Hewlett Packard
Enterprise

Scripting Tools for Windows PowerShell User Guide

iLO cmdlets v1.5

Abstract

This document contains instructions for using Scripting Tools for Windows PowerShell to manage iLO. It is intended for system administrators who use the Scripting Tools for Windows PowerShell to manage their IT infrastructure.

Part Number: 745004-007a
Published: October 2017
Edition: 2

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 either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Contents

Introduction to Scripting Tools for Windows PowerShell

The Scripting Tools for Windows PowerShell provides a simplified and consistent infrastructure management experience. This set of PowerShell utilities provides a comprehensive set of integration tools. It is designed for IT experts with experience in PowerShell scripting and configuring ProLiant server hardware.

The Scripting Tools for Windows PowerShell includes a set of PowerShell cmdlets for configuring hardware using familiar PowerShell syntax. Documentation describing how to apply these new tools to configure hardware is also included.

This guide is intended for system administrators who use the Scripting Tools for Windows PowerShell to manage their IT infrastructure. Users should be familiar with Windows PowerShell and iLO. For more information about iLO, see the HPE iLO 4 user guide and other related iLO documents on the iLO information library (**http://www.hpe.com/info/ilo/docs**).

This guide describes the iLO cmdlets included in version 1.5 of Scripting Tools for Windows PowerShell. For information on the cmdlets added in version 1.5, see the *Scripting Tools for Windows PowerShell Release Notes iLO Cmdlets v1.5*.

Windows PowerShell

Windows PowerShell is Microsoft's task automation framework, consisting of a command-line shell and associated scripting language built on a .NET Framework. As 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 hardware. 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: ILO Cmdlets provides the following features:

- Support for iLO 3 and iLO 4 configuration.
- Support for iLO Federation cmdlets.
- Enhanced security support for all cmdlets with Server Certificate check.
- Proven PowerShell technology to provide consistent and reliable server configuration.
- Support for the standard PowerShell architecture and scripting model.
- Object oriented—output from one command can be piped to another command.
- Built-in help for all PowerShell cmdlets, documenting syntax and usage examples.
- Supports updating iLO, BIOS, and server firmware components.
- Accepts RIBCL commands as input parameters.

Installation

System prerequisites

Before loading Scripting Tools for Windows PowerShell on your system, two prerequisites are necessary: Microsoft .NET Framework and Windows Management Framework. Make sure that you read and understand the system requirements and other information provided on the links that follow.

1. Install Microsoft .NET Framework 4.5 or later.

   Microsoft .NET Framework 4.5

2. Install 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

Scripting Tools for Windows PowerShell: iLO Cmdlets are supported on the following operating system versions:

- Microsoft Windows 7 SP1
- Microsoft Windows 8.1
- Microsoft Windows 10
- Microsoft Windows Server 2008 R2 SP1
- Microsoft Windows Server 2012 R2
- Microsoft Windows Server 2016

IMPORTANT:

Using multiple targets in a single cmdlet on a 32-bit operating system can produce unsatisfactory cmdlet output if there are more than 100 target IP addresses. HPE recommends using a 64-bit operating system to run the cmdlets with more than 100 targets.

Installing Scripting Tools for Windows PowerShell - iLO Cmdlets

The Scripting Tools for Windows PowerShell: iLO Cmdlets can be downloaded from the following website: **http://www.hpe.com/servers/powershell**.

Using the following guidelines when installing Scripting Tools for Windows PowerShell:

- Close any PowerShell windows before the installation and open new ones after the installation is complete.
- The installer should be run from an account with administrative privilege using any normal method of execution (command line or double click). If you run from an account without administrative privilege, use one of the following options:

Procedure 1

1. Launch Scripting Tools for Windows PowerShell ISE application using the **Run as administrator** option.
2. Browse (CD) to the directory where you unzipped the installer.
3. On the PowerShell command line run either the 64-bit (`HPiLOCmdlets-x64.msi`) or the 32-bit (`HPiLOCmdlets-x86.msi`) installer as appropriate for your system.

Procedure 2

1. Click **Start** and select **Run...**
2. In the Run dialog enter the path and filename of the correct installer for your system, either the 64-bit (`HPiLOCmdlets-x64.msi`) or the 32-bit (`HPiLOCmdlets-x86.msi`).

- It might be necessary to change the execution policy for PowerShell. Use the following help command to get more information 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
  ```
- Upgrading from a previous release is supported.
- The installation will halt and not complete successfully if any of the following conditions are detected:
  - Attempting to install the X86 package on a 64-bit operating system
  - Attempting to install without .NET 4.5 or above
  - Attempting to install without PowerShell 3.0 or above

Uninstalling Scripting Tools for Windows PowerShell - iLO Cmdlets

To uninstall the Scripting Tools for Windows PowerShell: iLO Cmdlets:

Procedure

1. Open Windows Control Panel.
2. Select **Programs and Features**.

3. Select **Scripting Tools for Windows PowerShell: iLO Cmdlets**.
4. Click **Uninstall**.

Scripting Tools for Windows PowerShell cmdlets

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

Cmdlet help

The iLO cmdlets include help support similar to other PowerShell cmdlet help. To display a list of the iLO cmdlets, type:

```
help *hpilo*
```

NOTE:

You can also use the following command to display the iLO cmdlets:

```
Get-Command –Module HPiLOCmdlets
```

To display complete help for a specific cmdlet, type:

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

Where `<cmdlet>` is the name of the iLO cmdlet

The iLO cmdlets support the PowerShell `Update-Help` feature. This command retrieves the most current help files from an HPE website and puts them in the correct location on your system.

Table 1: iLO cmdlets

Cmdlet	Description
Add-HPiLOFederationGroup	Adds an iLO Federation group or includes an iLO in an existing group membership.
Add-HPiLOSSORecord	Adds an HPE SIM Single Sign-On (SSO) Server record.
Add-HPiLOUser	Adds a local user account to the iLO.
Clear-HPiLOAHSData	Clears the AHS information from the AHS logs.
Clear-HPiLOEventLog	Clears the iLO event logs.
Clear-HPiLOIML	Clears the integrated management logs.
Clear-HPiLOPowerOnTime	Clears the virtual clock counter without power-cycling the server.
Disable-HPiLOCertificateAuthentication	Disables server certificate authentication in the current PowerShell session.
Disable-HPiLOERSIRSConnection	Disables Insight Remote Support functionality and unregisters the server.
Disable-HPiLOSecurityMessage	Disables the display of security text message in the iLO login banner.

Table Continued

Cmdlet	Description
Dismount-HPiLOVirtualMedia	If one is mounted, the command will dismount the virtual media image.
Enable-HPiLOCertificateAuthentication	Enables server certificate authentication in the current PowerShell session.
Enable-HPiLOFIPS	Enables the Federal Information Processing Standard *Enforce AES/3DES Encryption* setting.
Enable-HPiLOERSIRSConnection	Connects to the Insight Remote Support server and registers the server.
Enable-HPiLOSecurityMessage	Enables the security text message in the iLO login banner.
Find-HPiLO	Finds list of valid iLO servers in a specified subnet.
Get-HPiLOServerFQDN	Retrieves the fully qualified domain name of the host server.
Get-HPiLOSMHFQDN	Retrieves the fully qualified domain name of System Management Homepage (SMH).
Get-HPiLOAHSStatus	Gets the AHS status.
Get-HPiLOAssetTag	Gets the asset tag.
Get-HPiLOCertificateSigningRequest	Gets the Certificate Signing Request (CSR) status.
Get-HPiLOCriticalTemperatureAction	Retrieves the configured critical temperature shutdown behavior of the server.
Get-HPiLOCurrentBootMode	Retrieves the current boot mode of the server.
Get-HPiLODefaultLanguage	Gets the default language on iLO.
Get-HPiLODirectory	Gets the current directory configuration.
Get-HPiLODriveInfo	Gets the drive details for the server.
Get-HPiLOEncryptKeyManagerSetting	Displays the Enterprise Secure Key Manager (ESKM) settings configured in iLO for the specified server.
Get-HPiLOERSSetting	Gets the Insight Remote Support setting.
Get-HPiLOEventLog	Gets the iLO event logs.
Get-HPiLOFan	Gets the fan health details from the server.
Get-HPiLOFederationGroup	Gets a list of all iLO Federation group names.

Table Continued

Cmdlet	Description
Get-HPiLOFederationMulticast	Gets the iLO Federation status and multicast options.
Get-HPiLOFIPSStatus	Gets the current Enforce AES/3DES Encryption status.
Get-HPiLOFirmwareInfo	Gets the firmware details for the iLO.
Get-HPiLOFirmwareVersion	Gets the iLO firmware version.
Get-HPiLOGlobalSetting	Gets the iLO global settings.
Get-HPiLOHealthSummary	Gets the health information summary of the host server.
Get-HPiLOHostAPO	Gets the current automatic power on and power-on delay settings.
Get-HPiLOHostData	Gets the host data displayed on the server information page.
Get-HPiLOHostPower	Gets the power state of the server.
Get-HPiLOHostPowerMicroVersion	Gets the power micro version number.
Get-HPiLOHostPowerSaver	Gets the state of the processor power regulator feature of the server.
Get-HPiLOHotkeyConfig	Gets hotkeys available for use in remote console sessions.
Get-HPiLOIML	Gets the integrated management logs.
Get-HPiLOLanguage	Gets all languages on iLO.
Get-HPiLOLicense	Gets license types and keys.
Get-HPiLOMemoryInfo	Gets the memory details for the host server where iLO is located.
Get-HPiLOModuleVersion	Gets the module details for the iLO cmdlets.
Get-HPiLONetworkSetting	Gets the current network settings.
Get-HPiLONICInfo	Gets the NIC details for the system NIC and the iLO NIC.
Get-HPiLOOAInfo	Gets the Onboard Administrator information from the enclosure where iLO is located.
Get-HPiLOOneTimeBootOrder	Gets the current state of the one-time boot.
Get-HPiLOPendingBootMode	Retrieves the pending boot mode that will become active on the next server reboot.
Get-HPiLOPersistentBootOrder	Gets the current boot order.

Table Continued

Cmdlet	Description
Get-HPiLOPersistentMouseKeyboard	Gets the persistent mouse and keyboard status.
Get-HPiLOPowerAlertThreshold	Gets the power alert threshold for the iLO devices.
Get-HPiLOPowerCap	Gets the power cap of the server.
Get-HPiLOPowerOnTime	Gets the virtual clock value, in minutes, since the server was last powered on.
Get-HPiLOPowerReading	Gets the power readings from the server power supply.
Get-HPiLOPowerSupply	Gets the power supply details for the host server where the iLO is located.
Get-HPiLOProcessor	Gets the processor details for the host server.
Get-HPiLOProfile	Gets all the profile descriptors and the data stored in them in the perm directory of the blobstore.
Get-HPiLOProfileApplyResult	Gets the result of the Invoke-HPiLOProfileApply cmdlet.
Get-HPiLORackSetting	Gets the rack settings for an iLO.
Get-HPiLOSDCardStatus	Determines whether an internal secure digital (SD) card is connected to the server.
Get-HPiLOSecurityMessage	Gets the security message for the iLO login screen.
Get-HPiLOServerInfo	Gets the embedded health of the host server.
Get-HPiLOServerName	Gets the host server name used by iLO.
Get-HPiLOSessionTimeOut	Retrieves the RIBCL web request timeout set for the current session.
Get-HPiLOSNMPIMSetting	Gets the respective iLO SNMP IM settings.
Get-HPiLOSpatial	Gets the location information and system data with HPE Asset Manager to obtain more precise and complete asset data.
Get-HPiLOSSOSetting	Gets the SSO setting for the iLO.
Get-HPiLOStorageController	Gets the storage controller status of the server.
Get-HPiLOSupportedBootMode	Retrieves the list of boot modes that the server supports.
Get-HPiLOTemperature	Gets the temperature health details of the server.
Get-HPiLOTPMStatus	Retrieves the HPE Trusted Platform Module status.

Table Continued

Cmdlet	Description
Get-HPiLOUIDStatus	Gets the UID status of the server.
Get-HPiLOUser	Gets the local user information.
Get-HPiLOUserInfo	Gets all local user information.
Get-HPiLOUserList	Gets the list of local users.
Get-HPiLOVMStatus	Gets the virtual media drive status.
Get-HPiLOVRM	Gets the Voltage Regulator Module (VRM) health details from the server.
Import-HPiLOCertificate	Imports a signed certificate into iLO.
Import-HPiLOSSHKey	Imports an SSH key and associated username into iLO.
Install-HPiLOLanguagePack	Installs an iLO language pack, which enables you to change the iLO web interface from English to another supported language.
Invoke-HPiLORIBCLCommand	Invokes the RIBCL command provided by the user and returns the results in PSObject form.
Invoke-HPiLOProfileApply	Applies a deployment setting profile in iLO 4.
Invoke-HPiLOProfileDownload	Modifies a profile description, downloads a specific blob, and writes the blob to the blob store.
Invoke-HPiLOSNMPTestTrap	Invokes iLO to send a test SNMP trap to the configured alert destinations (IP).
Mount-HPiLOVirtualMedia	Mounts the specified media image.
Read-HPiLOSMBIOSRecord	Decodes System Management BIOS (SMBIOS) records from `Get-HPiLOHostData` output, which is encoded in Base64 format.
Register-HPiLOERSDirectConnect	Registers a supported server for Insight Online direct connect remote support.
Remove-HPiLOFederationGroup	Removes the iLO from an iLO Federation group membership.
Remove-HPiLOProfile	Deletes a deployment profile.
Remove-HPiLOSSORecord	Removes a SIM Trusted SSO Server record.
Remove-HPiLOUser	Removes an existing local user account.
Remove-HPiLOUserSSHKey	Deletes any SSH keys associated with a specified user login.

Table Continued

Cmdlet	Description
Reset-HPiLOAdministratorPassword	Resets the administrator password to the specified value.
Reset-HPiLORIB	Resets the iLO.
Reset-HPiLOServer	Resets the host server on which the iLO is operating.
Set-HPiLOServerFQDN	Sets the server fully qualified domain name (FQDN) or the host server IP address.
Set-HPiLOSMHFQDN	Sets the fully qualified domain name for SMH and allows it to be placed at a different FQDN or IP address.
Set-HPiLOAHSStatus	Enables or disables AHS logging.
Set-HPiLOAssetTag	Sets or clears the asset tag.
Set-HPiLOBrownout	Turns the brownout recovery feature on or off.
Set-HPiLOComputerLockConfig	Configures the lock settings of the computer.
Set-HPiLOCriticalTemperatureAction	Sets the server to power on or power off after shutting down due to critical temperature.
Set-HPiLODefaultLanguage	Sets the default language on iLO.
Set-HPiLODirectory	Modifies the directory settings on iLO.
Set-HPiLOEncryptKeyManagerSetting	Sets the communication settings for the Enterprise Secure Key Manager (ESKM) in iLO.
Set-HPiLOERSWebProxy	Sets the web proxy settings for servers that use Insight Online direct connect remote support.
Set-HPiLOFactoryDefault	Sets the iLO device to factory default settings.
Set-HPiLOFederationGroup	Modifies an existing iLO Federation group.
Set-HPiLOFederationMulticast	Enables iLO Federation and sets iLO Federation multicast operations.
Set-HPiLOGlobalSetting	Modifies global settings of the host server.
Set-HPiLOHostAPO	Sets the automatic power-on and power-on delay settings.
Set-HPiLOHostPower	Toggles the power on host server.
Set-HPiLOHostPowerSaver	Sets the power regulator setting for the server processor.
Set-HPiLOHotkeyConfig	Configures the remote console hot key settings in iLO.

Table Continued

Cmdlet	Description
Set-HPiLOIPv6NetworkSetting	Modifies the iLO IPv6 network setting.
Set-HPiLOKerberosConfig	Configures the Kerberos authentication.
Set-HPiLOLicenseKey	Applies a license key for the iLO advanced pack.
Set-HPiLOLockConfiguration	Enables the data center configuration lock for iLO.
Set-HPiLONetworkSetting	Modifies the iLO IPv4 network settings.
Set-HPiLOOneTimeBootOrder	Sets the one-time boot order.
Set-HPiLOPassword	Changes the password of a local user.
Set-HPiLOPendingBootMode	Sets the boot mode that will take effect when the server is rebooted.
Set-HPiLOPersistentBootOrder	Sets the persistent boot order.
Set-HPiLOPersistentMouseKeyboard	Enables or disables the persistent mouse and keyboard setting.
Set-HPiLOPowerAlertThreshold	Sets the power alert threshold value for the iLO.
Set-HPiLOPowerCap	Sets the power cap feature on the host server.
Set-HPiLORBSUPOSTIP	Configures the management processor RBSU to display the IP address during POST.
Set-HPiLOSchemalessDirectory	Modifies the current schemaless directory configuration on iLO.
Set-HPiLOServerName	Assigns the server name attribute shown in the user interface and host RBSU.
Set-HPiLOSessionTimeOut	Sets the RIBCL web request timeout to all cmdlets for the current PowerShell session.
Set-HPiLOSharedNetworkPort	Configures iLO device to pass network traffic on the shared host network port.
Set-HPiLOSNMPIMSetting	Modifies the respective iLO SNMP IM settings.
Set-HPiLOSSOSetting	Modifies the SSO settings for the iLO.
Set-HPiLOUIDStatus	Toggles the UID on host servers.
Set-HPiLOUser	Modifies an existing local user account present in the iLO.
Set-HPiLOVirtualPowerButton	Simulates the physical press of the power button, press and hold of the power button, and cold boot and warm boot of the server.

Table Continued

Cmdlet	Description
Set-HPiLOVLAN	Configures the iLO shared network port with a user-defined VLAN ID.
Set-HPiLOVMStatus	Sets the virtual media drive status.
Set-HPiLOVMPortSetting	Configures the virtual media port functionality for the iLO.
Start-HPiLOERSAHSSubmission	Initiates Active Health System data submission to the Insight Remote Support server.
Start-HPiLOL2Collection	Initiates a collection submission to the Insight Remote Support server.
Start-HPiLOTestEvent	Initiates a test service event submission to the Insight Remote Support server.
Test-HPiLOCertificateAuthentication	Tests the status of the server certificate authentication setting.
Test-HPiLODirectoryUserAuthentication	Validates the configured directory settings for a specified Active Directory domain user.
Update-HPiLOServerFirmware	Copies a specified firmware image to iLO and starts the update process. Once the firmware is flashed successfully, you must reboot the server for the changes to take effect.
Update-HPiLOFirmware	Updates the iLO firmware.
Update-HPiLOModuleVersion	Checks whether a newer version of Scripting Tools for Windows PowerShell: ILO Cmdlets is available on the website for download.

IPv6 support

Consider the following when using IPv6.

- IPv6 is supported in addition to 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 are supported with the comma character. For example, `1a00,1b00::1fe8` resolves to two addresses, `1a00::1fe8` and `1b00::1fe8`.
- Examples in this document use IPv4 but could use IPv6 instead if supported in the network. Both IPv4 and IPv6 addresses can be used within one cmdlet.

For more information on IPv6, see the following website or the references it links to:

http://en.wikipedia.org/wiki/IPv6.

Multithreaded operation of cmdlets

In early releases of Scripting Tools for Windows PowerShell, running one cmdlet that sent data to multiple targets (either iLO or OA) resulted in a significant amount of time spent waiting for responses. This time was a result of normal network delays and device response and data collection delays. But when added together in performing each operation serially, it resulted in a significant amount of time to perform operations on a large number of targets. To avoid this situation, multithreading has been implemented. When using multithreading, commands are sent to each target in parallel during the operation of one cmdlet and responses are waited for in parallel. Multithreading provides a significant performance improvement. Most commands that support multiple targets use multithreading for both iLO and OA cmdlets.

Up to 256 threads are used. This number was 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, and other network traffic.

To take advantage of multithreading, a single cmdlet is used but it is directed it to multiple targets in a single invocation by passing parameter sets as an array. Multiple threads are used automatically when you do this.

For example, the following executes a single invocation of the cmdlet, passing one parameter set at a time. This does not take advantage of multithreading.

```
foreach ($parameterset in $arrayofparametersets) {
 $parameterset | Get-HPThing
}
```

In order to take advantage of multithreading, send a cmdlet an array of parameter sets in a single invocation. The following uses multithreading, sending commands to up to 256 targets in parallel.

```
$arrayofparametersets | Get-HPThing
```

Using the `Find-HPiLO` cmdlet

When learning about the iLO cmdlets, a good place to start is with the `Find-HPiLO` cmdlet. This cmdlet scans IP addresses and finds iLOs that exist within the specified range. The `Range` parameter can be a single IP address, a subnet list, or a range of IP addresses. When the command finds an iLO, it obtains basic information about the iLO without requiring a username or password. This can be useful for performing a quick inventory within a data center, or perhaps determining what firmware versions exist. The information is returned as a single object or as an array of objects of iLOs found.

The following is an example of using `Find-HPiLO` with the `-Full` option:

```
Find-hpilo 10.1.11.5 -Full

WARNING: It might take a while to search for all the iLOs if the input
         is a very large range or iLO response is delayed.

IP                      : 10.1.11.5
HOSTNAME                :
HSI_SBSN                : 2M25450CRS
HSI_SPN                 : ProLiant BL460c Gen9
HSI_UUID                : 7270312M25450CRS
HSI_SP                  : 0
HSI_cUUID               : 30373237-3133-4D32-3235-343530435253
HSI_VIRTUAL_STATE       : Inactive
HSI_VIRTUAL_VID         : {@{BSN=; cUUID=}}
HSI_PRODUCTID           :  727031-B21
```

```
HSI_NICS                 :
MP_ST                    : 1
MP_PN                    : Integrated Lights-Out 4 (iLO 4)
MP_FWRI                  : 2.50
MP_BBLK                  :
MP_HWRI                  : ASIC: 17
MP_SN                    : ILO2M25450CRS
MP_UUID                  : ILO7270312M25450CRS
MP_IPM                   : 1
MP_SSO                   : 1
MP_PWRM                  : UNKNOWN
MP_ERS                   : 0
MP_EALERT                : 1
BLADESYSTEM_BAY          : 2
BLADESYSTEM_MANAGER      : @{TYPE=Onboard Administrator;
MGMTIPADDR=15.212.141.150;

MGMTIPv6ADDR1=FE54::4FE2:4AAB:34AC:FEFF:DEDF;
MGMTIPv6ADDR2=FE80::3A63:BBFF:FE31:66AF;

RACK=UnnamedRack; ENCL=1Z34AB7890; ST=2}
SPATIAL_DISCOVERY_RACK   : Not Supported
SPATIAL_DISCOVERY_DATA   : Server does not support Location Discovery
Services
SPATIAL_TAG_VERSION      : 0
SPATIAL_RACK_ID          : 0
SPATIAL_RACK_ID_PN       : 0
SPATIAL_RACK_DESCRIPTION : 0
SPATIAL_RACK_UHEIGHT     : 0
SPATIAL_UPOSITION        : 0
SPATIAL_ULOCATION        : 0
SPATIAL_cUUID            : 30373237-3133-4D32-3235-343530435253
SPATIAL_UHEIGHT          : 10.00
SPATIAL_UOFFSET          : 2
SPATIAL_BAY              : 2
SPATIAL_ENCLOSURE_cUUID  : 53553930-3645-3733-3152-4137 0 0 0 0 0
0454552464B415453
HEALTH_STATUS            : 4
```

The following is an example of using `Find-HPiLO` with a single IP address:

```
Find-HPiLO 192.168.1.1

Warning  : It might take a while to search for all the iLOs if the
           input is a very large range or iLO response is delayed.

IP       : 192.168.1.1
SPN      : ProLiant ML310e Gen8
FWRI     : 1.30
PN       : Integrated Lights-Out 4 (iLO 4)
HOSTNAME : ilomx2232004p.company.net
SerialNumber: 6CU4100JM1
UUID     : 30313436-3631-4336-5534-3130304A4D31
```

The following is an example of using `Find-HPiLO` with a search range which checks 11 addresses, in which three iLOs are found:

```
Find-HPiLO 192.168.1.1-11

Warning  : It might take a while to search for all the iLOs if the
           input is a very large range or iLO response is delayed.

IP       : 192.168.1.2
SPN      : ProLiant DL120 G7
FWRI     : 1.28
PN       : Integrated Lights-Out 3 (iLO 3)
HOSTNAME : ilohostv8.company.net
SerialNumber: 6CU4100JM1
UUID     : 30313436-3631-4336-5534-3130304A4D31

IP       : 192.168.1.4
SPN      : ProLiant MicroServer Gen8
FWRI     : 1.20
PN       : Integrated Lights-Out 4 (iLO 4)
HOSTNAME : fbtilodns.company.net
SerialNumber: 6CU4100JM1
UUID: 30313436-3631-4336-5534-3130304A4D31

IP       : 192.168.1.10
SPN      : ProLiant DL360p Gen8
FWRI     : 1.40
PN       : Integrated Lights-Out 4 (iLO 4)
HOSTNAME : ilohostbc.company.net
SerialNumber: 6CU4100JM1
UUID     : 30313436-3631-4336-5534-3130304A4D31
```

To monitor the operation of the `Find-HPiLO` cmdlet, use the `Verbose` parameter. The `Timeout` parameter default is 25 seconds (s). If the timeout value is not long enough for iLOs to respond, try using a `Timeout` parameter with a larger value.

In the preceding two commands, no double quotes are required around the `Range` parameter. If a comma is included in the range, double quotes are required. PowerShell interprets a comma as a list separator. Without double quotes, part of what should be a string is interpreted by PowerShell as a number. The operation of combined ranges is defined as creating a combination of each subnet address with each other subnet.

The following are examples of input range parameters using double quotes.

Range Parameter	Description
“192.168.1.1,15”	Specifies two addresses to check: 192.168.1.1 and 192.168.1.15.
“192.168.217,216.93,103”	Specifies four addresses to check: 192.168.217.93, 192.168.217.103, 192.168.216.93, 192.168.216.103.
“192.168.217,216.93-103”	Specifies 22 addresses to check: 192.168.217.93 through 192.168.217.103 and 192.168.216.93 through 192.168.216.103.

Using pipeline

Piping output from one command to another

In PowerShell, you can pipe output from one command to another. The preceding section provided examples of using `Find-HPiLO` to locate iLO devices. Instead of having to enter the iLO device information again, you could pipe the iLO device information to other cmdlets. You could also save the iLO device information and reuse it later.

The following is a script that pipes output from `Find-HPiLO` through `Add-Member` to add two required fields, and then to `Get-HPiLOFirmwareVersion` to produce the firmware version information for the iLO devices found. The `–Verbose` parameter is used to view more information.

PowerShell script:

```
Find-HPiLO 192.168.217.97-103 -Verbose |
% {Add-Member -PassThru -InputObject $_ Username admin}|
% {Add-Member -PassThru -InputObject $_ Password admin123}|
Get-HPiLOFirmwareVersion -Verbose
```

Script output:

The following is typical output from this script.

```
Warning  : It might take a while to search for all the iLOs if the
           input is a very large range or iLO response is delayed.
VERBOSE: Using 7 threads for search.
VERBOSE: No iLO at 192.168.217.97
VERBOSE: No system responds at 192.168.217.99
VERBOSE: No system responds at 192.168.217.100
VERBOSE: No system responds at 192.168.217.101
VERBOSE: No iLO at 192.168.217.102
VERBOSE: Using 2 threads.
VERBOSE: Sending to 192.168.217.98 - ilo2m203100ld.company.net
VERBOSE: Sending to 192.168.217.103 - iloromqap8207bc.company.net
VERBOSE: Errors-0, Warnings-0, Total-2


IP                   : 192.168.217.98
```

```
HOSTNAME             : ilo2m203100ld.company.net
STATUS_TYPE          : OK
STATUS_MESSAGE       : OK
FIRMWARE_DATE        : Jan 24 2013
FIRMWARE_VERSION     : 1.55
LICENSE_TYPE         : iLO 3 Advanced
MANAGEMENT_PROCESSOR : iLO3

IP                   : 192.168.217.103
HOSTNAME             : iloromqap8207bc.company.net
STATUS_TYPE          : OK
STATUS_MESSAGE       : OK
FIRMWARE_DATE        : Nov 05 2013
FIRMWARE_VERSION     : 1.32
LICENSE_TYPE         : iLO 4 Advanced
MANAGEMENT_PROCESSOR : iLO4
```

The verbose output indicates that seven threads are being used for `Find-HPiLO`. The output lists each address being checked, and then two threads in the `Get-HPiLOFirmwareVersion` command. (This threading allows multiple commands to multiple iLO devices to be sent at the same time.) The pipeline then sends to `Add-Member` twice. For each item, the pipeline adds the `-Username` and `-Password` parameters to the returned objects that represent the iLO devices found.

Those are in turn passed through to `Get-HPiLOFirmwareVersion` to drive its parameter inputs. The final results are the firmware version information for the two iLO devices found by `Find-HPiLO`. Without the `-Verbose` parameters, you would see the warning line from `Find-HPiLO` and the firmware information for the two iLO devices found in that range of addresses.

Piping parameters to iLO cmdlets

Piping a list of servers to an iLO cmdlet

You can pipe a range of servers or an array of servers to a cmdlet with the other needed parameters on the cmdlet. The following script pipes a range of servers to iLO cmdlets and uses parameters for `username` and `password`.

PowerShell script

```
PS C:\Windows> "192.168.243.181-182" | Get-HPiLOFIPSStatus -username
“username”
   -password “password”
```

Script output

```
IP             : 192.168.243.181
HOSTNAME       : bl460g8-2-ilo.company.net
STATUS_TYPE    : OK
STATUS_MESSAGE : OK
FIPS_MODE      : Disabled

IP             : 192.168.243.182
HOSTNAME       : bl620g7-ilo.company.net
STATUS_TYPE    : OK
STATUS_MESSAGE : OK
FIPS_MODE      : Disabled
```

You can also use an array of servers as input as shown in the following command.

```
PS C:\Windows> @("192.168.243.181”,”192.168.243.182") | Get-HPiLOFIPSStatus
   -username “username” -password “password”
```

This command produces the same output as the previous example.

Piping an object or a list of objects to iLO cmdlets

The following script pipes one object with needed properties to iLO cmdlet.

PowerShell script

```
PS C:\Windows> $inputObject = New-Object -TypeName PSObject -Property
@{ "Server"="192.168.243.180";
   "Username"="username"; "Password"="password"; "ServerName="newname" }
PS C:\Windows> $inputObject | Set-HPiLOServerName
```

Script output

If no errors occur when executing the script, no output is produced because this is a Set-cmdlet. The server name will be updated for 192.168.243.180. If an error occurs, the corresponding error message is displayed.

The following script pipes a list of objects with needed properties to iLO cmdlets.

PowerShell script

```
PS C:\Windows> $object1 = New-Object -TypeName PSObject -Property
@{ "Server"="192.168.243.180";
   "Username"="username"; "Password"="password";  "ServerName="newname1" }
PS C:\Windows> $object2 = New-Object -TypeName PSObject -Property
@{ "Server”=”192.168.243.181";
   "Username"="username"; "Password"="password";  "ServerName="newname2" }
PS C:\Windows> $objarray=@($object1, $object2)
PS C:\Windows> $objarray | Set-HPiLOServerName
```

Script output

If no errors occur when executing the script, no output is produced because this is a Set-cmdlet. The server name will be updated for 192.168.243.180 and 192.168.243.181. If an error occurs, the corresponding error message is displayed.

Using the `Get-HPiLOModuleVersion` and `Update-HPiLOModuleVersion` cmdlets

These cmdlets determine the current version of the Scripting Tools for Windows PowerShell: ILO Cmdlets installed and update the module if necessary.

The `Get-HPiLOModuleVersion` 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-HPiLOModuleVersion` cmdlet output.

```
PS C:\WINDOWS\system32> Get-HPiLOModuleVersion
```

```
Name                    : HPiLOCmdlets
Path                    : C:\Program Files\Hewlett-Packard\PowerShell\Modules
\HPiLOCmdlets\HPiLOCmdlets.psm1
Description             : Scripting Tools for Windows PowerShell : iLO
Cmdlets use the RIBCL interface to
                          communicate to iLO. These cmdlets can be used to
configure and manage iLO on
                          HPE ProLiant Gen7, Gen8 or Gen9 servers.
GUID                    : 05545ade-5f25-4696-bfcc-e1d67fe32519
Version                 : 1.5.0.0
CurrentUICultureName    : en-US
CurrentUICultureVersion : 1.5.0.0
AvailableUICulture      : {@{UICultureName=en-US; UICultureVersion=1.5.0.0},
@{UICultureName=zh-CN;
                          UICultureVersion=1.5.0.0}, @{UICultureName=ja-JP;
UICultureVersion=1.5.0.0}}
```

The `Update-HPiLOModuleVersion` cmdlet has no parameters. It checks the version number of the installed cmdlets against the version number available for download. If the local version is the most recent, it is indicated in the output.

```
PS C:\Users\Username> Update-HPiLOModuleVersion
The currently installed version 1.5.0.0 is the most current.
```

If there is a more recent version available than the one installed locally, you will be given the option to download the latest version.

```
PS C:\Users\Username> Update-HPiLOModuleVersion
There is a newer version of HPiLOCmdlets available at
http://www.hpe.com/info/powershell.
Do you want to go there to download the new version?(Y/N): Y
```

If you respond Yes to the download prompt, a new browser window opens, from which you can download and install the newer version.

Authentication parameters for iLO

There are two ways to authenticate user access to iLO for each cmdlet. One is to use the `Username` and `Password` parameters, the other is to use the `Credential` parameter.

Authenticating using Username and Password parameters

NOTE:

If using Active Directory or LDAP, use the correct form of the credentials. For example, `username@us.xxxx-dns.com`.

NOTE:

Both Single quote (') and double quote (") characters in combination are not supported as a parameter value in any Set cmdlet.

For example:

```
Set-HPiLOAssetTag -Server 192.168.1.1 –Username "username" –Password
"password" –AssetTag "asset'"tag"
```

The following script uses the `Username` and `Password` parameters for Authentication.

PowerShell script

```
PS C:\Users\Administrator> Get-HPiLOAssetTag -Server 192.168.1.1 –Username
"username"
     –Password "password"
```

Script output

```
IP             : 192.168.1.1
HOSTNAME       : ilohostbc.company.net
STATUS_TYPE    : OK
STATUS_MESSAGE : OK
ASSET_TAG      : assettag
```

Authenticating using the `Credential` parameter

The following script uses the `Credential` parameter for Authentication.

PowerShell script

```
PS C:\Users\Administrator> $getCrObj = Get-Credential -Message
     "Please input username and password"
PS C:\Users\Administrator> Get-HPiLOAssetTag -Server 192.168.1.1 -Credential
$getCrObj
```

Script output

```
IP             : 192.168.1.1
HOSTNAME       : ilohostbc.company.net
STATUS_TYPE    : OK
STATUS_MESSAGE : OK
ASSET_TAG      : assettag
```

The following script creates a new object type `PSCredential` for Authentication.

PowerShell script

```
PS C:\Users\Administrator> $crObj = New-Object
system.Management.Automation.PSCredential("username",(ConvertTo-SecureString
     "password" -AsPlainText -Force))
PS C:\Users\Administrator> Get-HPiLOAssetTag -Server 192.168.1.1 -Credential
$crObj
```

Script output

```
IP             : 192.168.1.1
HOSTNAME       : ilohostbc.company.net
STATUS_TYPE    : OK
STATUS_MESSAGE : OK
ASSET_TAG      : assettag
```

The following script creates a new object for different iLO server types using `PSCredential` as an array with `Credential` objects.

PowerShell script

```
PS C:\Users\Administrator>$crObj = New-Object
system.Management.Automation.PSCredential("username",(ConvertTo-SecureString
"password" -AsPlainText -Force))

PS C:\Users\Administrator>$crObj1 = New-Object
system.Management.Automation.PSCredential("username1",(ConvertTo-SecureString
"password1" -AsPlainText -Force))

PS C:\Users\Administrator> Get-HPiLOAssetTag -Server 192.168.1.1,192.168.1.2
-Credential $crObj,$crObj1
```

Script output

```
IP             : 192.168.1.1
HOSTNAME       : ilohostbc.company.net
STATUS_TYPE    : OK
STATUS_MESSAGE : OK
ASSET_TAG      : assettag


IP             : 192.168.1.2
HOSTNAME       : ilohostbc.company.net
STATUS_TYPE    : OK
STATUS_MESSAGE : OK
ASSET_TAG      : assettag
```

Server certificate authentication

All cmdlets now use server certificate authentication by default. If the iLO server does not have a valid server certificate installed, iLO cmdlet execution will fail.

The following example illustrates typical output for a situation where there is no valid certificate on the iLO server.

PowerShell script

```
PS C:\Program Files\HEWLETT-PACKARD\PowerShell\Modules\HPiLOCmdlets> Get-
HPiLOServerName
     -Server 192.168.1.1 -Username admin -Password admin

WARNING: DNS name translation not available for 192.168.1.1 - Host name left
blank.
```

Script output

```
Error - 192.168.1.1 -  - Error retrieving information from iLO
Server Certificate is not valid. Please check help of the cmdlet for more
details
or -DisableCertificateAuthenticaiton switch parameter to bypass the
certificate check
Exception calling "UploadString" with "3" argument(s): "The underlying
connection was closed:
Could not establish trust relationship for the SSL/TLS secure channel."
At C:\Program Files\HEWLETT-PACKARD\PowerShell\Modules\HPiLOCmdlets
\HPiLOCmdlets.psm1:17983 char:25
+                         throw $retobject.err
+                         ~~~~~~~~~~~~~~~~~~~
+CategoryInfo          :OperationStopped: (Exception calli...cure
channel.":String) [], RuntimeException
+FullyQualifiedErrorId :Exception calling "UploadString" with "3"
argument(s):
"The underlying connection was closed: Could not establish trust
relationship for the SSL/TLS secure channel."
```

Certificate checking can be disabled using the `-DisableCertificateAuthentication` parameter. If this parameter is set, the certificate on the iLO server is not checked before executing the cmdlet.

NOTE:

Disable certificate authentication until signed certificates are installed on iLO. If certificate checking is disabled, errors that would normally occur when iLO has an unsigned certificate installed are ignored. For more information, see the help for the `Disable-HPiLOCertificateAuthentication` and `Enable-HPiLOCertificateAuthentication` cmdlets, and the `-DisableCertificateAuthentication` parameter.

The following example uses `Disable-HPiLOCertificateAuthentication` to disable certificate checking.

PowerShell script

```
PS C:\Program Files\HEWLETT-PACKARD\PowerShell\Modules\HPiLOCmdlets> Get-
HPiLOServerName
     -Server 192.168.1.1 -Username admin -Password admin -
DisableCertificateAuthentication
WARNING: DNS name translation not available for 192.168.1.1 - Host name left
blank.
```

Script output

```
IP               : 192.168.1.1
HOSTNAME         :
STATUS_TYPE      : OK
STATUS_MESSAGE   : OK
SERVER_NAME      : WINDOWS-PEIORBB
SERVER_OSNAME    :
SERVER_OSVERSION :
```

Using the `Update-HPiLOServerFirmware` cmdlet

The `Update-HPiLOServerFirmware` cmdlet is used to update the complex programmable logic device (CPLD), Power PIC, and ROM firmware image of the server. To update the firmware, perform the following steps:

Procedure

1. Locate and download the server firmware package from **http://www.hpe.com/info/ilo**.
2. Execute the downloaded firmware package `CPxxxxxx.exe` and extract the package to a local folder.
3. Execute `Update-HPiLOServerFirmware` with the `Location` parameter set to the full path of the image that was extracted from the download.
4. Supported image extensions are the following:

Server Component	Format Type
ROM	.full or .flash
CPLD	.vme
Power PIC	.hex

5. Reboot the server for changes to take effect.

The command will be similar to the following:

```
PS C:\> Update-HPiLOServerFirmware -Server ilomx2232004p.company.net -
Username "username"
-Password "password" -Location C:\U15_2.30_08_06_2016.signed.flash

IP               : 1.4.217.187
HOSTNAME         : iLOwqc.domain.com
STATUS_TYPE      : WARNING
STATUS_MESSAGE   : Server will need to be reset for changes to be applied.
```

If the firmware is updated successfully, no other message is displayed. If an error occurs, an output message similar to the following is displayed.

```
PS C:\> Update-HPiLOServerFirmware -Server ilomx2232004p.company.net -
Username “username”
-Password "password" -Location “C:\U15_2.30_08_06_2016.signed.flash”

IP             HOSTNAME                   STATUS_TYPE        STATUS_MESSAGE
--             --------                  -----------          --------------
192.168.1.1    ilomx2232004p.company.net    ERROR            Firmware flash
failed.
```

NOTE:

Updating the firmware version should take no more than 5 minutes.

Using the `Update-HPiLOFirmware` cmdlet

The `Update-HPiLOFirmware` cmdlet is used to update a firmware image on iLO. To update the firmware, perform the following steps:

Procedure

1. Locate and download the iLO firmware package from the following website: **http://www.hpe.com/info/ilo**
2. Execute the downloaded firmware package `CPxxxxxx.exe` and extract the package to a local folder.
3. Execute `Update-HPiLOFirmware` with the `Location` parameter set to the full path of the bin image that was extracted from the download.

The command will be similar to the following:

```
PS C:\> Update-HPiLOFirmware -Server ilomx2232004p.company.net -Username
"username"
        -Password "password" -Location C:\ilo3_165.bin
```

If the firmware is updated successfully, no other message is displayed. If an error occurs, an output message similar to the following is displayed.

```
PS C:\> Update-HPiLOFirmware -Server ilomx2232004p.company.net -Username
"username"
        -Password "password" -Location "C:\iLOFW.bin"

IP             HOSTNAME                    STATUS_TYPE        STATUS_MESSAGE
--             --------                   -----------         --------------
192.168.1.1    ilomx2232004p.company.net    ERROR             Firmware flash
failed.
```

NOTE:

Updating the firmware version should take no more than five minutes.

Using the `Invoke-HPiLORIBCLCommand` cmdlet

The `Invoke-HPiLORIBCLCommand` cmdlet takes any RIBCL command as input, processes it, sends it to the iLO server, and returns the RIBCL response as PSObject. It takes all `READ` and `WRITE` operations of RIBCL, except `UPDATE`; sends them to the iLO server, using the input parameters mentioned, and returns the RIBCL output in PSObject format.

NOTE:

This cmdlet will not support any of the entity characters (<, >, &, ", ') in the `RIBCLCommand`, `Username`, and `Password` parameter fields.

For example, consider a sample RIBCL xml which is inputted to the current cmdlet as a string. The cmdlet then executes the sample xml on an iLO server and returns the RIBCL response as PSObject.

The RIBCL command is passed as a string to the cmdlet through the parameter `RIBCLCommand`. The RIBCL command can be read from an XML file, or can be directly passed as a string, as shown in the following example.

```
$xml= ([string](get-content "C:\Get_Firmware_Versions.xml")

Or

$xml=@"
@"
 <RIBCL VERSION="2.0">
   <LOGIN USER_LOGIN="admin" PASSWORD="admin123">
   <RIB_INFO MODE="read">
   <GET_FW_VERSION/>
   <GET_ALL_LICENSES/>
   </RIB_INFO>
   <SERVER_INFO MODE="READ">
   <GET_SERVER_NAME />
   </SERVER_INFO>
   </LOGIN>
 </RIBCL>

"@
```

Several RIBCL commands that perform read operations, or commands that perform write operations, can be combined and inputted as shown in the previous example. In the `$xml` variable, the RIBCL commands `<GET_FW_VERSION/>`, `<GET_ALL_LICENSES/>`, and `<GET_SERVER_NAME/>` are provided together in their respective tags to retrieve the firmware, license, and server information.

```
Invoke-HPiLORIBCLCommand -server "server" -Username "Username" -Password
"password" -RIBCLCommand $xml

IP                   : 15.212.141.162
HOSTNAME             :
STATUS_TYPE          : WARNING
STATUS_MESSAGE       : The system needs to be reset for changes to take
effect.
LICENSE              : @{LICENSE_CLASS=BETA; LICENSE_INSTALL_DATE=Sun Oct  2
05:41:09
                       2016; LICENSE_KEY=35DRH-WY2QD-CGXV6-XV6WJ-XMQ3R;
                       LICENSE_TYPE=iLO Advanced Limited NFR}
FIRMWARE_DATE        : Aug 19 2016
FIRMWARE_VERSION     : 2.50
LICENSE_TYPE         : iLO Advanced Limited NFR
MANAGEMENT_PROCESSOR : iLO4
SERVER_NAME          : WIN-PGFR400K46K
SERVER_OSNAME        :
SERVER_OSVERSION     :
```

Using iLO cmdlets on multiple targets

There are several ways to operate on multiple iLO servers with one command. A simple method to perform the same commands on multiple iLO servers is to use a CSV file containing some or all the

command parameters. A CSV file can be created using a Microsoft Excel spreadsheet saved in CSV format. The following examples demonstrate the use of an input CSV file.

NOTE:

Both single quote (') and double quote (") characters in combination are not supported as a parameter value in any Set cmdlet.

For example:

```
Set-HPiLOAssetTag -Server 192.168.1.1 –Username "username" –Password
"password" –AssetTag "asset'"tag"
```

The following examples use the `Set-HPiLOHostPower` and `Get-HPiLOHostPower` cmdlets. They operate on only two iLO servers, but more iLO servers could be included. The cmdlet parameter values are read from a CSV file with the following headings:

- **`Server`**—IP address or Hostname of the iLO
- **`Username`** and **`Password`**—credentials to log in to the iLO.
- **`HostPower`**—value which can be Yes to power on the server, or No to power off the server.[1]

When a CSV file is imported into PowerShell, it creates an object array that has elements with member name properties set to the first row names. Each element of the array is set to each line of the spreadsheet.

CSV input file

```
Input1.csv:
Server,Username,Password,HostPower
192.168.1.1,admin,admin123,Yes
192.168.1.3,admin,admin123,Yes
```

If the input CSV file has `Server`, `Username`, `Password`, and `HostPower` values, the following PowerShell script could be used.

PowerShell script

```
$path = ".\input1.csv"
$csv = Import-Csv $path
$rt = Set-HPiLOHostPower -Server $csv.Server -Username $csv.Username `
    -Password $csv.Password -HostPower $csv.HostPower
$rt | Format-List
$rt = Get-HPiLOHostPower -Server $csv.Server -Username $csv.Username `
    -Password $csv.Password
$rt | Format-List
```

The preceding example imports the CSV file into `$csv` and then uses the multivalued parameters to operate on multiple iLO servers in a single command. The script then gets the current power setting using

[1] Whether a server turns off or not when using `Set-HPiLOHostPower` depends on the operating system power button setting and state of the system. The operating system may ignore a power off request using this command. To force power off, use the `Set-HPiLOVirtualPowerButton` cmdlet with parameter `–PressType HOLD`.

the same `$csv` and lists the results. If both servers were already powered on, the following output would be displayed.

Script output

```
IP             : 192.168.1.1
HOSTNAME       : ilohostbc.company.net
STATUS_TYPE    : WARNING
STATUS_MESSAGE : Host power is already ON.

IP             : 192.168.1.3
HOSTNAME       : isabella-vp2.company.net
STATUS_TYPE    : WARNING
STATUS_MESSAGE : Host power is already ON.



IP             : 192.168.1.1
HOSTNAME       : ilohostbc.company.net
STATUS_TYPE    : OK
STATUS_MESSAGE : OK
HOST_POWER     : ON

IP             : 192.168.1.3
HOSTNAME       : isabella-vp2.company.net
STATUS_TYPE    : OK
STATUS_MESSAGE : OK
HOST_POWER     : ON
```

If no errors or warnings were returned from the set cmdlet, only the output from the `get` cmdlet would be displayed. `Set` cmdlets return nothing unless there is an error or warning returned from iLO.

Alternatively, if a single user name, password, and power setting apply to all iLO servers, just the iLO IP address or host name could be stored in the CSV file.

CSV input file

```
Input2.csv:
Server
192.168.1.1
192.168.1.3
```

If the input CSV file has only iLO IP or hostname, there is a common username and password for logging in, and all the servers have to be switched on, the following script could be used.

PowerShell script

```
$path = ".\input2.csv"
$csv = Import-Csv $path
$rt = Set-HPiLOHostPower -Server $csv.Server -Username "admin" `
    -Password "admin123" -HostPower "Yes"
$rt | Format-List
$rt = Get-HPiLOHostPower -Server $csv.Server -Username "admin" `
    -Password "admin123"
$rt | Format-List
```

The preceding example imports into `$csv` and then uses the multiple server array to power on all servers included in the CSV file. The same username, password, and power setting are used for both iLO servers. The output is the same as the previous command.

A similar command to power off could also be entered at the PowerShell prompt as follows:

```
PS C:\Users\yourname> Set-HPiLOHostPower -Server @("192.168.1.1","
192.168.1.3")
   -Username "admin" -Password "admin123" -HostPower No
```

If different usernames and passwords are used, the following command could be used:

```
PS C:\Users\yourname> Set-HPiLOHostPower -Server @("192.168.1.1","
192.168.1.3")
   -Username @("admin1","admin2") -Password @("password111","password222")
   -HostPower No
```

You could also use the imported server list and have the cmdlet prompt for the necessary parameters as shown in the following script.

PowerShell script

```
$path = ".\input2.csv"
$csv = Import-Csv $path
$rt = Set-HPiLOHostPower -Server $csv.Server
$rt | Format-List
$rt = Get-HPiLOHostPower -Server $csv.Server -Username "admin" -Password
"admin123"
$rt | Format-List
```

Script output

```
Username is not provided for the following iLO Server(s):
192.168.1.1
192.168.1.3
Use same Username for these servers? (Y/N) : Y
Please enter Username: admin
Password is not provided for the following iLO Server(s):
192.168.1.1
192.168.1.3
Use same Password for these servers? (Y/N) : Y
Please enter Password: *******
HostPower is not provided for the following iLO Server(s):
192.168.1.1
192.168.1.3
Use same HostPower for these servers? (Y/N) : Y
Please enter HostPower: Yes

IP             : 192.168.1.1
HOSTNAME       : ilohostbc.company.net
STATUS_TYPE    : WARNING
STATUS_MESSAGE : Host power is already ON.

IP             : 192.168.1.3
```

```
HOSTNAME       : isabella-vp2.company.net
STATUS_TYPE    : WARNING
STATUS_MESSAGE : Host power is already ON.


IP             : 192.168.1.1
HOSTNAME       : ilohostbc.company.net
STATUS_TYPE    : OK
STATUS_MESSAGE : OK
HOST_POWER     : ON

IP             : 192.168.1.3
HOSTNAME       : isabella-vp2.company.net
STATUS_TYPE    : OK
STATUS_MESSAGE : OK
HOST_POWER     : ON
```

Because the items in the CSV file are named the same as the parameters, you can pipe the imported object into the cmdlet and get the same results again as shown in the following script.

PowerShell script

```
$path = ".\input1.csv"
$csv = Import-Csv $path
$rt = $csv | Set-HPiLOHostPower
$rt | Format-List
$rt = $csv | Get-HPiLOHostPower
$rt | Format-List
```

As demonstrated by the preceding examples, the methods available for providing input to a cmdlet are flexible. These same techniques can be used on most iLO cmdlets.

Interoperation of iLO and OA cmdlets

NOTE:

The `Invoke-HPOAiLOCommand` cmdlet used to support interoperation requires iLO cmdlets 1.2 or later. Attempting to use this cmdlet with an earlier version of the iLO cmdlets will cause an error.

In C7000 and C3000 blade enclosures it is possible to configure the internal network to either expose the iLO connections to an external network, or to keep them accessible only through the OA. This leaves a gap for the iLO cmdlets (for iLO) in being able to operate on iLOs only accessible within an OA network.

Beginning with version 1.2 of the iLO cmdlets and version 1.1 of the OA cmdlets (for Onboard Administrator), it is possible to send commands created by the iLO cmdlets to `Invoke-HPOAiLOCommand`, which in turn sends it to one or more iLOs within the OA network and returns the results, much the same as if the iLO cmdlet was executed directly on an iLO.

Requirements

Interoperation requires iLO cmdlets version 1.2 or later and OA cmdlets version 1.1 or later. See **Installation** on page 6 for other requirements. An attempt to run the OA cmdlet without the iLO cmdlets being installed or the wrong version will cause an error.

Changes to iLO cmdlets

Beginning with the iLO cmdlets version 1.2, cmdlets for which this applies have been modified to support an additional value of `ExternalCommand` for the `–OutputType` parameter. This causes the cmdlets to only generate the RIBCL command as output, instead of sending it to an actual iLO. The parameters `Server`, `Username`, and `Password` must be assigned a value, such as an OA IP address, username, and password. Validation of the parameters is not performed, but these parameters are required. Other parameters need to be specified just like you would in sending them to an iLO or group of iLOs.

Using the `Invoke-HPOAiLOCommand` cmdlet

The `Invoke-HPOAiLOCommand` cmdlet is used to send the RIBCL generated by iLO cmdlets to iLOs inside the OA network. This cmdlet has four parameters:

- **iLOCommand**

  : String, RIBCL Text generated by an iLO command only, (no external files or sources of RIBCL are supported). This is a mandatory parameter.

- **Bay**

  : Integer indicating Server Bay # or All for all server bays. This is a mandatory parameter.

- **Connection**

  : SSH Connection to the OA. This is a mandatory parameter.

- **OutputType**

  : String, either PSObject or RawText (defaults to PSObject)

This cmdlet is called after the iLO RIBCL command is generated by the iLO cmdlets using the `ExternalCommand OutputType`. The output from the iLO cmdlet (RIBCL) is passed to `Invoke-HPOAiLOCommand` in the `iLOCommand` parameter. This cmdlet sends the RIBCL command to the OAs indicated by the `Connection` parameter. Each OA passes it through to the iLOs indicated by the `Bay` parameter. The results are returned from this cmdlet as indicated by the `–OutputType` parameter. PSObject is the default type and RawText will return the unconverted RIBCL response from iLO.

Usage example

The following example illustrates getting an iLO data item from OA. This script gets the default language of the iLO in Bay 2 of the enclosure.

PowerShell script

```
#create an OA connection
$c = Connect-HPOA -OA 192.168.242.62 -username “username” -password
“password”
#get the RIBCL to send
$d = Get-HPiLODefaultLanguage -Server "iLOipaddress" -Username “username”
     -Password “password” –OutputType ExternalCommand
#use the connection to OA and the RIBCL command and send it to Bay 2
$c | Invoke-HPOAiLOCommand –iLOCommand $d –Bay 2
```

Even though the iLO cmdlet has a dummy server IP address, username and password as the OA, these parameters are not used for anything and are just place holders. However, they must be included.

Script output

```
BAY             : 2
IP              : 192.168.242.62
HOSTNAME        : westwind.hpe.com
```

```
STATUS_TYPE    : OK
STATUS_MESSAGE : OK
LANG_ID        : en
LANGUAGE       : English
```

Log processing examples

The following examples demonstrate how to access iLO log data. They illustrate how to obtain a summary of events within the iLO logs without having to view all the log events. The summary enables you to focus on the details for specific types of events in the logs.

The following input CSV file and script create an event summary for two iLOs.

CSV input file:

```
Input3.csv:
Server,Username,Password
192.168.1.9,admin,admin123
192.168.1.14,admin,admin123
```

PowerShell script:

```
$path = ".\input3.csv"
$csv = Import-Csv $path
$rt = $csv | Get-HPiLOEventLog
#process the ilo event log returned from each iLO
foreach ($ilo in $rt) {
    $ilo.IP + " has " + $ilo.EVENT.Count + " iLO log entries."
    $sevs = $(foreach ($event in $ilo.EVENT) {$event.SEVERITY})
    $uniqsev = $($sevs | Sort-Object | Get-Unique)
    $sevcnts =  $ilo.EVENT | group-object -property SEVERITY –noelement
    "There are " + $uniqsev.Count + " type(s) of events in the iLO log.”
    $sevcnts | Format-Table
}
```

The IP addresses, including the user names and passwords, for the iLOs that you want to view are imported from the CSV file. For each iLO included, you can identify the types of entries and the count of each. This preceding script works for many entries in the input CSV file.

Script output:

```
192.168.1.9 has 93 iLO log entries.
There are 2 type(s) of events in the iLO log.
Count Name
----- ----
  90  Informational
   3  Caution


192.168.1.14 has 255 iLO log entries.
There are 2 type(s) of events in the iLO log.
Count Name
----- ----
  221 Informational
```

```
   34 Caution
```

From this output you can see that there are many Informational messages that you might want to ignore. However, you might want to view the Caution messages. There are no Critical messages.

The preceding script can be modified to view the Integrated Management Log (IML). This can easily be done with one code change and a few message changes. The following is the modified script.

PowerShell script:

```
$path = ".\input3.csv"
$csv = Import-Csv $path
$rt = $csv | Get-HPiLOIML
#process the system IML returned from each iLO
foreach ($ilo in $rt) {
    $ilo.IP + " has " + $ilo.EVENT.Count + " IML entries."
    $sevs = $(foreach ($event in $ilo.EVENT) {$event.SEVERITY})
    $uniqsev = $($sevs | Sort-Object | Get-Unique)
    $sevcnts =  $ilo.EVENT | group-object -property SEVERITY –noelement
    "There are " + $uniqsev.Count + " type(s) of events in the IML.”
    $sevcnts | Format-Table
}
```

Script output:

```
192.168.1.9 has 3 IML entries.
There are 1 type(s) of events in the IML.
Count Name
----- ----
   3  Informational


192.168.1.14 has 84 IML entries.
There are 3 type(s) of events in the IML.
Count Name
----- ----
  22  Informational
  19  Repaired
  43  Caution
```

Return objects and error handling

The iLO cmdlets return PowerShell custom objects (`PSObject`) as the default.[2] Values for the returned objects can be accessed and used as any other objects in PowerShell. If you have `$rt` set to the returned object from iLO, it should contain either `$null` (no value is returned) or contain a returned object. For example, to access the `HOST_POWER` property in the returned object, use `$rt.HOST_POWER`.

As in the preceding examples, when there is an error or warning message returned from an iLO, it is indicated by a property in the returned object called `STATUS_TYPE`. Enclosing iLO cmdlets in `try` blocks and using `catch` for errors is a good practice, but it does not handle a returned iLO error or warning. Three values can be returned in `STATUS_TYPE`: OK, WARNING, or ERROR.

The following script modifies one of the preceding examples by adding error handling.

[2] XML and RIBCL output types can also be selected with the –OutputType parameter.

PowerShell script:

```
$path = ".\input1.csv"
$csv = Import-Csv $path
try {
    $rt = $csv | Set-HPiLOHostPower
    if ($rt -ne $null) {
        foreach ($iloreturn in $rt) {
          switch ($iloreturn.STATUS_TYPE) {
            #OK status is not returned in a Set cmdlet
            #but you can get a warning or error
            'WARNING' { "I have been warned by " + $iloreturn.IP +
                 " that: " + $iloreturn.STATUS_MESSAGE}
            'ERROR'   { "Somthing bad returned by " + $iloreturn.IP +
                 ": " + $iloreturn.STATUS_MESSAGE}
          }
      }
    }
    $rt = $csv | Get-HPiLOHostPower
    $rt | Format-List
}
catch {
#code for however you want to handle a PowerShell error in the try block
    exit
}
```

Script output:

```
I have been warned by 192.168.1.1 that: Host power is already ON.
I have been warned by 192.168.1.3 that: Host power is already ON.

IP             : 192.168.1.1
HOSTNAME       : ilohostbc.company.net
STATUS_TYPE    : OK
STATUS_MESSAGE : OK
HOST_POWER     : ON

IP             : 192.168.1.3
HOSTNAME       : isabella-vp2.company.net
STATUS_TYPE    : OK
STATUS_MESSAGE : OK
HOST_POWER     : ON
```

Because PowerShell errors print the error and continue, it might be sufficient to leave out the `try` – `catch` handling unless you want to exit, or perform some other handling such as logging the error.

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. You can test code and view objects to get a better understanding how to accomplish what you want to do.

NOTE:

A collection of sample scripts are provided with Scripting Tools for Windows PowerShell: ILO Cmdlets v1.2 and later. For more information, see **Sample scripts** on page 41.

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 command interactively to retrieve the data.
3. After viewing the command results, decide what part of the object you are interested in.
4. Determine iLO servers or other sources of information that will drive the process.
5. Create the main processing loop.
6. 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.

As demonstrated in the preceding examples, consider using .CSV files to drive input when there are multiple inputs to act on. It is also possible to use XML files and import data from a source that generates or maintains XML type data, such as a database. To get the same object from an XML file, you could create it by using the `Export-Clixml` command to see what it looks like. The same `input3.csv` data that is exported to an XML file looks like the following:

```
<Objs Version="1.1.0.1" xmlns="http://schemas.microsoft.com/powershell/
2004/04">
  <Obj RefId="0">
    <TN RefId="0">
      <T>System.Management.Automation.PSCustomObject</T>
      <T>System.Object</T>
    </TN>
    <MS>
      <S N="Server">192.168.1.9</S>
      <S N="Username">admin</S>
      <S N="Password">admin123</S>
    </MS>
  </Obj>
  <Obj RefId="1">
    <TNRef RefId="0" />
    <MS>
      <S N="Server">192.168.1.14</S>
      <S N="Username">admin</S>
      <S N="Password">admin123</S>
    </MS>
  </Obj>
</Objs>
```

Troubleshooting

General issues

Verifying iLO firmware versions

If a problem occurs, your first action should be to verify that the most current versions of iLO firmware are installed. Updating to the most current firmware might solve the problem. For information on updating iLO firmware, see the iLO 4 user guide.

To determine if there is a newer version of iLO cmdlets available, see **Using the Get-HPiLOModuleVersion and Update-HPiLOModuleVersion cmdlets** on page 22.

iLO protocols and ports

The following table identifies the protocols and ports for iLO:

Product	Protocol	Port
iLO	HTTP	80
	HTTPS	443

The iLO cmdlets do not work after enabling the "Enforce AES/3DES Encryption" setting in HPE iLO 3 or iLO 4

If the iLO cmdlets do not work after enabling the "Enforce AES/3DES Encryption" setting in iLO 3 or iLO 4, use the following tables to verify that your environment has the correct versions.

iLO cmdlets 1.5.0.0

iLO Firmware	.NET Framework version	Windows PowerShell version	Transport Layer Security (TLS) version
Any version of iLO 3 or iLO 4	4.5 or later	3.0, 4.0, 5.0, or 5.1	TLS 1.0, 1.1, or 1.2

iLO cmdlets 1.4.0.2

iLO Firmware	.NET Framework version	Windows PowerShell version	Transport Layer Security (TLS) version
Any version of iLO 3 or iLO 4	4.5 or later	3.0, 4.0, or 5.0	TLS 1.0, 1.1, or 1.2

iLO cmdlets 1.3.0.0, 1.4.0.0, or 1.4.0.1

iLO Firmware	.NET Framework version	Windows PowerShell version	Transport Layer Security (TLS) version
iLO 4 version 2.30, 2.40, or later	4.5.1 or later	4.0 or 5.0	TLS 1.2
iLO 4 version 2.20 or earlier	4.0 or later	3.0, 4.0, or 5.0	TLS 1.1
iLO 3 version 1.88 or later	4.0 or later	3.0, 4.0, or 5.0	TLS 1.1
iLO 3 version 1.87 or earlier	4.0 or later	3.0, 4.0, or 5.0	TLS 1.0

iLO cmdlets 1.2.0.3

iLO Firmware	.NET Framework version	Windows PowerShell version	Transport Layer Security (TLS) version
iLO 4 version 2.30, 2.40, or later	4.5.1 or later	4.0 or 5.0	TLS 1.2
iLO 4 version 2.20 or earlier	4.0 or later	3.0, 4.0, or 5.0	TLS 1.1
iLO 3 version 1.88 or later	4.0 or later	3.0, 4.0, or 5.0	TLS 1.0
iLO 3 version 1.87 or earlier	4.0 or later	3.0, 4.0, or 5.0	TLS 1.0

iLO cmdlets 1.2.0.2

iLO Firmware	.NET Framework version	Windows PowerShell version	Transport Layer Security (TLS) version
iLO 4 version 2.30, 2.40, or later	4.5.1 or later	4.0 or 5.0	TLS 1.0
iLO 4 version 2.20 or earlier	4.0 or later	3.0, 4.0, or 5.0	TLS 1.0
iLO 3 version 1.88 or later	4.0 or later	3.0, 4.0, or 5.0	TLS 1.0
iLO 3 version 1.87 or earlier	4.0 or later	3.0, 4.0, or 5.0	TLS 1.0

Find-HPiLO cmdlet response time is longer when client proxy server setting is invalid

If there is an invalid proxy server Internet setting on the client environment, the Find-HPiLO cmdlet response time is longer.

Usage tips

Sample scripts

In HPiLOCmdlets v1.2 and later, several sample scripts are included in a folder named `Samples`, which is located in the directory where HPiLOCmdlets install package was unzipped. The primary purpose of these sample scripts is to provide instructions on the following scripting aspects:

- Samples for inputting parameters for a cmdlet (such as named parameters, pipeline parameters, imported parameters from CSV, etc.)
- Samples for handling with errors or exceptions in the script.
- Samples for implementing scripts for complicated functions, which need several cmdlets. This includes generating and importing iLO certification, setting iLO up to use Active Directory authentication, etc.

Script examples

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

Feature not supported

If the return STATUS_MESSAGE of an iLO cmdlet is `Feature not supported – xxx`, it typically indicates the cmdlet is not supported on the current iLO version or server type. Check the cmdlet help (`Get-Help <cmdlet name> -Full`) or check the current iLO firmware version using `Get-HPiLOFirmwareVersion` or `Find-HPiLO`.

For more information, see the following website: **http://www.hpe.com/info/ilo**.

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.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:

  http://www.hpe.com/assistance

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

  http://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

  www.hpe.com/support/hpesc

  Hewlett Packard Enterprise Support Center: Software downloads

  www.hpe.com/support/downloads

  Software Depot

  www.hpe.com/support/softwaredepot

- To subscribe to eNewsletters and alerts:

  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:

  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.

Customer self repair

Hewlett Packard Enterprise customer self repair (CSR) programs allow you to repair your product. If a CSR part needs to be replaced, it will be shipped directly to you so that you can install it at your convenience. Some parts do not qualify for CSR. Your Hewlett Packard Enterprise authorized service provider will determine whether a repair can be accomplished by CSR.

For more information about CSR, contact your local service provider or go to the CSR website:

http://www.hpe.com/support/selfrepair

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

www.hpe.com/services/getconnected

HPE Proactive Care services

www.hpe.com/services/proactivecare

HPE Proactive Care service: Supported products list

www.hpe.com/services/proactivecaresupportedproducts

HPE Proactive Care advanced service: Supported products list

www.hpe.com/services/proactivecareadvancedsupportedproducts

Proactive Care customer information

Proactive Care central

www.hpe.com/services/proactivecarecentral

Proactive Care service activation

www.hpe.com/services/proactivecarecentralgetstarted

Warranty information

To view the warranty for your product or to view the *Safety and Compliance Information for Server, Storage, Power, Networking, and Rack Products* reference document, go to the Enterprise Safety and Compliance website:

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

Additional warranty information

HPE ProLiant and x86 Servers and Options

www.hpe.com/support/ProLiantServers-Warranties

HPE Enterprise Servers

www.hpe.com/support/EnterpriseServers-Warranties

HPE Storage Products

www.hpe.com/support/Storage-Warranties

HPE Networking Products

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:

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:

www.hpe.com/info/reach

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

www.hpe.com/info/ecodata

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

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.