HP Scripting Tools for Windows PowerShell User Guide

iLO Cmdlets

Abstract

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

HP Part Number: 745004-002a
Published: April 2014
Edition: 2

Contents

1 Introduction to HP Scripting Tools for Windows PowerShell

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

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

This guide is intended for system administrators who use the HP Scripting Tools for Windows PowerShell to manage their IT infrastructure. Users should be familiar with Windows PowerShell.

This guide describes the cmdlets included in version 1.1 of the HP iLO Cmdlets for Windows PowerShell. For information on the cmdlets added in version 1.1, see the *HP Scripting Tools for Windows PowerShell Release Notes iLO Cmdlets v1.1*.

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, HP Scripting Tools for Windows PowerShell is a powerful set of utilities that can be used to perform various configuration tasks on HP 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

HP iLO Cmdlets for Windows PowerShell provides the following features:

- Support for iLO 3 and iLO 4 configuration.
- Uses proven PowerShell technology to provide consistent and reliable server configuration.
- Supports the standard PowerShell architecture and scripting model.
- Object oriented—output from one command can be piped to another command.
- Built-in online help for all HP PowerShell cmdlets, documenting syntax and usage examples.

2 Installation

System prerequisites

The following prerequisites must be met before installing HP iLO Cmdlets for Windows PowerShell. There are also several items to consider in the install.

- Microsoft Management Framework 3.0 or later (which includes PowerShell 3.0 or later) must be installed on your system before installing the HP Scripting Tools for Windows PowerShell. Microsoft .NET Framework 4 or later must be installed before installing Microsoft Management Framework 3.0 or later. The following links provide access to the Microsoft download sites for these applications.
    - Microsoft .NET Framework 4
    - Microsoft .NET Framework 4.5
    - Windows Management Framework 3.0
    - Windows Management Framework 4.0
- Make sure you read and understand the system requirements and other information provided on the above pages.

Supported operating systems

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

- Microsoft Windows 7 SP1
- Microsoft Windows 8
- Microsoft Windows 8.1
- Microsoft Windows Server 2008 R2 SP1
- Microsoft Windows Server 2012
- Microsoft Windows Server 2012 R2

Installing HP iLO Cmdlets for Windows PowerShell

The HP iLO Cmdlets for Windows PowerShell can be downloaded from the following website: http://www.hp.com/go/powershell.

Using the following guidelines when installing HP iLO Cmdlets 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. Click **Start** and select **All Programs**→**Accessories**→**Windows PowerShell**.
2. Right-click one of the PowerShell options and select **Run as administrator**.
3. Change (CD) to the directory where you unzipped the installer.
4. 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 4.0 or above
  - Attempting to install without PowerShell 3.0 or above

Uninstalling HP iLO Cmdlets for Windows PowerShell

To uninstall the HP iLO Cmdlets for Windows PowerShell:

1. Open Windows Control Panel.
2. Select **Programs and Features**.
3. Select **Hewlett-Packard iLO Cmdlets**.
4. Click **Uninstall**.

3 HP Scripting Tools for Windows PowerShell cmdlets

Table 1 provides a list and brief description of all the HPiLOCmdlets for Windows PowerShell.

Cmdlet Help

The HPiLOCmdlets are supported by help, which is used in the same way as other PowerShell cmdlets. To display a complete list of the HPiLOCmdlets in PowerShell, type:

```
help *hpilo*
```

NOTE: You can also use the following command to display the HP 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 HP iLO Cmdlet.

The HP iLO Cmdlets support the PowerShell `Update-Help` feature. When you execute this command, it accesses an HP website, gets the most current help file(s), and puts them in the correct location on your system.

Table 1 HP iLO Cmdlets for Windows PowerShell

Cmdlet	Description
Add-HPiLOSSORecord	Add a new HP SIM Single Sign-On (SSO) Server Record
Add-HPiLOUser	Adds a local user account to the iLO
Clear-HPiLOAHSData	Clear the AHS information from the AHS logs
Clear-HPiLOEventLog	Clear the iLO event logs
Clear-HPiLOIML	Clear the Integrated Management Logs
Clear-HPiLOPowerOnTime	Clears the virtual clock counter without power-cycling the server
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
Dismount-HPiLOVirtualMedia	Dismounts the Virtual Media image if one is mounted.
Enable-HPiLOFIPS	Enables the Federal Information Processing Standard *Enforce AES/3DES Encryption* setting
Enable-HPiLOERSIRSConnection	Connect to the Insight Remote Support server, and register the server
Enable-HPiLOSecurityMessage	Enables the security text message in the iLO Login Banner
Find-HPiLO	Find list of iLO in a specified subnet
Get-HPiLOAHSStatus	Gets the AHS status
Get-HPiLOAssetTag	Gets the Asset tag
Get-HPiLOCertificateSigningRequest	Gets the Certificate Signing Request status
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-HPiLOERSSetting	Gets the Integrated Remote Support setting

Table 1 HP iLO Cmdlets for Windows PowerShell *(continued)*

Cmdlet	Description
Get-HPiLOEventLog	Gets the iLO event logs
Get-HPiLOFan	Gets the Fan health details from the server
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 the iLO is located
Get-HPiLOModuleVersion	Gets the module details for the HPiLOCmdlets
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-HPiLOPersistentBootOrder	Gets the current boot order
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-HPiLOSecurityMessage	Gets the security message for the iLO login screen
Get-HPiLOServerName	Gets the host server name used by iLO

Table 1 HP iLO Cmdlets for Windows PowerShell *(continued)*

Cmdlet	Description
Get-HPiLOSNMPIMSetting	Gets the respective iLO SNMP IM settings
Get-HPiLOSpatial	Gets the location information and system data with HP 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-HPiLOTemperature	Gets the temperature health details of the server
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
Invoke-HPiLOProfileApply	Applies a deployment setting profile in iLO 4
Invoke-HPiLOProfileDownload	Modify a Profile Description, download a specific blob and write the blob to the blob store
Mount-HPiLOVirtualMedia	Mounts the specified media image
Remove-HPiLOProfile	Deletes a deployment profile
Remove-HPiLOSSORecord	Removes an HP SIM Trusted SSO Server record
Remove-HPiLOUser	Removes an existing local user account
Remove-HPiLOUserSSHKey	Deletes any SSH keys associated with a particular UserLogin
Reset-HPiLOAdministratorPassword	Resets the administrator password to a default value
Reset-HPiLORIB	Resets the iLO
Reset-HPiLOServer	Resets the host server on which the iLO is operating
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 computer's lock settings
Set-HPiLODefaultLanguage	Sets the default language on iLO
Set-HPiLODirectory	Modifies the directory settings on iLO
Set-HPiLOFactoryDefault	Sets the Integrated Lights-Out device to factory default settings
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 1 HP iLO Cmdlets for Windows PowerShell *(continued)*

Cmdlet	Description
Set-HPiLOKerberosConfig	Configures the Kerberos authentication
Set-HPiLOLicenseKey	Applies a license key for the Integrated Lights-Out Advanced Pack
Set-HPiLOLockConfiguration	Enable the data-center configuration lock for iLO
Set-HPiLONetworkSetting	Modifies the network settings of the host server where the iLO is located
Set-HPiLOOneTimeBootOrder	Set One Time Boot Order
Set-HPiLOPassword	Changes the password of a local user
Set-HPiLOPersistentBootOrder	Sets the Persistent boot order
Set-HPiLOPowerAlertThreshold	Set the power alert threshold value for the iLO
Set-HPiLOPowerCap	Set 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-HPiLOSharedNetworkPort	Configures Integrated Lights-Out device to pass network traffic on the shared host network port
Set-HPiLOSNMPIMSetting	Modifies the respective iLO SNMP IM settings
Set-HPiLOSSOSetting	Modifies the HP 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, cold boot and warm boot of the server
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 an L2 data collection submission to the Insight Remote Support server
Start-HPiLOTestEvent	Initiates a test service event submission to the Insight Remote Support server
Update-HPiLOFirmware	Updates the iLO firmware
Update-HPiLOModuleVersion	Checks if a newer version of the HPiLO Cmdlets is available on the HP 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.

Using the `Find-HPiLO` cmdlet

When learning about the HP iLO Cmdlets for Windows PowerShell, 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 datacenter, 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 a single IP address:

```
Find-HPiLO 192.168.1.1

Warning : It might take a while to search all the HP iLO servers if the input
is a very large range. Use Verbose for more information.

IP       : 192.168.1.1
SPN      : ProLiant ML310e Gen8
FWRI     : 1.30
PN       : Integrated Lights-Out 4 (iLO 4)
HOSTNAME : ilomx2232004p.company.net
```

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

```
Find-HPiLO 192.168.1.1-11

Warning : It might take a while to search all the HP iLO servers if the input
is a very large range. Use Verbose for more information.

IP       : 192.168.1.2
SPN      : ProLiant DL120 G7
FWRI     : 1.28
PN       : Integrated Lights-Out 3 (iLO 3)
HOSTNAME : ilohostv8.company.net

IP       : 192.168.1.4
SPN      : ProLiant MicroServer Gen8
FWRI     : 1.20
PN       : Integrated Lights-Out 4 (iLO 4)
HOSTNAME : fbtilodns.company.net

IP       : 192.168.1.10
SPN      : ProLiant DL360p Gen8
FWRI     : 1.40
PN       : Integrated Lights-Out 4 (iLO 4)
HOSTNAME : ilohostbc.company.net
```

To monitor the operation of the `Find-HPiLO` cmdlet, use the `Verbose` parameter. The `Timeout` parameter default is 300 milliseconds. If the timeout value is not long enough for iLOs to respond, try using a `Timeout` parameter with a larger value. A value of 1000 milliseconds should provide reliable operation over even the longest distance.

In the preceding two commands no double quotes are required around the `Range` parameter, but if a comma is included in the range, double quotes are required. This is because the use of a comma is interpreted as a list separator by PowerShell. 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 twenty-two addresses to check, 192.168.217.93 through 192.168.217.103 and 192.168.216.93 through 192.168.216.103.

Piping output from one command to another

A useful feature of PowerShell is the ability to pipe output from one command to another. The preceding section provided examples of using `Find-HPiLO` to locate iLO devices. You may want to use those with other commands rather than input the iLOs you find again or store them somewhere and re-use them.

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 iLOs 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 all the HP iLO servers if the input
is a very large range. Use Verbose for more information.
VERBOSE: Using 7 threads for search.
VERBOSE: Pinging 192.168.217.97
VERBOSE: Pinging 192.168.217.98
VERBOSE: Pinging 192.168.217.99
VERBOSE: Pinging 192.168.217.100
VERBOSE: Pinging 192.168.217.101
VERBOSE: Pinging 192.168.217.102
VERBOSE: Pinging 192.168.217.103
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 shown indicates that seven threads are being used for `Find-HPiLO`, which lists each address being checked, and then two threads in the `Get-HPiLOFirmwareVersion` command. (This threading allows multiple commands to multiple iLOs to be sent at the same time.) The pipeline then sends to `Add-Member` twice and for each item adds the `-Username` and `-Password` parameters to the returned objects that represent the iLOs 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 iLOs found by `Find-HPiLO`. Without the `-Verbose` parameters you would see the Warning line from `Find-HPiLO` and the firmware information for the two iLOs found in that range of addresses.

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

These cmdlets are used to determine the current version of the HPiLOCmdlets module installed and update the HPiLOCmdlets 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:\Users\Username> Get-HPiLOModuleVersion


Name                : HPiLOCmdlets
Path                : C:\Program Files\Hewlett-Packard\PowerShell\Modules\
                      HPiLOCmdlets\HPiLOCmdlets.psm1
Description         : Cmdlets to interface with HP iLO
GUID                : 05545ade-5f25-4696-bfcc-e1d67fe32519
Version             : 1.1.0.0
UICultureName       : en-US
UICultureVersion : 1.1.0.0
```

The `Update-HPiLOModuleVersion` cmdlet has no parameters. This cmdlet checks the version number of the installed cmdlets against the version number available for download. If the local version is the most recent, the output will indicate this.

```
PS C:\Users\Username> Update-HPiLOModuleVersion
The currently installed version 1.1.0.0 is the most current.
```

If there is a more recent version available than the one currently installed locally, the output will indicate this and give you option to download the latest version.

```
PS C:\Users\Username> Update-HPiLOModuleVersion
There is a newer version of HPiLOCmdlets available at
```

```
http://www.hp.com/go/powershell.
Do you want to go there to download the new version?(Y/N): Y
```

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

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:

1. Locate and download the iLO firmware package from the following website:

   http://www.hp.com/go/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 admin
        -Password "admin123" -Location C:\ilo3_165.bin
PS C:\>
```

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 admin
        -Password "admin123" -Location C:\aaaa.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 5 minutes.

Using iLO cmdlets on multiple targets

There are several ways to operate on multiple iLOs with one command. If there are many iLOs on which the same commands must be performed, a simple method is to use a CSV file containing some or all of 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.

The following examples use the `Set-HPiLOHostPower` and `Get-HPiLOHostPower` cmdlets. They operate on only two iLOs, but more iLOs 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 login to the iLO
- **HostPower**—value which can be Yes to power on the server, or No to power off the server[1]

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`.

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, and each element of the array 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 the 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 `$csv` and then uses the multi-valued parameters to operate on multiple iLOs in a single command. The script then gets the current power setting using the same `$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, just the iLO IP address or host name could be stored in the CSV file, if a single user name, password, and power setting apply to all iLOs.

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 and there is a common username and password for logging in, and if 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 iLOs. 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, then 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 needed parameters as 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 not provided...
Use same username for all servers (y/n) : y
Please enter Username: admin
Password not provided...
Use same Password for all servers (y/n) : y
Please enter password: ********
Hostpower not provided...
Use same Hostpower for all 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 very flexible. These same techniques can be used on most iLO cmdlets.

Log processing examples

The following examples demonstrate how to access iLO log data. In these examples, you want to get a summary of the events in 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.

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.

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

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, testing code and viewing objects to get a better understanding how to accomplish what you want to do.

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

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 iLOs 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 of 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 this:

```
<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>
```

4 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 *HP 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" (page 13).

5 Support and other resources

Contacting HP

For worldwide technical support information, see the HP support website:

http://www.hp.com/support

Before contacting HP, collect the following information:

- Error messages
- Operating system type and revision level
- Detailed questions

Support Information

HP offers a number of additional software support services, many of which are provided to our customers at no additional charge.

Subscription service

Receive, by email, support alerts announcing product support communications, driver updates, software releases, firmware updates, and customer-replaceable component information by signing up at http://www.hp.com/go/myadvisory.

To change options for support alerts you already receive, click the **Sign in** link on the right.

HP Support Center

Join the discussion at the HP Support Center, a community-based, user-supported tool for HP customers to participate in discussions amongst the customer community about HP products.

Contact support

HP Worldwide Customer Service contact numbers are available at http://www.hp.com/country/us/en/wwcontact.html.

Reporting errors to HP

If you get a PowerShell error that indicates that it is reporting something within the HPiLOCmdlet module code, please contact HP. Provide as much information as possible, including screen captures if appropriate. Also include the output of the following command:

```
PS C:\Users\yourname> Get-HPiLOModuleVersion
```

Related information

The following documents and websites provide related information:

Documents

- *HP Scripting Tools for Windows PowerShell Release Notes*
- *HP iLO 4 User Guide*

Websites

- HP Scripting Tools for Windows PowerShell: http://www.hp.com/go/powershell
- HP iLO Information Library: http://www.hp.com/go/ilo/docs

Windows PowerShell resources

The following websites provide useful information for using PowerShell.

- Microsoft Script Center
- Windows PowerShell Blog
- PowerShell.com
- PowerShell Community Groups
- PowerShell.org
- PowerShell Magazine

6 Documentation feedback

HP 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@hp.com). Include the document title and part number, version number, or the URL when submitting your feedback.

Index