Hewlett Packard
Enterprise

Scripting Tools for Windows PowerShell User Guide

OA Cmdlets v2.0

Abstract

This document contains instructions for using Scripting Tools for Windows PowerShell to manage HPE Onboard Administrator (OA). It is intended for system administrators who use the Scripting Tools for Windows PowerShell to manage their IT infrastructure.

Part Number: 781072-004
Published: August 2018
Edition: 1

Contents

Introduction to Scripting Tools for Windows PowerShell

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

The Scripting Tools for Windows PowerShell includes sets of PowerShell cmdlets for configuring HPE 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 OA. For more information about OA, see the HPE OA user guide and other related OA documents on the OA information library **http://www.hpe.com/info/oa-docs**.

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

OA Cmdlets for Windows PowerShell provides the following features:

- 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 PowerShell cmdlets, documenting syntax and usage examples.

Installation

System prerequisites

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

1. Install Microsoft .NET Framework 4.5 or later.

   Microsoft .NET Framework 4.5

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

   ---

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: OA Cmdlets are supported on the following operating system versions:

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

Installing the OA Cmdlets

Procedure

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

- It might be necessary to change the execution policy for PowerShell. 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
  ```

- The installation will halt and not complete successfully if any of the following conditions are detected:
  - Attempting to install without .NET 4.5 or above
  - Attempting to install without PowerShell 3.0 or above

NOTE:

You can install both the Scripting Tools for Windows PowerShell: OA Cmdlets versions 1.x and 2.x on the same system.

Installing OA cmdlets from the PowerShell Gallery

The PowerShell Gallery is a market place where PowerShell module or scripts from vendors, users, and individuals are stored in a cloud environment. It is a central repository for PowerShell content. You can choose to install the online version of OA cmdlets from the Microsoft gallery—`Install-Module -Name HPEOACmdlets -Verbose`. Visit the Microsoft gallery at **https://www.powershellgallery.com** and search for "HPEOACmdlets" for more details.

NOTE: Downloading from the PowerShell Gallery using the Cmdlet `Install-Module` may display the warning message, `WARNING: The specified module 'HPEOACmdlets' with PowerShellGetFormatVersion '2.0' is not supported by the current version of PowerShellGet. Get the latest version of the PowerShellGet module to install this module, 'HPEOACmdlets'.`

To resolve the problem, update the `PowerShellGet` module—**https://docs.microsoft.com/en-us/powershell/gallery/installing-psget**—and then install the `HPEOACmdlets` module.

Uninstalling the OA Cmdlets

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

Procedure

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

Uninstalling the OA cmdlets using the `Uninstall-Module` cmdlet

Use the `Uninstall-Module` cmdlet to remove the module from your system: `Uninstall-Module -Name HPEOACmdlets -Verbose`.

Recommendation

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

Repairing the OA Cmdlets

Use the installer repair option for the following scenarios:

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

Procedure

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

Scripting Tools for Windows PowerShell cmdlets

The following table provides a list and brief description of all the OA Cmdlets.

Cmdlet Help

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

```
help *hpeoa*
```

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

```
Get-Command –Module HPEOACmdlets
```

To display complete help for a specific cmdlet, type:

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

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

The OA cmdlets support the PowerShell `Update-Help` feature. When you execute this command, it accesses a Hewlett Packard Enterprise website, gets the most current help file(s), and puts them in the correct location on your system.

Table 1: OA Cmdlets for Windows PowerShell

Cmdlet	Description
Add-HPEOACertificate	Adds a CA, HPE SIM, User, RemoteSupport, or LDAP certificate to the command line.
Add-HPEOADNS	Adds an IP address of a DNS server to the list of DNS servers for either SERVER bays or INTERCONNECT bays.
Add-HPEOAEBIPA	Adds a DNS server IP address to the list of DNS servers.
Add-HPEOAIPv6Address	Adds an IPv6 static address for the OA.
Add-HPEOALanguage	Uploads and installs a language pack.
Add-HPEOALDAPBay	Assigns one or more bays to a specified LDAP group.
Add-HPEOALDAPGroup	Adds an LDAP group to the list of LDAP groups.
Add-HPEOALDAPPrivilege	Provides the specified group with access to the OA.
Add-HPEOARouteIPv6	Adds an IPv6 static route to the Onboard Administrator routing table.
Add- HPEOASNMPTrapReceiver	Adds a new SNMP trap receiver to the SNMP configuration.

Table Continued

Cmdlet	Description
Add-HPEOASNMPUser	Creates a new user to be used for SNMP v3 queries, traps, and informs.
Add-HPEOASSHKey	Adds an SSH key or keys to the Administrator local account.
Add-HPEOATrustedHost	Adds a new IPv4 or IPv6 address to the list of addresses being handled by the IP Security feature.
Add-HPEOAUser	Adds a user to the system.
Add-HPEOAUserBay	Assigns one or more bays to a user.
Add-HPEOAUserPrivilege	Assigns the OAs specified to an existing user.
Add-HPEOAVLAN	Creates a VLAN ID and an optional VLAN Name.
Clear-HPEOALoginBannerText	Clears the currently configured login banner text.
Clear-HPEOANTP	Disables access to the Primary or Secondary NTP server.
Clear-HPEOAFirmwareLog	Clears all data from the Enclosure Firmware Management logs.
Clear-HPEOASSHKey	Removes the authorized key file used for SSH login.
Clear-HPEOASysLog	Clears the OA system log.
Clear-HPEOAVCMode	All servers in the enclosure should be powered off before clearing the VCMODE.
Connect-HPEOA	Connects to an OA and creates an SSH session.
Disable-HPEOACRL	Disables certificate revocation checks.
Disable-HPEOAOCSP	Disables the certification revocation check.
Disconnect-HPEOA	Closes an SSH session with the OA.
Find-HPEOA	Find list of OA in a specified subnet.
Get-HPEOAAlertmail	Gets the alert mail configuration.
Get-HPEOABladeDeviceSerialNumber	Gets the specified direct attached blade device serial number.
Get-HPEOACACertificate	Gets a list of installed CA certificates.
Get-HPEOACACInfo	Gets the CAC authentication status.
Get-HPEOACertificate	Gets the certificate information for the OA, CA, Remote support, and LDAP.

Table Continued

Cmdlet	Description
Get-HPEOAConfig	Gets the script required to recreate the settings of the enclosure.
Get-HPEOADate	Gets the current date, time, and time zone of the internal clock of the enclosure.
Get-HPEOADiscoveryInfo	Gets information in OA discovery.
Get-HPEOAEBIPA	Gets EBIPA information.
Get-HPEOAEnclosureFan	Gets information for current status of the specified enclosure fan.
Get-HPEOAEnclosureInfo	Gets enclosure information.
Get-HPEOAEnclosureLCD	Gets information about the Insight Display screen.
Get-HPEOAEnclosurePowerCap	Gets the current Enclosure Dynamic Power Cap in watts.
Get-HPEOAEnclosurePowerCapBaysToExclude	Gets the bays in the enclosure that are exempt from the Enclosure Dynamic Power Cap.
Get-HPEOAEnclosurePowerSummary	Gets a detailed summary of the enclosure's present power state.
Get-HPEOAEnclosurePowerSupply	Gets power supply details of the enclosure.
Get-HPEOAEnclosureStatus	Gets the basic health and status of the enclosure subsystem.
Get-HPEOAEnclosureTemp	Gets the highest ambient temperature being reported by the installed blade devices.
Get-HPEOAFIPSMode	Gets the FIPS mode setting.
Get-HPEOAFRU	Gets FRU Summary section, which provides information on all field replaceable units within the enclosure.
Get-HPEOAFWLogServer	Gets the firmware log for the selected server or range of servers.
Get-HPEOAFWLogSession	Gets the firmware log session for the selected server or range of servers.
Get-HPEOAFWManagement	Gets enclosure firmware management configuration settings.
Get-HPEOAFWManagementLog	Gets the enclosure firmware management log.
Get-HPEOAFWSummary	Gets a summary of enclosure firmware components.
Get-HPEOAHealth	Gets current health of all components in the enclosure.
Get-HPEOAHPSIMInfo	Gets the current HPE SIM SSO configuration for the OA.

Table Continued

Cmdlet	Description
Get-HPEOAInfo	Gets information about the OA.
Get-HPEOAInterconnectInfo	Gets interconnect settings and information.
Get-HPEOAInterconnectList	Gets the interconnect list.
Get-HPEOAInterconnectPortMap	Gets the port mapping or the specified interconnects or range of interconnects.
Get-HPEOAInterconnectPowerDelay	Gets the PowerDelay status for the specified interconnects or range of interconnects.
Get-HPEOAInterconnectSession	Gets which users have serial console sessions in progress for each interconnect.
Get-HPEOAInterconnectStatus	Gets interconnect status information; UID state and health state.
Get-HPEOALanguage	Gets all language support packs installed.
Get-HPEOALDAPCertificate	Gets all LDAP certificates that are in effect on the OA.
Get-HPEOALDAPGroup	Gets the list of LDAP groups or the specified LDAP group details.
Get-HPEOALDAPInfo	Gets the LDAP settings, including enabled or disabled status, LDAP server, LDAP port, search contexts, and NT mapping state.
Get-HPEOALoginBanner	Gets the login banner settings.
Get-HPEOAModuleVersion	Gets the module details for the OA cmdlets.
Get-HPEOANetworkSetting	Gets network settings of OA.
Get-HPEOAPasswordSetting	Gets the current minimum password length and strong password settings.
Get-HPEOAPower	Gets the current power configuration.
Get-HPEOARackInfo	Gets the rack information for the enclosure.
Get-HPEOARackName	Gets the user defined rack name setting for the enclosure.
Get-HPEOARemoteSupport	Gets Remote Support settings and information.
Get-HPEOARemoteSupportEvents	Gets Remote Support events that have been sent.
Get-HPEOAServerBoot	Gets the boot settings for the specified servers.
Get-HPEOAServerDVD	Gets the DVD connection status for the specified server or range of servers.

Table Continued

Cmdlet	Description
Get-HPEOAServerFW	Gets the firmware log for the selected server or range of servers.
Get-HPEOAServerInfo	Gets server settings and details.
Get-HPEOAServerList	Gets a brief description of all server blades to which the current user has access.
Get-HPEOAServerName	Gets a brief description of all server blades to which the current user has access.
Get-HPEOAServerPortMap	Gets the port mapping for the server specified by the bay number.
Get-HPEOAServerPowerDelay	Gets the PowerDelay status for the specified server blade or range of server blades.
Get-HPEOAServerStatus	Gets power, health, thermal and UID setting of the server blade.
Get-HPEOAServerTemp	Gets the temperature sensor information for a specified server blade or range of server blades.
Get-HPEOASessionTimeout	Gets the current OA user session timeout.
Get-HPEOASNMP	Gets SNMP settings.
Get-HPEOASNMPUser	Gets information regarding SNMP users.
Get-HPEOASSHFingerprint	Gets the key fingerprint of the OA host public key.
Get-HPEOASSHKey	Gets the contents of the existing SSH authorized key files.
Get-HPEOASSlCipher	Gets the status of security cipher.
Get-HPEOASSLProtocol	Gets the status of SSL security protocol.
Get-HPEOAStatus	Gets health status for the OA.
Get-HPEOASyslog	Gets the syslog for the OA, for the specified server blade, enclosure, iLO, or OA.
Get-HPEOASyslogHistory	Gets the extended system log history for the OA.
Get-HPEOASyslogSetting	Gets the remote syslog settings for the OA.
Get-HPEOATopology	Gets the enclosures connected by the enclosure link), and all the linked enclosures information (enclosure name, status, Local or not, IP address, and UUID).
Get-HPEOATwoFactorInfo	Gets the configuration details for Two-Factor Authentication.
Get-HPEOAUpgradableDevice	Gets enclosure devices that are available for firmware upgrade.

Table Continued

Cmdlet	Description
Get-HPEOAUpTime	Gets uptime for the OA.
Get-HPEOAURB	Gets the URB reporting settings.
Get-HPEOAUSB	Gets which USB controller is currently enabled.
Get-HPEOAUSBKey	Gets a list of firmware images, configuration scripts, ISO images and other binary files present on the enclosure USB media.
Get-HPEOAUser	Gets all the user list or the specified user's details.
Get-HPEOAVCMode	Gets Virtual Connect Mode settings.
Get-HPEOAVLAN	Gets VLAN settings.
Invoke-HPEOAiLOCommand	Execute the iLO cmdlets through OA. It is used to interoperate with cmdlets in iLO cmdlets 1.2 or later.
Remove-HPEOACertificate	Removes the specified trust certificate (CA, HPSIM, User, RemoteSupport, LDAP).
Remove-HPEOADNS	Removes the IP address of a DNS server from the list for the specified OA.
Remove-HPEOAEBIPA	Removes the DNS server specified by the IP address from the list of DNS servers for either SERVER bays or INTERCONNECT bays.
Remove-HPEOAIPv6Address	Removes the IPv6 static address for the OA.
Remove-HPEOALanguage	Removes the user specified language.
Remove-HPEOALDAPBay	Disables access to the bays for the specified LDAP group.
Remove-HPEOALDAPGroup	Removes LDAP group from the system.
Remove-HPEOALDAPPrivilege	Disables access to the OA for the specified LDAP group.
Remove-HPEOARouteIPv6	Removes an IPv6 static route from the Onboard Administrator IPv6 routing table.
Remove-HPEOASNMPTrapReceiver	Removes an IP address from the list of systems that receive SNMP traps.
Remove-HPEOASNMPUser	Removes the specified SNMP user.
Remove-HPEOATrustedHost	Removes an IPv4 or IPv6 address from the list of addresses being handled by the IP Security feature.
Remove-HPEOAUser	Removes a user from the system.

Table Continued

Cmdlet	Description
Remove-HPEOAUserBay	Removes user access from a bay.
Remove-HPEOAUserCertificate	Removes the certificate associated with a user.
Remove-HPEOAUserPrivilege	Removes the OA from the control of the user to whom the OA is currently assigned.
Remove-HPEOAVLAN	Removes a VLAN ID.
Reset-HPEOA	Restarts OA, server, interconnect specified by bay number.
Reset-HPEOAiLO	Resets the iLO in a bay.
Save-HPEOAEBIPA	Saves EBIPA settings to flash drive.
Save-HPEOAVLAN	Saves VLAN configuration to FLASH.
Save-HPEOAVLANIPConfig	Saves VLAN IPCONFIG changes to FLASH.
Send-HPEOADataCollection	Sends a data collection to a remote server.
Set-HPEOAActiveHealthSystem	Enables or disables logging of inventory and health status for shared infrastructure items such as fans and power supplies to the blades that depend upon them.
Set-HPEOAAlertmail	Enables or disables the sending of alert emails when an event occurs and also edits the Alertmail settings.
Set-HPEOABladeDeviceSerialNumber	Sets the serial number of the specified Storage, Tape, or I/O expansion blade.
Set-HPEOADate	Sets the date of the enclosure.
Set-HPEOADHCPDomainName	Enables or disables DHCP domain name and sets the domain name for the OA.
Set-HPEOADHCPv6	Enables or disables DHCPv6 mode for management interfaces of all devices in the enclosure.
Set-HPEOAEBIPA	Enables or disables the OA to provide IP addresses to the devices in the bays using DHCP. This cmdlet also modifies EBIPA settings for bay device(s).
Set-HPEOAEnclosure	Sets the Enclosure details.
Set-HPEOAEnclosureiLOFederationSupport	Enables or disables OA support required to allow peer-to-peer network communication necessary for iLO Federation.
Set-HPEOAEnclosureIPMode	Enables or disables Enclosure IP Mode.

Table Continued

Cmdlet	Description
Set-HPEOAEnclosurePowerCap	Sets the Enclosure Dynamic Power Cap in watts AC.
Set-HPEOAFactoryDefault	Restores the OA to its factory defaults.
Set-HPEOAFIPSMode	Enables or disables use of the OA in a FIPS 140-2-compliant mode.
Set-HPEOAFQDNLinkSupport	Sets the `FQDNLinkSupport` to enable or disable.
Set-HPEOAFWManagement	Enables or disables enclosure firmware management and configures the firmware management settings.
Set-HPEOAGateway	Sets the network default gateway.
Set-HPEOAGUILoginDetail	Enables or disables extended enclosure information available in the GUI on the login page.
Set-HPEOAHPSIMTrustMode	Enables or disables the HPE SIM SSO mode.
Set-HPEOAHTTPS	Enables or disables HTTPS access to the OA.
Set-HPEOAIPConfig	Configures IP settings for the OA to DHCP or Static mode.
Set-HPEOAIPv6	Enables or disables IPv6 protocol for management interfaces of all devices in the enclosure.
Set-HPEOAIPv6DynamicDNS	Enables or disables dynamic DNS using IPv6 for either the specified bay or the active or standby OA.
Set-HPEOALDAP	Enables or disables directory authentication.
Set-HPEOALDAPServiceAccount	Sets the service account username and password of the LDAP server for CAC authentication.
Set-HPEOALDAPSetting	Sets the LDAP settings.
Set-HPEOALLF	Enables or disables Link Loss Failover for OA Redundancy and modifies the LLF interval.
Set-HPEOALoginBanner	Enables or disables the display of the configured login banner when the user attempts to log in to the OA and also modifies the banner text.
Set-HPEOAMinimumPasswordLength	Sets a minimum length for passwords.
Set-HPEOAName	Sets the OA name.
Set-HPEOANIC	Configures the external NIC for Auto-negotiation or forced link settings.

Table Continued

Cmdlet	Description
Set-HPEOANTP	Enables or disables NTP support for the OA. This cmdlet also sets the polling interval of the NTP servers and the primary and secondary servers for synchronizing time and date using the NTP.
Set-HPEOABladeBootFWDiscovery	Sets the offline firmware discovery policy for the enclosure.
Set-HPEOAPassword	Sets the password of the user that executed the command.
Set-HPEOAPower	Sets the power on the specified bay device.
Set-HPEOAPowerDelay	Set the PowerDelay status for the specified server or range of servers or for specified interconnect or range of interconnects.
Set-HPEOAPowerSetting	Configures power settings (NotRedundant, Redundant, and PowerSupply).
Set-HPEOARackName	Sets the rack name.
Set-HPEOARemoteSupport	Registers or unregisters the OA with the Insight Remote Support/ aggregator.
Set-HPEOARemoteSupportDirect	Sets Remote Support proxy settings, registers, or unregisters the OA enclosure for Remote Support Direct Connect (DIRECT mode).
Set-HPEOARemoteSupportMaintenance	Enables or disables the Remote Support maintenance window.
Set-HPEOARemoteSyslog	Sets the IP port number and IP address (or DNS name) for remote system log.
Set-HPEOARouterAdvertisement	Enables or disables auto-configuration of IPv6 addresses from Router Advertisement messages for management interfaces of all devices in the enclosure.
Set-HPEOASecureSh	Enables or disables SSH support for the OA.
Set-HPEOASerialBaud	Configures the baud rate settings for the OA serial console port.
Set-HPEOAServerBootOrder	Sets the boot device on the next boot or on permanent boot order.
Set-HPEOAServerDVD	Connects or disconnects the specified server or range of servers from the enclosure DVD drive.
Set-HPEOASessionTimeout	Sets the number of minutes before inactive sessions are removed.
Set-HPEOASLAAC	Enables or disables the auto-configuration of IPv6 addresses from SLAAC messages.

Table Continued

Cmdlet	Description
Set-HPEOASNMP	Enables or disables SNMP and modifies SNMP settings.
Set-HPEOASSlCipher	Sets SSL cipher to enable/disable for TLS connection.
Set-HPEOASSLProtocol	Sets the SSL security protocol.
Set-HPEOAStrongPassword	Enables or disables strong password requirements for user passwords.
Set-HPEOASysLogRemote	Enables or disables remote system logging.
Set-HPEOATelnet	Enables or disables telnet access to the OA.
Set-HPEOATimezone	Sets the time zone.
Set-HPEOATrustedHost	Enables or disables IP security for the OA.
Set-HPEOAUID	Sets the UID on or off for the OA, interconnect, server or enclosure.
Set-HPEOAURB	Enables or disables URB reporting and modifies URB reporting setting.
Set-HPEOAUSB	Allows the OA to select which USB controller to enable.
Set-HPEOAUser	Enables or disables a user account that was previously disabled by the Disable-HPEOAUser command and modifies user access level and other user information.
Set-HPEOAVLAN	Enables or disables VLAN on the enclosure.
Set-HPEOAVLANDefault	Sets or changes the default VLAN ID for the enclosure.
Set-HPEOAVLANFactoryDefault	Restores the VLAN settings to factory defaults.
Set-HPEOAVLANID	Sets the VLAN ID for the specified interconnect or range of interconnects.
Set-HPEOAVLANIPConfig	Temporarily sets the OA to DHCP or Static IP mode and the specified VLAN ID.
Set-HPEOAVLANName	Edits VLAN name for the specified VLAN ID.
Set-HPEOAVLANOA	Sets or changes the VLAN ID of the OA.
Set-HPEOAVLANRevert	Reverts VLAN settings back to saved FLASH configuration data in time indicated by the delay parameter.
Set-HPEOAXMLReply	Enables or disables XML reply data over an HTTP connection.

Table Continued

Cmdlet	Description
Start-HPEOACertificateDownload	Downloads the specified type of trusted certificate (CA, OA, HPSIM, LDAP, RemoteSupport, User).
Start-HPEOACertificateGeneration	Generates a pkcs#10 certificate request or a self-signed certificate.
Start-HPEOAConfigDownload	Downloads a previously saved configuration script file from a specific IP host, and then executes it.
Start-HPEOAConfigUpload	Uploads a script to the specified URL which duplicates the current runtime configuration.
Start-HPEOAFirmwareServerDiscovery	Starts manual firmware discovery.
Start-HPEOAKeyGeneration	Generates new private keys associated with the OA SSH service and/or SSL web services.
Start-HPEOASSHKeyDownload	Downloads an authorized key file to use for SSH logins.
Start-HPEOASupportDumpUpload	Uploads support dump data to the specified URL.
Start-HPEOASyslogUpload	Uploads the extended system log history for the current OA.
Switch-HPEOA	Forces the redundant OA to become the active OA.
Test-HPEOAAlertMail	Sends a test message to the configured email address.
Test-HPEOAConnection	Checks if the SSH connection to the OA is currently valid or expired.
Test-HPEOALDAP	Run LDAP tests and optionally attempt to login to the LDAP server using the username and password.
Test-HPEOALDAPServiceAccount	Tests the LDAP service account.
Test-HPEOARemoteSupport	Sends a test service alert.
Test-HPEOASNMP	Sends a test SNMP trap to all of the configured trap destinations.
Test-HPEOASyslog	Tests the remote system log settings by logging a test message to the syslog.
Test-HPEOAURB	Manually sends the URB message to the endpoint.
Update-HPEOAFirmware	Update OA firmware by multi methods.
Update-HPEOAFWServer	Initiates manual update of the selected servers using the configured HPE firmware ISO image URL.

Table Continued

Cmdlet	Description
Update-HPEOAiLO	Downloads a new flash image from the network and uses it to update the firmware for iLO.
Update-HPEOAModuleVersion	Checks if an updated version of the OA cmdlets is available on Hewlett Packard Enterprise website and returns the link to download this new version.

NOTE: The following cmdlets are supported only on OA firmware version 4.8 or later:

1. Add-HPEOARouteIPv6
2. Remove-HPEOARouteIPv6
3. Set-HPEOAFQDNLinkSupport
4. Set-HPEOASSLProtocol
5. Get-HPEOASSLProtocol
6. Set-HPEOASSlCipher
7. Get-HPEOASSlCipher
8. Clear-HPEOAFirmwareLog
9. Set-HPEOABladeBootFWDiscovery
10. Get-HPEOACACInfo
11. Disable-HPEOAOCSP
12. Set-HPEOALDAPServiceAccount
13. Test-HPEOALDAPServiceAccount

IPv6 support

Consider the following when using IPv6.

- IPv6 is supported on OA firmware version 4.3 or later
- 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-HPEOANetworkSetting
}
```

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-HPEOANetworkSetting
```

Using the `Find-HPEOA` cmdlet

When learning about the OA cmdlets, a good place to start is with the `Find-HPEOA` cmdlet. This cmdlet scans IP addresses and finds OAs 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 OA, it obtains basic information about the OA 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 OAs found.

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

```
PS C:\> Find-HPEOA 192.168.1.2
Warning : It might take a while to search all the OAs if the input is a
very large range. Use Verbose for more information.

IP            : 192.168.1.2
Hostname      : eastwind.company.net
ProductName   : c3000 Tray with embedded DDR2 Onboard Administrator
Firmware      : 3.11
Role          : ACTIVE
SerialNumber  : PAMMT0C9VXA09J
UUID          : 18PAMMT0C9VXA09J
```

The following script is an example of using `Find-HPEOA` with a search range which checks six addresses in which three OAs are found:

PowerShell script:

```
$OAS = Find-HPEOA 192.168.242.60-65 -Timeout 1000 -Verbose
$OAS
```

Script output:

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


VERBOSE: Using 6 threads for search
VERBOSE: Pinging 192.168.242.60
VERBOSE: Pinging 192.168.242.61
VERBOSE: Pinging 192.168.242.62
VERBOSE: Pinging 192.168.242.63
VERBOSE: Pinging 192.168.242.64
VERBOSE: Pinging 192.168.242.65
VERBOSE: No OA at 192.168.242.60
VERBOSE: No system responds at 192.168.242.61
VERBOSE: No system responds at 192.168.242.65

IP            : 192.168.242.62
Hostname      : eastwind.company.net
ProductName   : c3000 Tray with embedded DDR2 Onboard Administrator
Firmware      : 3.11
Role          : ACTIVE
SerialNumber  : PAMMT0C9VXA09J
UUID          : 18PAMMT0C9VXA09J

IP            : 192.168.242.63
Hostname      : westwind.company.net
ProductName   : BladeSystem c7000 DDR2 Onboard Administrator with KVM
Firmware      : 4.22
Role          : ACTIVE
```

```
SerialNumber  : OB93BP0215
UUID          : 09OB93BP0215

IP            : 192.168.242.64
Hostname      : southwind.company.net
ProductName   : c3000 Tray with embedded DDR2 Onboard Administrator
Firmware      : 4.22
Role          : ACTIVE
SerialNumber  : PAMMT0A9VWY01N
UUID          : 18PAMMT0A9VWY01N
```

The previous example script assigns the objects returned to a variable and then prints it out. There are two additional parameters in this script also. To monitor the operation of the `Find-HPEOA` cmdlet, we use the `Verbose` parameter. The verbose output tells you that six threads are being used for the operation and then it pings each address to test for systems in the range. You then see verbose output that lists unsuccessful attempts at finding an OA. The last output is the printing of the object `$OAS` that contains the three OAs found.

The `Timeout` parameter default is 6000 milliseconds. If the timeout value is not long enough for OAs to respond, try using a `Timeout` parameter with a larger value. A value of 10000 milliseconds (or more, if needed) 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-HPEOA` to locate OA devices. You may want to use those with other commands rather than input the OAs you find again or store them somewhere and re-use them.

The following is a script that pipes output from `Find-HPEOA` through `Add-Member` to add two required fields, then to `Connect-HPEOA`, and then to `Get-HPEOAPower` to produce the power information for the OAs found. The `–Verbose` parameter is used to view more information.

PowerShell script:

```
$EncInfo = Find-HPEOA 192.168.242.60-65 -Verbose |
% {Add-Member -PassThru -InputObject $_ Username Administrator}|
% {Add-Member -PassThru -InputObject $_ Password Admin}|
Connect-HPEOA |
Get-HPEOAPower -Verbose
$EncInfo
```

Script output:

The following is typical output from this script.

```
Warning : It might take a while to search for all the HP OAs if the input is a
very large range. Use Verbose for more information.
VERBOSE: Using 6 threads for search
VERBOSE: Pinging 192.168.242.60
VERBOSE: Pinging 192.168.242.61
VERBOSE: Pinging 192.168.242.62
VERBOSE: Pinging 192.168.242.63
VERBOSE: Pinging 192.168.242.64
VERBOSE: Pinging 192.168.242.65
VERBOSE: No OA at 192.168.242.60
VERBOSE: No system responds at 192.168.242.61
VERBOSE: No system responds at 192.168.242.65
VERBOSE: Using 3 threads


IP             : 192.168.242.62
Hostname       : eastwind.company.net
StatusType     : OK
StatusMessage  : OK
PowerMode      : Not Redundant
DynamicPower   : Enabled
SetPowerLimit  : Not Set
PowerCapacity  : 4800 Watts DC
PowerAvailable : 4193 Watts DC
PowerAllocated : 607 Watts DC
PresentPower   : 218 Watts AC
PowerLimit     : 5837 Watts AC

IP             : 192.168.242.63
Hostname       : westwind.company.net
StatusType     : OK
StatusMessage  : OK
PowerMode      : Not Redundant
DynamicPower   : Enabled
SetPowerLimit  : 12000 Watts AC
PowerCapacity  : 9860 Watts DC
PowerAvailable : 8384 Watts DC
PowerAllocated : 1476 Watts DC
PresentPower   : 914 Watts AC
PowerLimit     : 12000 Watts AC

IP             : 192.168.242.64
Hostname       : southwind.company.net
StatusType     : OK
StatusMessage  : OK
PowerMode      : Redundant
DynamicPower   : Disabled
SetPowerLimit  : Not Set
PowerCapacity  : 1674 Watts DC
```

```
PowerAvailable : 0 Watts DC
PowerAllocated : 1674 Watts DC
PresentPower   : 1043 Watts AC
PowerLimit     : 2036 Watts AC
```

The verbose output shown indicates that six threads are being used for `Find-HPEOA`, which lists each address being checked, and then three threads in the `Get-HPEOAPower` command. (This threading enables multiple commands to multiple OAs 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 OAs found. These are piped to `Connect-HPEOA` which makes the connection object array of the three OAs found.

Those are in turn passed through to `Get-HPEOAPower` which uses those connections and requests the power information from each OA. The final results are the power information for the three OAs found by `Find-HPEOA`. Without the `-Verbose` parameters you would see the Warning line from `Find-HPEOA` and the power information for the three OAs found in that range of addresses.

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

These cmdlets are used to determine the current version of the OA cmdlets module installed and update the OA cmdlets module if necessary.

The `Get-HPEOAModuleVersion` 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-HPEOAModuleVersion` cmdlet output.

```
Get-HPEOAModuleVersion


Name                      : HPEOACmdlets
Path                      : C:\Program Files (x86)\Hewlett Packard
                            Enterprise\PowerShell\Modules\HPEOACmdlets\HPEOACmdlets.dll
Description               : Scripting Tools for Windows PowerShell : OA Cmdlets create
                            an interface to HPE Onboard Administrator(OA).
                            These cmdlets can be used to get and set OA settings in
                            HPE BladeSystem c3000 and c7000 Enclosures.
GUID                      : 4a26845a-9a75-482c-8850-75fa4c1e6698
Version                   : 2.0.0.0
CurrentUICultureName      : en-US
CurrentUICultureVersion   : 2.0.0.0
AvailableUICulture        : {@{UICultureName=en-US; UICultureVersion=2.0.0.0},
                            @{UICultureName=zh-CN; UICultureVersion=2.0.0.0},
                            @{UICultureName=ja-JP; UICultureVersion=2.0.0.0}}
```

The `Update-HPEOAModuleVersion` 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-HPEOAModuleVersion
The currently installed version 2.0.0.0 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 the option to download the latest version.

```
PS C:\Users\Username> Update-HPEOAModuleVersion
There is a newer version of HPEOACmdlets available at
http://www.hpe.com/servers/powershell.
Do you want to info 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 `Connect-HPEOA` cmdlet

The `Connect-HPEOA` cmdlet is used to connect to OA using an IP address or hostname. The OA target can be a single OA or multiple OAs. For more information, see the `Connect-HPEOA` cmdlet help.

The following is an example of using the `Connect-HPEOA` cmdlet. In this example the connection is successful and the value of `$conObj.IsConnected` is True.

```
PS C:\> $conObj = Connect-HPEOA 192.168.242.61 -Username "username" -Password "password"

PS C:\> $conObj

ForwardedPorts       ConnectionInfo                                IsConnected KeepAliveInterval
--------------       --------------                                ----------- -----------------
{}                   Renci.SshNet.PasswordConnectionInfo                  True 00:00:00
```

If the connection fails, an error message similar to the following is displayed:

```
 Connect-HPEOA : Failed for 192.168.242.61:A connection attempt failed
because the connected party did not properly respond after a period of time,
or
established connection failed because connected host has failed to respond
192.168.242.61:22
At line:1 char:1
+ Connect-HPEOA 192.168.242.61 -Username Administrator -Password Admin
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : InvalidData: (:) [Connect-HPEOA], HPEOAErrorMsg
    + FullyQualifiedErrorId : HPEOACmdlets.ConnectHPEOA
```

You can use either Username and Password parameters or a Credential parameter to connect to an OA target. The following is an example of using the Credential parameter:

```
PS C:\> $credential = Get-Credential -Message "Please input username and password"
PS C:\> $conObj = Connect-HPEOA -OA 192.168.1.2 -Credential $credential
PS C:\> $conObj
ForwardedPorts ConnectionInfo IsConnected KeepAliveInterval
-------------- -------------- ----------- -----------------
{} Renci.SshNet.PasswordConnectionInfo True 00:00:00
```

Using cmdlets to manage OA

Once connection to OA is established other cmdlets can be used to manage OA. For example, the `Get-HPEOAEnclosureStatus` cmdlet can be used to display OA enclosure information as shown:

```
PS C:\> Get-HPEOAEnclosureStatus $conObj

IP                        : 192.168.1.2
Hostname                  : oahostv2.company.net
StatusType                : OK
StatusMessage             : OK
Enclosure                 : @{Status=OK; UnitIdentificationLED=Off; DiagnosticStatus=}
OnboardAdministrator      : @{Status=OK}
PowerSubsystem            : @{Status=OK; PowerMode=Not Redundant;
                            PowerCapacity=4800 Watts DC; PowerAvailable=4193
                            Watts DC; PresentPower=214 Watts AC}
CoolingSubsystem          : @{Status=OK; Good=6; Wanted=6; Needed=5;
                            Fan=System.Management.Automation.PSObject[]}
```

Other cmdlets can be used to set or update OA settings. For example, the `Set-HPEOAName` cmdlet can be used to change the OA name as follows:

```
PS C:\>  Set-HPEOAName $conObj -Name Examples
```

If the cmdlet is successful, no other message is displayed. If an error occurs, an output message similar to the following is displayed.

```
IP          Hostname             StatusType  StatusMessage
--          --------             ----------- --------------
192.168.1.2 oahostv2.company.net ERROR       {Invalid Arguments.}
```

You can use output type `RawText` to display what information is returned from the OA command line as shown in the following example.

```
PS C:\>  Set-HPEOAName $conObj -Name Examples -OutputType RawText
Onboard Administrator name changed to Examples.
```

Using the `Update-HPEOAFirmware` cmdlet

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

Procedure

1. Locate and download the OA firmware package from the following website: **http://www.hpe.com/info/oa**
2. Execute the downloaded firmware package `CPxxxxxx.exe` and extract the package to a local folder.
3. Execute `Update-HPEOAFirmware` 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-HPEOAFirmware -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-HPEOAFirmware -Server ilomx2232004p.company.net -Username admin
        -Password "admin123" -Location C:\aaaa.bin

IP                 : 192.168.1.1
Hostname           : ilomx2232004p.company.net
StatusType         : Warning
StatusMessage      : Detecting OA firmware version and settings...
                     This OA      (ACTIVE) :     4.80
                     Redundant OA (STANDBY):     4.80
                     The Active and Standby Onboard Administrator already
                     have the same firmware version installed.
```

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

Using parameters from a file

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

CSV file input

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

```
OAinputs.csv:

OA,Username,Password,Connection,Bay
192.168.1.2,Administrator,Admin,,All
192.168.1.3,Administrator,Admin,,All
192.168.1.4,Administrator,Admin,,All
```

When `Import-CSV` is used in the following example, the column headings in the CSV file become the data object field names in PowerShell. In the case of this CSV, it creates an array of three objects with five fields each. Note that the `Connection` field is left empty in the CSV file. This field is filled in during the processing in the script. Because this field is created by the `Connect-HPEOA` cmdlet, it cannot be imported from the CSV, but the `Connection` field placeholder is used until it is filled in. Once connections are made and the `Connection` fields are filled in, the input parameter array is piped to the `Get-HPEOABladeDeviceSerialNumber` cmdlet. Field names that are used by this cmdlet are `Connection` and `Bay`. All others are ignored.

CSV file input script

```
#set the working directory to where the script and data file are
```

```
$scriptdir = Split-Path -Parent $PSCommandPath
cd $scriptdir
#import the data that has these fields
#OA,Username,Password,Connection,Bay
#Connection is empty and filled in, in the next step
$path = ".\OAinputs.csv"
$csv = Import-Csv $path
#make each connection and fill in the Connection property
foreach ($oaitem in $csv) {
    $oaitem.Connection = Connect-HPEOA -OA $oaitem.OA -Username $oaitem.Username `
    -Password $oaitem.Password
}
#Get-HPEOABladeDeviceSerialNumber results
$sernum = $csv | Get-HPEOABladeDeviceSerialNumber
$sernum.Blade | Format-List
```

Script output

```


SerialNumber : US23456789
Bay  : 1

SerialNumber : USE951W9J8
Bay  : 2

SerialNumber : US23456789
Bay  : 3

SerialNumber : 6CU4100JM4
Bay  : 4

BladeStatus : The specified bay is subsumed.
Bay     : 5

BladeStatus : The specified bay is subsumed.
Bay     : 6

BladeStatus : The specified bay is subsumed.
Bay     : 7

BladeStatus : The specified bay is subsumed.
Bay     : 8

BladeStatus : The specified bay is empty.
Bay     : 1

BladeStatus : The specified bay is empty.
Bay     : 2

BladeStatus : The specified bay is empty.
Bay     : 3

BladeStatus : The specified bay is empty.
Bay     : 4

SerialNumber : USE951W9JE
```

```
Bay  : 5

SerialNumber : USE951W9JE
Bay  : 6

SerialNumber : USE951W9JE
Bay  : 7

SerialNumber : USE951W9JE
Bay  : 8

BladeStatus : The specified bay is empty.
Bay     : 9

BladeStatus : The specified bay is empty.
Bay     : 10

BladeStatus : The specified bay is empty.
Bay     : 11

BladeStatus : The specified bay is empty.
Bay     : 12

BladeStatus : The specified bay is subsumed.
Bay     : 13

BladeStatus : The specified bay is subsumed.
Bay     : 14

BladeStatus : The specified bay is subsumed.
Bay     : 15

BladeStatus : The specified bay is subsumed.
Bay     : 16
```

Interoperation of iLO and OA cmdlets

NOTE: The `Invoke-HPEOAiLOCommand` 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 HPE 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 OA), it is possible to send commands created by the iLO cmdlets to `Invoke-HPEOAiLOCommand`, 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 **Installing Scripting Tools for Windows PowerShell - OA Cmdlets** 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 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 generate just 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, a username, and a password. Validation of the parameters is not performed, but these parameters are required. Other parameters need to be specified as you would when sending them to an iLO or group of iLOs.

Using the `Invoke-HPEOAiLOCommand` cmdlet

The `Invoke-HPEOAiLOCommand` 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-HPEOAiLOCommand` 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-HPEOA -OA 192.168.242.62 -username “username” -password “password”
#get the RIBCL to send
$d = Get-HPiLODefaultLanguage -Server 192.168.242.62 -Username “username”
     -Password “password” –OutputType ExternalCommand
#use the connection to OA and the RIBCL command and send it to Bay 2
$c | Invoke-HPEOAiLOCommand –iLOCommand $d –Bay 2
```

Even though the iLO cmdlet has the same server, 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.hp.com
STATUS_TYPE    : OK
STATUS_MESSAGE : OK
LANG_ID        : en
LANGUAGE       : English
```

Log processing examples

The following examples demonstrate how to access OA log data. In these examples, you want to get a summary of the events in the logs available to the OA (that come from blades in the enclosure) 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 if necessary for a Caution or Critical event.

The first example gets and processes iLO event logs.

PowerShell script:

```
#Find the OAs that I want
$OAS = Find-HPEOA 192.168.242.63-65 -Verbose |
% {Add-Member -PassThru -InputObject $_ Username Administrator}|
% {Add-Member -PassThru -InputObject $_ Password Admin}
#Connect to the OAs
$conObj = $OAS | Connect-HPEOA
#get the iLO event logs from the blades in Bay 4
$rt = $conObj | Get-HPEOASysLog -Target iLO -Bay 4
#process the ilo event log returned from each OA from the iLO in Bay 4
foreach ($ilo in $rt) {
    $ilo.IP + " OA has " + $ilo.Bay.iLOEventLog.Count + " Bay 4 iLO log entries."
    $sevs = $(foreach ($event in $ilo.Bay.iLOEventLog) {$event.SEVERITY})
    $uniqsev = $($sevs | Sort-Object | Get-Unique)
    $sevcnts = $ilo.Bay.iLOEventLog | group-object -property SEVERITY –noelement
    "There are " + $uniqsev.Count + " type(s) of events in the iLO log in Bay 4.”
    $sevcnts | Format-Table
}
Disconnect-HPEOA $conObj
```

This script again uses the `Find-HPEOA` cmdlet to locate OAs. It adds username and password and then connects to the OAs. The `Get-HPEOASysLog` cmdlet is used with the `-Target` parameter set to iLO and the -Bay parameter set to 4. This gets the iLO logs from Bay 4 in the two OAs found. (You may notice that this code is very similar to log processing code in the Scripting Tools for Windows PowerShell user guide iLO Cmdlets.) The `Disconnect-HPEOA` cmdlet disconnects from the OAs that were connected to earlier.

Script output:

```
Warning : It might take a while to search all the HP OAs if the input is a
very large range. Use Verbose for more information.
VERBOSE: Using 3 threads for search
VERBOSE: Pinging 192.168.242.63
VERBOSE: Pinging 192.168.242.64
VERBOSE: Pinging 192.168.242.65
VERBOSE: No system responds at 192.168.242.65
192.168.242.63 OA has 150 Bay 4 iLO log entries.
There are 2 type(s) of events in the iLO log in Bay 4.

Count Name
----- ----
  109 Informational
   41 Caution


192.168.242.64 OA has 245 Bay 4 iLO log entries.
There are 2 type(s) of events in the iLO log in Bay 4.
```

```
Count Name
----- ----
  231 Informational
   14 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 a few code changes. The following is the modified script.

PowerShell script:

```
#Find the OAs that I want
$OAS = Find-HPEOA 192.168.242.63-65 -Verbose |
% {Add-Member -PassThru -InputObject $_ Username Administrator}|
% {Add-Member -PassThru -InputObject $_ Password Admin}
#Connect to the OAs
$conObj = $OAS | Connect-HPEOA
#Get the server information for the logs I am looking at
$serv = $conObj | Get-HPEOAServerInfo -Bay All
#get the Server Integrated Management Log (IML) from all Bays
$rt = $conObj | Get-HPEOASysLog -Target Server -Bay All
#process the IML entries returned from each OA from the iLOs in All Bays
for ($s = 0; $s -lt $rt.Count; $s++) {
    $ilo = $rt[$s]
    $sysinfo = $serv[$s]
    for ($b = 0; $b -lt $ilo.Bay.Count; $b++) {
        #Check to see if a string is returned (which is not log information)
        if ($ilo.Bay[$b].Syslog.GetType().Name -ne "String") {
            "Bay $($b+1) contains a $($sysinfo.ServerBlade[$b].Manufacturer)
            $($sysinfo.ServerBlade[$b].ProductName) $($sysinfo.ServerBlade[$b].Type)."
            "The OA at " + $ilo.IP + " retrieved " + $ilo.Bay[$b].Syslog.Count +
            " IML log entries from Bay $($b+1)."
            $sevs = $(foreach ($event in $ilo.Bay[$b].Syslog) {$event.SEVERITY})
            $uniqsev = $($sevs | Sort-Object | Get-Unique)
            $sevcnts = $ilo.Bay[$b].Syslog | group-object -property SEVERITY –noelement
            "There are " + $uniqsev.Count + " type(s) of events in the IML log in Bay $($b+1).”
            $sevcnts | Format-Table
        }
    }
}
Disconnect-HPEOA $conObj
```

Script output:

```
Warning : It might take a while to search all the HP OAs if the input is a
very large range. Use Verbose for more information.
VERBOSE: Using 3 threads for search
VERBOSE: Pinging 192.168.242.63
VERBOSE: Pinging 192.168.242.64
VERBOSE: Pinging 192.168.242.65
VERBOSE: No system responds at 192.168.242.65
Bay 2 contains a HP ProLiant BL420c Gen8 Server Blade.
The OA at 192.168.242.63 retrieved 7 IML log entries from Bay 2.
There are 3 type(s) of events in the IML log in Bay 2.
```

```
Count Name
----- ----
    4 Informational
    1 Caution
    2 Critical


Bay 4 contains a HP ProLiant BL660c Gen8 Server Blade.
The OA at 192.168.242.63 retrieved 13 IML log entries from Bay 4.
There are 2 type(s) of events in the IML log in Bay 4.

Count Name
----- ----
    2 Informational
   11 Caution


Bay 7 contains a HP ProLiant BL460c Gen8 Server Blade.
The OA at 192.168.242.63 retrieved 5 IML log entries from Bay 7.
There are 3 type(s) of events in the IML log in Bay 7.

Count Name
----- ----
    3 Informational
    1 Caution
    1 Critical


Bay 4 contains a HP ProLiant BL620c G7 Server Blade.
The OA at 192.168.242.64 retrieved 119 IML log entries from Bay 4.
There are 2 type(s) of events in the IML log in Bay 4.

Count Name
----- ----
   68 Informational
   51 Critical
```

Return objects and error handling

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

As in the preceding examples, when there is an error or warning message returned from an OA, it is indicated by a property in the returned object called `StatusType`. Enclosing OA cmdlets in `try` blocks and using `catch` for errors is a good practice, but it does not handle a returned OA error or warning. Three values can be returned in `StatusType`: 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 = $csv | Set-HPEOATrustedHost
if ($rt -ne $null) {
foreach ($item in $rt) {
switch ($item.StatusType) {
#OK status is not returned in a Set cmdlet
#but you can get a warning or error
'WARNING' { "I have been warned by " + $item.IP +
" that: " + $item.StatusMessage}
'ERROR' { "Somthing bad returned by " + $item.IP +
": " + $item.StatusMessage}
}
}
}
$rt = $csv | Get-HPEOAStatus
$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: Protocol is already disabled
I have been warned by 192.168.1.3 that: Protocol is already disabled

IP                    : 192.168.1.1
Hostname              :
StatusType            : OK
StatusMessage         : OK
OnboardAdministrator : {@{Name=OA-ABCXYZ; Role=Active; UID=Off; Status=OK; DiagnosticStatus=; Bay=2}}


IP                    : 192.168.1.3
Hostname              :
StatusType            : OK
StatusMessage         : OK
OnboardAdministrator : {@{Name=OA-XYZABC; Role=Active; UID=Off; Status=OK; DiagnosticStatus=; Bay=2}}
```

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, 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.

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 OAs 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.

Previous examples used `Find-HPEOA` to locate OAs to connect to. You may want to use other data sources such as .CSV files, .XML files, or databases to store and retrieve data to use for OAs and their usernames and passwords. These may need to be encrypted for security purposes. Encrypted storage and data use is beyond the scope of this document. It is not a recommended practice to embed passwords in scripts; instead they can be prompted for by omitting them as a parameter. You must be cognizant of your organizations security policies and code accordingly.

Troubleshooting

General issues

Verifying OA firmware versions

If a problem occurs, your first action should be to verify that the most current versions of OA firmware are installed. Updating to the most current firmware might solve the problem. For information on updating OA firmware, see the *HPE BladeSystem OA User Guide* or the `Update-HPEOAFirmware` cmdlet.

To determine if there is a newer version of OA cmdlets available, see **Using the Get-HPEOAModuleVersion and Update-HPEOAModuleVersion cmdlets** on page 24.

Usage tips

Invalid Argument(s) or Command

If the return data of an OA cmdlet is `Invalid Argument(s)` or `Invalid Command`, it typically indicates the cmdlet is not supported on the current OA version. Check the help info for the cmdlet (`Get-Help <cmdlet name> -Full`) or check the current OA firmware version (`Get-HPEOAInfo` or `Find-HPEOA`).

Leave XML Reply On

If xml reply of OA is turned off, `Find-HPEOA` does not find the OA, `Get-HPEOADiscoveryInfo` fails to get the OA data, and `Get-HPEOARackInfo` only gets limited OA data.

Cmdlets that can cause communication disruptions to an OA

Execution of some cmdlets may reset or reboot OA, cause connection to be lost, or set OA to the default factory settings (all the previous saved settings will be removed, even the IP may change). The `Set-HPEOAFIPSMode` cmdlet restores OA to the default factory settings and changes the OA password.

The cmdlets that may cause these types of behaviors include the following:

- `Set-HPEOASecureSh`
- `Set-HPEOATrustedHost`
- `Set-HPEOAFactoryDefault`
- `Set-HPEOAFIPSMode`
- `Set-HPEOAVLANOA`
- `Reset-HPEOA`
- `Update-HPEOAFirmware`
- `Set-HPEOAVLANIPConfig`
- `Set-HPEOAFactoryDefault`
- `Start-HPEOACertificateGeneration`
- `Start-HPEOAKeyGeneration`

Input parameter matching

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

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

```
#”value1” is used with $con1 and ”value2” is used with $con2
Get-HPEOAXXXX –Connection @($con1, $con2) –MandatoryParam @(“value1”,
“value2”)

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

#”value” is used with both $con1 and $con2
Get-HPEOAXXXX –Connection @($con1, $con2) –MandatoryParam “value”

#”value1” is used with $con1 and it will ask the user to input the value of
MandatoryParam for $con2
Get-HPEOAXXXX –Connection @($con1, $con2) –MandatoryParam @(“value1”)

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

In the following example, `Get-HPEOAYYYY` has an optional parameter (named “`OptionalParam`”).

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

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

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

#”value1” is used with $con1. No OptionalParam value for $con2 is given so a default value (if any) is used
Get- HPEOAYYYY –Connection @($con1, $con2) –OptionalParam @(“value1”)

#Value3 is used with $con1, value4 is used with $con2. value1 and value2 are not used.
#Priority is given to values from the commandline if both pipeline and commandline have the values.
$p1 = New-Object -TypeName PSObject -Property @{ "Connection"=$con1;"Fan"=value1} ;
$p2 = New-Object -TypeName PSObject -Property @{ "Connection"=$con2;"Fan"=value2} ;
@($p1,$p2)| Get-HPEOAXXX -Fan @(value3, value4)
```

Websites

General websites

Hewlett Packard Enterprise Information Library

www.hpe.com/info/EIL

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

Windows PowerShell websites

The following websites provide useful information for using PowerShell.

Microsoft Script Center

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

Windows PowerShell Blog

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

PowerShell.com

http://powershell.com/cs/

PowerShell Community Groups

http://powershellgroup.org/

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

Reporting PowerShell errors to Hewlett Packard Enterprise

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

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

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.