to manage scripts by having the parameters for calling them contained in an external file or database. This is especially true if either the list of systems to communicate with or the number of parameters to enter is unwieldy. PowerShell provides cmdlets that support many different types of input data and can convert them to internal PowerShell data objects. The following example illustrates the method of using a Comma-Separated Value (CSV) file.

CSV file input

CSV files are easy to create and maintain with Microsoft Excel or a text editor like Notepad. For this example script, the following CSV file is used.

```
ServerList.csv:

IP,Username,Password
192.168.1.2,Administrator,Admin
192.168.1.3,Administrator,Admin
192.168.1.4,Administrator,Admin
```

By using the Windows PowerShell embedded cmdlet `import-csv ServerList.csv`, all CSV data will be converted to one object collection by row. For information on the `import-csv` cmdlet, see **https://technet.microsoft.com/en-us/library/hh849891.aspx**.

The `import-csv` results are passed to Connect-HPEBIOS in the pipeline. All data fields supported by Connect-HPEBIOS will be used as input parameters for each object. In the following example there are three objects in the collection. All three objects will be used by Connect-HPEBIOS. Then it is available to pass the Connect-HPEBIOS result to Get-HPEBIOSPowerProfile in the pipeline.

CSV file input script

```
PS C:\> Import-csv ServerList.csv | Connect-HPEBIOS -DisableCertitificateAuthentication |
Get-HPEBIOSPowerProfile | Format-List
```

Script output

```
PS C:\> $PowerProfile


IP              : 192.168.243.100
Hostname        : server1.Company.net
Status          : OK
PowerProfile    : MaximumPerformance

IP              : 192.168.243.101
Hostname        : server2.Company.net
Status          : OK
PowerProfile    : BalancedPowerAndPerformance

IP              : 192.168.243.102
Hostname        : server3.Company.net
Status          : OK
PowerProfile    : Custom
```

Script writing methodology

When deciding to write a script, you generally know what you want to accomplish. One of the powerful features of PowerShell ISE is that you can build a script piece-by-piece. Along the way, you can test code and view objects to understand better how to accomplish your goals.

Here is a typical process you might want to use for creating PowerShell scripts.

Procedure

1. Determine what type of data you want to get.
2. Execute the appropriate cmdlet interactively to retrieve the data.
3. After viewing the command results, decide what part of the object you are interested in.
4. Create the main processing to manage BIOS by different cmdlets.
5. Summarize or output the data in the desired format.

If there are many steps, repeat the process until all the requirements of the data collection or setting have been completed.

When using data sources such as CSV files, XML files, or databases to store and retrieve data to use for targets, it may be necessary to provide their usernames and passwords. These may need to be encrypted for security purposes. Encrypted storage and data use is beyond the scope of this document. It is not a recommended practice to embed passwords in scripts; instead they can be prompted for by omitting them as a parameter. You must be cognizant of your organization's security policies and code accordingly.

Script examples

BIOS script examples are packaged along with the .msi installer and Readme First installation document. Beginning with BIOS 1.1, comprehensive PowerShell script examples are available on the Hewlett Packard GitHub repository at **https://github.com/HewlettPackard/PowerShell-ProLiant-SDK**.

Understanding the operating process

Each BIOS cmdlet is designed to follow a particular operating sequence when executed. The functions performed often depend upon certain prerequisites. The following cmdlet examples detail the flow of this process.

Connect-HPEBIOS process flow

When executed, the Connect-HPEBIOS cmdlet first detects whether the target is an iLO IP address or a ProLiant host with a Windows-loaded IP address.

If the target is an iLO IP address:

Connect-HPEBIOS identifies whether the server is Gen9, Gen10, or Gen10 Plus.

If the target iLO address is a Gen9, Gen10, or Gen10 Plus server, Connect-HPEBIOS verifies that the target iLO RESTful interfaces are accessible from the management client. It then returns the session object back to the user.

Expected result (both target types)

If the cmdlet fails to connect, it returns an error record with the reason for failure.

```
Connect-HPEBIOS : Failed for 192.168.243.56 abc.domain.com:Access is denied.
(Exception from HRESULT: 0x80070005 (E_ACCESSDENIED))
At line:1 char:6
+ $c = Connect-HPEBIOS -IP "192.168.243.56" -Username "Administrator" -Password "Abc ...
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : InvalidOperation: (:) [Connect-HPEBIOS], HPEBIOSException
    + FullyQualifiedErrorId : HPEBIOSCmdlets.ConnectHPEBIOS
```

If the cmdlet successfully connects to the target, it returns a connection object with information about the target system. The information will be used in later Get or Set cmdlet.

NOTE: Most of the properties are added into a structure titled `ConnectionInfo`.

```
DisableServerCertificateAuthenticationFlag : True
Location                                   : https://192.168.10.21
                                             /rest/v1/SessionService
                                             /Sessions/admin000689bcfa5e353f
RootUri                                    : https://192.168.10.21/rest/v1
RootData                                   : @{@odata.context=/redfish/v1
                                             /$metadata#SessionService/Sessions
                                             /Members/$entity; @odata.id=
                                             /redfish/v1/SessionService
                                             /Sessions/admin000689bcfa5e353f/;
                                             @odata.type=#Session.1.0.0.Session;
                                             Description=Manager User Session;
                                             Id=admin000689bcfa5e353f;
                                             Name=User Session; Oem=;
                                             Type=Session.1.0.0; UserName=admin;
                                             links=}
AttributeRegistry                          : HpBiosAttributeRegistryU20.1.1.40
ConnectionType                             : REST
IP                                         : 10.20.30.25
Hostname                                   : abc.domain.com
IsConnected                                : True
Username                                   : admin
ProductName                                : ProLiant DL160 Gen9
CurrentROMFamily                           : U20
CurrentROMDate                             : 02/17/2017
CurrentROMVersion                          : U20 v2.40 (02/17/2017)
BackupROMFamily                            : U20
BackupROMDate                              : 02/17/2017
BackupROMVersion                           : U20 v2.40 (02/17/2017)
ProcessorInfo                              : Intel(R) Xeon(R) CPU E5-2620
                                             v3 @ 2.40GHz
```

Disconnect-HPEBIOS process flow

1. When closing a connection using Disconnect-HPEBIOS, connection properties values are removed and files generated by this connection are deleted.
2. If the connection successfully closed or was previously disconnected, no message is displayed.

   If Disconnect-HPEBIOS is used during a Get/Set operation, the error message `already disconnected` appears.

BIOS cmdlets: Logs

You can enable the logging feature on BIOS cmdlets using the following cmdlet. By default, logging is disabled in this module.

```
PS C:\> Enable-HPEBIOSLog
LogFilePath
-----------

C:\Program Files (x86)\Hewlett Packard Enterprise\PowerShell\
Modules\HPEBIOSCmdlets\Logs\
HPEBIOSCmdlets_242018_235454584.log
```

If you want to switch off the logging feature, use the following cmdlet.

```
PS C:\> Disable-HPEBIOSLog
Logging disabled for the current session!
```

You can use the Clear-HPEBIOSLog cmdlet to clear all or the last *<<number>>* logs from the installed BIOS cmdlets directory.

```
PS C:\> Clear-HPEBIOSLog

PS C:\> Clear-HPEBIOSLog -Last 2
```

BIOS cmdlets and server security

HPE Gen10 security features help protect your hardware, firmware, and network components from unauthorized access and unapproved use. HPE offers an array of embedded and optional software and firmware for HPE Gen10 that enables you to institute the best mix of remote access and control for your network and data center.

Use the following BIOS cmdlets to configure UEFI security features:

Cmdlets	Gen10 server UEFI security feature
Set-HPEBIOSPowerOnPassword Reset-HPEBIOSPowerOnPassword	Power On Password
Set-HPEBIOSAdminPassword Reset-HPEBIOSAdminPassword	Admin Password
Get-HPEBIOSTPMConfiguration Set-HPEBIOSTPMConfiguration Get-HPEBIOSTPMChipInfo	TPM1.2 and TPM2.0 UEFI Option ROM measurement
Get-HPEBIOSServerSecurity Set-HPEBIOSServerSecurity	Intel® TXT Support Intelligent Provisioning (**F10** prompt) disable Processor AES-NI Support
Get-HPEBIOSSecureBootState Set-HPEBIOSSecureBootState	Secure Boot
Get-HPEBIOSNetworkBootOption Set-HPEBIOSNetworkBootOption	TLS (HTTPS) Boot
Get-HPEBIOSSATAControllerOption Set-HPEBIOSSATAControllerOption	SATA Secure Erase

See the HPE Gen10 Security Reference Guide for detailed information about the HPE Gen10 UEFI security features and setting up security on your system: **http://www.hpe.com/support/gen10-security-ref-en**

Troubleshooting

General issues

Verifying BIOS cmdlet version

Solution: If a problem occurs, verify that the most current version of the BIOS cmdlets is installed. Updating to the most current version might solve the problem.

To determine if there is a newer version of the BIOS cmdlets is available, see **Using the Get-HPEBIOSModuleVersion cmdlet** .

Remote server returns `(401) Unauthorized`

You may encounter an `(401) Unauthorized` error when attempting to Get/Set BIOS configurations over a stale target server connection. Stale connections result on established connections where no activity takes place over a prolonged period. Connection time-out is based on iLO connection time-out.

Solution: Reconnect with Connect-HPEBIOS and execute the BIOS cmdlets.

Parameter supportability and supported values

Symptom

To find out whether a parameter or its values are supported, perform the action below.

Action

Execute Get-HPEBIOSCmdletInfo to determine the supportability of parameters and their supported values.

For more information, see **Cmdlet, parameter, and parameter value supportability on target servers**.

Usage tips

Input parameter matching

For input parameters, arrays of objects are supported. If an array of objects is provided to both parameter A and parameter B of a BIOS cmdlet, it matches input values for parameter A with values for parameter B.

In the following example, `Get-HPEBIOSXXXX` has a mandatory parameter (named "`MandatoryParam`"), the execution result of each case is added as comments before each case.

```
#"value1" is used with $con1 and "value2" is used with $con2
Get-HPEBIOSXXXX -Connection @($con1, $con2) -MandatoryParam @("value1", "value2")

#"value1" is used with $con1, "value2" is used with $con2, and "value3" is discarded
Get-HPEBIOSXXXX -Connection @($con1, $con2) -MandatoryParam @("value1", "value2", "value3")

#"value" is used with both $con1 and $con2
Get-HPEBIOSXXXX -Connection @($con1, $con2) -MandatoryParam "value"

#"value1" is with for $con1 and an error record is written for $con2 because the Force parameter is used.
#(With the Force parameter, users will not be asked to input data but will get an error.)
Get-HPEBIOSXXXX -Connection @($con1, $con2) -MandatoryParam @("value1") -Force
```

In the following example, `Get-HPEBIOSYYYY` has an optional parameter (named "`OptionalParam`").

```
#”value1” is used with $con1, ”value2” is used with $con2
Get-HPEBIOSYYYY -Connection @($con1, $con2) -OptionalParam @(“value1”, “value2”)

#”value1” is used with $con1, ”value2” is used with $con2, and “value3” is discarded
Get-HPEBIOSYYYY -Connection @($con1, $con2) -OptionalParam @(“value1”, “value2”, “value3”)

#”value” is used with both $con1 and $con2
Get-HPEBIOSYYYY -Connection @($con1, $con2) -OptionalParam “value”

#Value3 is used with $con1, value4 is used with $con2. value1 and value2 are not used.
#Priority is given to values from the commandline if both pipeline and commandline have the values.
$p1 = New-Object -TypeName PSObject -Property @{ "Connection"=$con1;"Parameter"=value1} ;
$p2 = New-Object -TypeName PSObject -Property @{ "Connection"=$con2;"Parameter"=value2} ;
,@($p1,$p2) | Get-HPEBIOSxxx -Parameter @(Value3, Value4)
```

Websites

General websites

Hewlett Packard Enterprise Information Library

www.hpe.com/info/EIL

Single Point of Connectivity Knowledge (SPOCK) Storage compatibility matrix

www.hpe.com/storage/spock

Storage white papers and analyst reports

www.hpe.com/storage/whitepapers

For additional websites, see **Support and other resources**.

Windows PowerShell websites

The following websites provide useful information for using PowerShell.

Microsoft Script Center

http://technet.microsoft.com/en-us/scriptcenter/default

Windows PowerShell Blog

http://blogs.msdn.com/b/powershell/

PowerShell.com

http://powershell.com/cs/

PowerShell.org

http://powershell.org/

PowerShell Magazine

http://www.powershellmagazine.com/

Support and other resources

Accessing Hewlett Packard Enterprise Support

- For live assistance, go to the Contact Hewlett Packard Enterprise Worldwide website:

  https://www.hpe.com/info/assistance

- To access documentation and support services, go to the Hewlett Packard Enterprise Support Center website:

  https://www.hpe.com/support/hpesc

Information to collect

- Technical support registration number (if applicable)
- Product name, model or version, and serial number
- Operating system name and version
- Firmware version
- Error messages
- Product-specific reports and logs
- Add-on products or components
- Third-party products or components

Accessing updates

- Some software products provide a mechanism for accessing software updates through the product interface. Review your product documentation to identify the recommended software update method.
- To download product updates:

  Hewlett Packard Enterprise Support Center

  https://www.hpe.com/support/hpesc

  Hewlett Packard Enterprise Support Center: Software downloads

  https://www.hpe.com/support/downloads

  My HPE Software Center

  https://www.hpe.com/software/hpesoftwarecenter

- To subscribe to eNewsletters and alerts:

  https://www.hpe.com/support/e-updates

- To view and update your entitlements, and to link your contracts and warranties with your profile, go to the Hewlett Packard Enterprise Support Center **More Information on Access to Support Materials** page:

  https://www.hpe.com/support/AccessToSupportMaterials

IMPORTANT: Access to some updates might require product entitlement when accessed through the Hewlett Packard Enterprise Support Center. You must have an HPE Passport set up with relevant entitlements.

Remote support

Remote support is available with supported devices as part of your warranty or contractual support agreement. It provides intelligent event diagnosis, and automatic, secure submission of hardware event notifications to Hewlett Packard Enterprise, which will initiate a fast and accurate resolution based on your product's service level. Hewlett Packard Enterprise strongly recommends that you register your device for remote support.

If your product includes additional remote support details, use search to locate that information.

Remote support and Proactive Care information

HPE Get Connected

https://www.hpe.com/services/getconnected

HPE Proactive Care services

https://www.hpe.com/services/proactivecare

HPE Datacenter Care services

https://www.hpe.com/services/datacentercare

HPE Proactive Care service: Supported products list

https://www.hpe.com/services/proactivecaresupportedproducts

HPE Proactive Care advanced service: Supported products list

https://www.hpe.com/services/proactivecareadvancedsupportedproducts

Proactive Care customer information

Proactive Care central

https://www.hpe.com/services/proactivecarecentral

Proactive Care service activation

https://www.hpe.com/services/proactivecarecentralgetstarted

Warranty information

To view the warranty information for your product, see the links provided below:

HPE ProLiant and IA-32 Servers and Options

https://www.hpe.com/support/ProLiantServers-Warranties

HPE Enterprise and Cloudline Servers

https://www.hpe.com/support/EnterpriseServers-Warranties

HPE Storage Products

https://www.hpe.com/support/Storage-Warranties

HPE Networking Products

https://www.hpe.com/support/Networking-Warranties

Regulatory information

To view the regulatory information for your product, view the *Safety and Compliance Information for Server, Storage, Power, Networking, and Rack Products*, available at the Hewlett Packard Enterprise Support Center:

https://www.hpe.com/support/Safety-Compliance-EnterpriseProducts

Additional regulatory information

Hewlett Packard Enterprise is committed to providing our customers with information about the chemical substances in our products as needed to comply with legal requirements such as REACH (Regulation EC No 1907/2006 of the European Parliament and the Council). A chemical information report for this product can be found at:

https://www.hpe.com/info/reach

For Hewlett Packard Enterprise product environmental and safety information and compliance data, including RoHS and REACH, see:

https://www.hpe.com/info/ecodata

For Hewlett Packard Enterprise environmental information, including company programs, product recycling, and energy efficiency, see:

https://www.hpe.com/info/environment

Documentation feedback

Hewlett Packard Enterprise is committed to providing documentation that meets your needs. To help us improve the documentation, send any errors, suggestions, or comments to Documentation Feedback (**docsfeedback@hpe.com**). When submitting your feedback, include the document title, part number, edition, and publication date located on the front cover of the document. For online help content, include the product name, product version, help edition, and publication date located on the legal notices page.