

Collecting Hardware Information from Client Machines using PowerShell Script

This document explains how to collect hardware and system information from one or many remote Windows client machines from a host machine using PowerShell Script.

Prerequisites (Except Windows 7)

1. Administrative access on the target client machines (local administrator) to query hardware info remotely.
2. PowerShell Remoting enabled on target clients (WinRM). You can enable remoting with **Enable-PSRemoting -Force** on the client (run as admin).
3. Network connectivity between host and client machines and firewall rules allowing WinRM **Enable-NetFirewallRule -Name "WINRM-HTTP-In-TCP"** (TCP 5985 for HTTP, TCP 5986 for HTTPS).

4. Credentials: a username/password with privileges on the remote machines (Host).

Allow your computer to trust specific remote hosts.

```
$cred = Get-Credential
```

```
Set-Item WSMan:\localhost\Client\TrustedHosts -Value "192.168.1.142"
```

This will prompt you to enter a username and password (e.g., MachineName\Administrator or Domain\UserName).

Then, you can use it with a remoting command, for example:

```
Invoke-Command -ComputerName REMOTE_PC_NAME -Credential $cred -  
ScriptBlock { hostname }
```

5. PowerShell 5.1+ (Windows) or PowerShell 7+

High-level Steps: (Host Machine)

1. Prepare a list of target computer names (or IPs).
2. Make sure remoting (WinRM) or CIM access is configured and firewall rules allow connections.
3. Run the provided PowerShell script from the host machine with appropriate credentials.

To execute the script, hit this command on shell:

```
Set-ExecutionPolicy Unrestricted
```

```
Set-ExecutionPolicy RemoteSigned
```

```
.\script.ps1
```

4. The script will collect hardware/system objects and write per-host summary CSV/HTML if requested.

Collecting Hardware Information from Client Machines using PowerShell Script

FOR WINDOWS 7

By default, Windows 7 comes with Windows PowerShell version 2.0 pre-installed.

To check PowerShell version:

➤ *\$PSVersionTable.PSVersion*

To get **Get-CimInstance** and many modern **cmdlets** you may want to install **WMF 5.1** for Windows 7

OR

You can still use the older WMI cmdlets such as **Get-WmiObject**, which are available by default in PowerShell 2.0 on Windows 7.

.NET Framework required by WMF upgrades (typically .NET 4.x).

Prerequisites:

For Client Machine (eg. Windows 7)

- ✓ **Administrative privileges**
- ✓ **To check PowerShell version**
\$PSVersionTable.PSVersion
- ✓ **Enable PowerShell remoting: (WMF 5.0+)**
Enable-PSRemoting -Force
This configures WinRM and firewall rules for basic remoting > powershell 2.0.
- ✓ **Enable PowerShell remoting: (Default Version of WMF / PowerShell 2.0)**
 - I. Starts the WinRM service
 - *winrm quickconfig*
 - II. This command manually creates a firewall rule to allow WinRM traffic.
 - *netsh advfirewall firewall add rule name="WinRM HTTP" dir=in action=allow protocol=TCP localport=5985*
 - III. This command lists all existing WinRM listeners on the system.
 - *winrm enumerate winrm/config/listener*
 - IV. This command manually creates a WinRM HTTP listener.
(Used when no listener exists or you need to recreate a broken listener)
 - *winrm create winrm/config/Listener?Address=*&Transport=HTTP*
Check the updated listeners detail again.
 - *winrm enumerate winrm/config/listener*
 - V. Checks the WinRM service status
 - *Get-Service WinRM*

Collecting Hardware Information from Client Machines using PowerShell Script

High-level Steps: (Host machine)

1. Prepare a list of target computer names (or IPs).
2. Make sure remoting (WinRM) or CIM access is configured and firewall rules allow connections.
3. Run the provided PowerShell script from the host machine with appropriate credentials.

To execute the script, hit this command on shell:

```
Set-ExecutionPolicy Unrestricted  
Set-ExecutionPolicy RemoteSigned  
.\script.ps1
```

4. The script will collect hardware/system objects and write per-host summary CSV/HTML if requested.

TIPS:

When you try to collect hardware information through a script on a Windows machine, you may run into several types of **errors** depending on environment (via PowerShell, WMI).

1. **Permission / Access Errors:**
Running without administrator privileges can block queries for system info (like BIOS, drivers, etc...)
2. **WMI / CIM Related Errors:**
 - i. **RPC/WMI Server Unavailable** : Happens when querying a remote machine but RPC/WMI services are not running or blocked by firewall.
 - ii. **Invalid Class** : The requested WMI class (e.g., Win32_BIOS, Win32_Processor) doesn't exist on that OS version.
3. **Execution Policy & Script Restrictions:**
Execution-policy errors: If the script is blocked due to PowerShell security policies
4. **Data Collection Failures**
 - i. **Null / Empty Values** : Some hardware details may not be exposed (e.g., Serial Number hidden in BIOS or virtual machines with missing fields).
 - ii. **Unsupported Hardware / Drivers** : If drivers aren't installed, queries for GPU, network, or storage may return incomplete or incorrect data.
 - iii. **Inconsistent Property Names** : Older Windows versions may not support the same WMI properties.
5. **Remote Collection Errors (if used)**
 - i. **Authentication failures** : Wrong credentials or insufficient privileges when collecting info remotely.

Collecting Hardware Information from Client Machines using PowerShell Script

- ii. **Firewall blocks** : Ports required for WMI (TCP 135, dynamic RPC ports) or WinRM (5985/5986) are blocked.
- iii. **Network issues** : Machine unreachable due to DNS or IP issues.
- ✓ **To avoid most issues:**
 - i. Always run PowerShell as **Administrator**.
 - ii. Use **Get-CimInstance** instead of **Get-WmiObject** (more modern, works better on newer Windows).
 - iii. Check **firewall/WinRM** when running remotely.
 - iv. Add error handling (-ErrorAction SilentlyContinue, Try/Catch) in your script.

Common Errors You May Encounter while collecting Windows 7:

1. PowerShell Version / Compatibility
 - Error: The term 'Get-ComputerInfo' is not recognized as the name of a cmdlet...
 - i. **Reason:** Get-ComputerInfo works only on PowerShell 5.1+ (Windows 10+). Windows 7 comes with PowerShell 2.0 by default.
 - ii. **Fix:** Use Get-WmiObject or Get-CimInstance instead.
2. WMI Service / Namespace Issues
 - Error: Get-WmiObject : Invalid class
 - i. **Reason:** Some WMI classes don't exist in Windows 7 (like Win32_BiosSerialNumber isn't valid, only Win32_BIOS works).
 - Error: The RPC server is unavailable.
 - i. **Reason:** WMI service not running, firewall blocking RPC, or remote query not enabled.
 - Error: Access is denied.
 - i. **Reason:** Running script without admin rights, or insufficient permissions for remote queries.
3. Execution Policy Restrictions
 - Error: File C:\script.ps1 cannot be loaded because running scripts is disabled on this system.
 - i. **Reason:** Windows 7 defaults to Restricted execution policy.
 - ii. **Fix:** Run Set-ExecutionPolicy RemoteSigned (with admin rights).
4. Missing .NET or Drivers
 - Error: Certain hardware classes return empty/null values.
 - i. **Reason:** Old .NET Framework or missing drivers (e.g., GPU, chipset) not exposing proper WMI data.
 - ii. **Example:** Win32_VideoController may not show memory info if drivers are generic.

Collecting Hardware Information from Client Machines using PowerShell Script

5. Remote Collection Errors (if running from host machine)
 - Error: WinRM cannot process the request.
 - i. **Reason:** WinRM is disabled by default on Windows 7.
 - ii. **Fix:** Run winrm quickconfig on client machine.
 - Error: Access denied even with correct credentials.
 - i. **Reason:** Firewall blocking WinRM calls.
- ✓ **Best Practices to Avoid Errors:**
 - i. Always run PowerShell as **Administrator**.
 - ii. Use **Get-WmiObject** instead of **Get-ComputerInfo** for Windows 7.
 - iii. Check **WMI service** (winmgmt) is running.
 - iv. Ensure **WinRM/firewall** configured if collecting remotely.