Microsoft Automation PowerPacks

Microsoft Hyper-V Automation PowerPack version 101

Windows PowerShell Automations PowerPack version 104

Windows PowerShell User-Initiated Automation PowerPack version 100

Table of Contents

Chapter 1

Introduction

Overview

This manual describes how to use the automation policies, automation actions, and custom action types found in Microsoft Automation Power Packs.

NOTE: This PowerPack is available with a ScienceLogic SL1 Standard solution. Contact your ScienceLogic Customer Success Manager or Customer Support to learn more.

NOTE: ScienceLogic provides this documentation for the convenience of ScienceLogic customers. Some of the configuration information contained herein pertains to third-party vendor software that is subject to change without notice to ScienceLogic. ScienceLogic makes every attempt to maintain accurate technical information and cannot be held responsible for defects or changes in third-party vendor software. There is no written or implied guarantee that information contained herein will work for all third-party variants. See the End User License Agreement (EULA) for more information.

This chapter covers the following topics:

Microsoft Automation PowerPacks

This manual describes content from the following PowerPack versions:

- Microsoft Hyper-V Automation, version 100
- Windows PowerShell Automations, version 104

Chapter 2

Windows PowerShell Automations

Overview

This chapter describes how to use the automation policies, automation actions, and custom action types found in the *Windows PowerShell Automations* PowerPack.

See the *Microsoft Hyper-V Automation* section for information about that PowerPack.

This chapter covers the following topics:

What is the Windows PowerShell AutomationsPowerPack?

The *Windows PowerShell Automations* PowerPack includes:

- A custom action type for running PowerShell commands on remote devices
- A dynamic device group with rules that include only Windows devices
- A set of automation actions that run diagnostic commands on Windows systems via PowerShell
- A set of automation policies that tie events from monitoring PowerPacks to the automation actions

The Windows PowerShell Automations actions are executed on the SL1 All-In-One Appliance or Data Collector.

In addition to using the standard content, you can use the content in the *Windows PowerShell Automations* PowerPack to:

- Create your own automation policies that include the pre-defined actions that run different sets of diagnostic commands.
- Use the supplied "Execute PowerShell Request" custom action type to configure your own automation action by supplying a set of commands to be executed via PowerShell.

Installing the Windows PowerShell AutomationsPowerPack

Before completing the steps in this manual, you must import and install the latest version of the *Windows PowerShell Automations*PowerPack.

IMPORTANT: You must install the *Datacenter Automation Utilities* PowerPack before using the *Windows PowerShell Automations* PowerPack.

NOTE: The *Windows PowerShell Automations*PowerPack requires SL1 version 8.10.0 or later. For details on upgrading SL1, see the appropriate SL1 Release Notes.

TIP: By default, installing a new version of a PowerPack overwrites all content from a previous version of that PowerPack that has already been installed on the target system. You can use the ***Enable Selective PowerPack Field Protection*** setting in the **Behavior Settings** page (System > Settings > Behavior) to prevent new PowerPacks from overwriting local changes for some commonly customized fields. (For more information, see the ***System Administration*** manual.)

IMPORTANT: The minimum required MySQL version is 5.6.0.

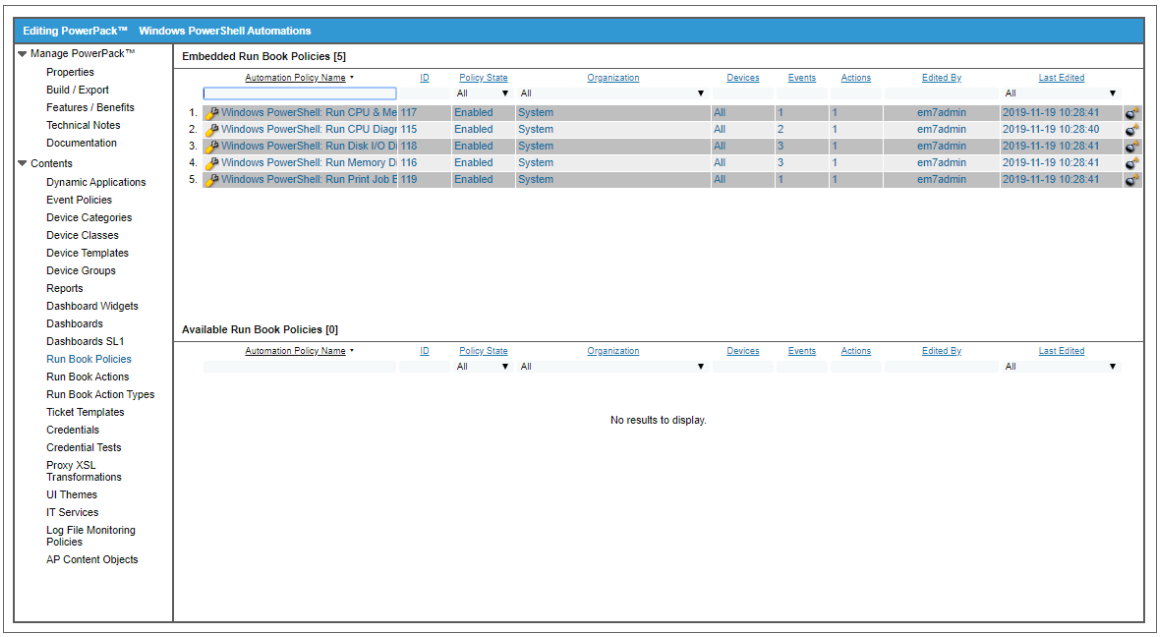
To download and install the PowerPack:

1. Search for and download the PowerPack from the **PowerPacks** page (Product Downloads > PowerPacks & SyncPacks) at the ScienceLogic Support Site.
2. In SL1, go to the **PowerPacks** page (System > Manage > PowerPacks).
3. Click the **[Actions]** button and choose *Import PowerPack*. The **Import PowerPack** dialog box appears.
4. Click **[Browse]** and navigate to the PowerPack file from step 1.
5. Select the PowerPack file and click **[Import]**. The **PowerPack Installer** modal displays a list of the PowerPack contents.
6. Click **[Install]**. The PowerPack is added to the **PowerPacks** page.

> **NOTE:** If you exit the **PowerPack Installer** modal without installing the imported PowerPack, the imported PowerPack will not appear in the **PowerPacks** page. However, the imported PowerPack will appear in the **Imported PowerPacks** modal. This page appears when you click the **[Actions]** menu and select *Install PowerPack*.

Standard Automation Policies

The *Windows PowerShell Automations* PowerPack includes five standard automation policies, shown in the following figure. Each policy triggers a single automation action that collects diagnostic data within a PowerShell session, and an action that formats the output as HTML. All of the automation actions use the same custom action type, "Execute PowerShell Request", which is supplied in the PowerPack.



All of the standard automation policies are tied to included ScienceLogic SL1 events generated by the Dynamic Applications from the Windows Server PowerPack.

Several of the automation actions use the substitution character feature of the “Execute PowerShell Request” custom action type. If an event variable is included in a command (such as "%Y" for the sub-entity name), the custom action type automatically replaces that variable with the value from the triggering event.

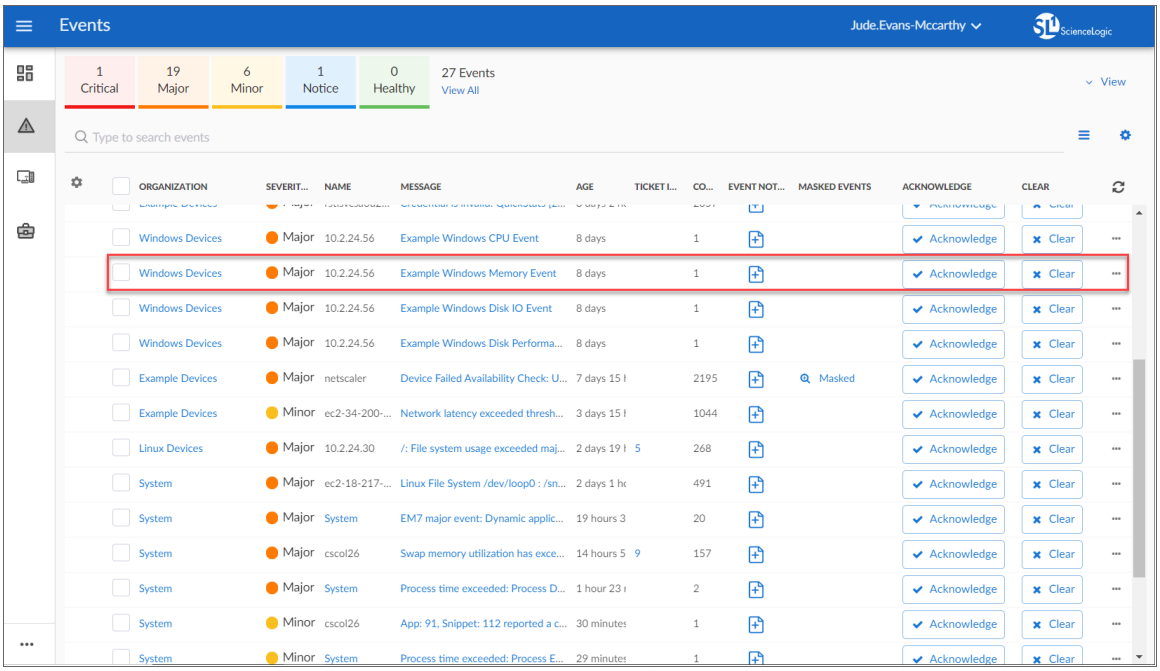
The following table shows the standard automation policies, their aligned events, and the automation action that runs in response to the events.

> **NOTE:** The aligned events are included as part of the *Microsoft Windows Server* PowerPack and are not installed with the SL1 platform. You must install the *Microsoft Windows Server* PowerPack to obtain these events.

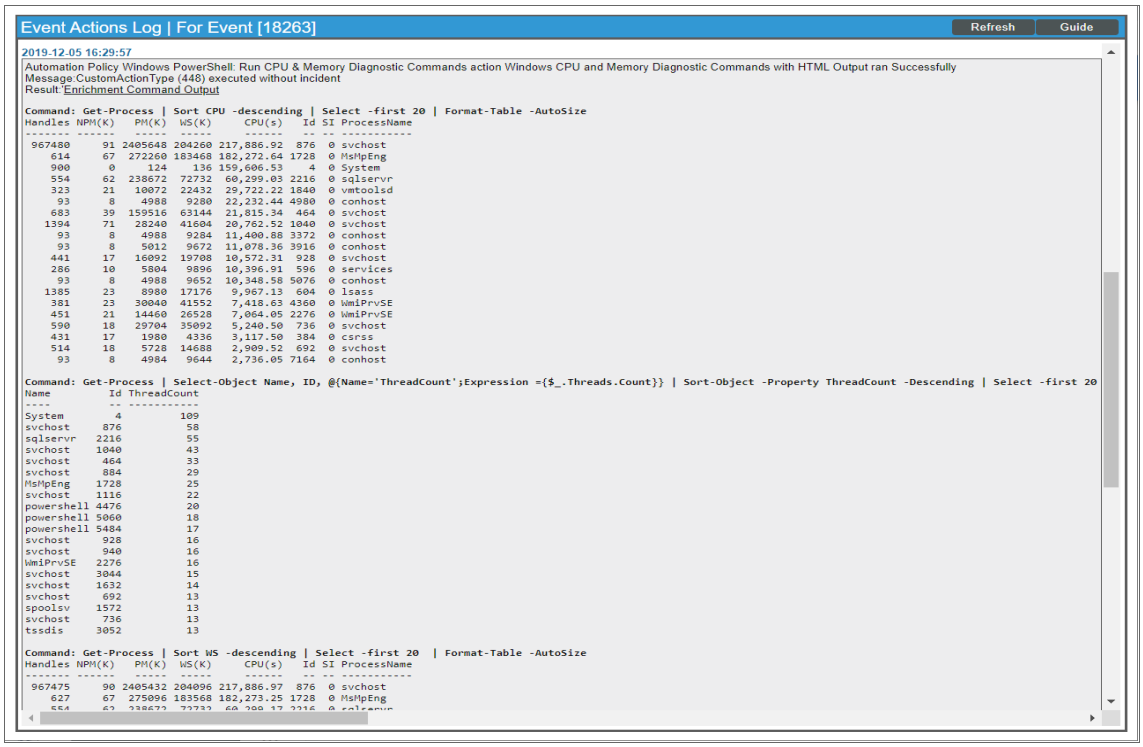
Automation Policy Name	Aligned Events	Automation Action
Windows PowerShell: Run CPU & Memory Diagnostic Commands	• Minor: Microsoft: Windows Disk Transfer Time (Physical Disk) exceeded threshold	Windows CPU and Memory Diagnostic Commands
Windows PowerShell: Run CPU Diagnostic Commands	• Minor: Microsoft: Windows CPU Utilization has exceeded the threshold • Minor: Microsoft: Windows Processor Queue Length exceeded the threshold	Windows CPU Diagnostic Commands
Windows PowerShell: Run Disk I/O Diagnostic Commands	• Minor: Microsoft: Windows % Disk Time (Logical Disk) exceeded threshold • Minor: Microsoft: Windows % Disk Time (Physical Disk) exceeded threshold • Minor: Microsoft: Windows Current Disk QueueLength (Physical Disk)exceeded threshold	Windows Disk I/O Diagnostic Commands
Windows PowerShell: Run Disk Usage Diagnostic Commands	• Poller: File system usage exceeded (major) threshold • Poller: File system usage exceeded (critical) threshold **NOTE:** This automation policy is aligned with the Windows Automation device group.	• Automation Utilities: Calculate Memory Size for Each Action • Windows Get Largest Event Log Files • Windows Get Largest Files on Disk • Windows Disk I/O Diagnostic Commands • Datacenter Automation: Format Output as HTML
Windows PowerShell: Run Memory Diagnostic Commands	• Major: Microsoft: Windows Available Memory below threshold • Major: Microsoft: Windows Pages per Second has exceeded threshold	Run Memory Diagnositc Commands

Automation Policy Name	Aligned Events	Automation Action
	• Minor: Microsoft: Windows Paging File Usage has exceeded threshold	
Windows PowerShell: Run Print Job Error Diagnostic Commands	• Minor: Microsoft: Windows: PowerShell: Print Job Errors exceeded threshold	Windows Print Job Error Diagnostic Commands

The following figure shows a memory event with a classification of "Major" appears on the **Events** page. Click the **[Actions]** button (···) for an event, and select *View Automation Actions* to see the automation actions triggered by the events.



The results shown for this event, in the Event Actions Log, include the automation policy that ran (shown at the top of the following figure), along with the automation actions (commands) that ran. Results for each command are also displayed. The following figure shows an example of this HTML output.



To learn more about which commands are executed by default for a given automation action, see *Customizing Actions*.

TIP: Although you can edit the automation actions described in this section, it is a best practice to use "Save As" to create a new automation action, rather than to customize the standard automation policies.

Authentication for Windows Devices with the Windows PowerShell Automations PowerPack

The "Execute PowerShell Request" custom action type supports hard-coded credentials (wherein you specify the ID of a credential in the automation action), or the custom action type can dynamically determine the credential to use. By default, the automation actions use the dynamic method (by specifying credential ID 0 in the input parameters). The dynamic method uses the first credential that matches the following rules:

- If the "Microsoft: Windows Server Configuration Cache" Dynamic Application (from the *Microsoft: Windows Server*PowerPack) is aligned to the device associated with the triggering event, the credential aligned to that Dynamic Application is used.
- If the "Microsoft: Windows Server Performance Cache" Dynamic Application (from the Microsoft: Windows ServerPowerPack) is aligned to the device associated with the triggering event, the credential aligned to that Dynamic Application is used.

- If the "Microsoft: Windows Server OS Configuration" Dynamic Application (from the Microsoft: Windows ServerPowerPack) is aligned to the device associated with the triggering event, the credential aligned to that Dynamic Application is used.
- If none of the listed Dynamic Applications are aligned to the device associated with the triggering event, the first available credential aligned to the device as a secondary credential is used.

Creating a Credential for Windows PowerShell

If you do not have the *Microsoft: Windows Server* PowerPack installed, you must create a credential that includes the username and password to communicate with your Windows devices. To create a credential, refer to the *Creating a Credential* section for more information.

To prepare your Windows systems for monitoring, follow the instructions in *Configuring Windows Servers for Monitoring with PowerShell*.

> **NOTE:** If you have the *Microsoft: Windows Server* PowerPack installed and configured, you may skip this section.

For more information about configuring credentials in SL1, see the ***Discovery and Credentials*** manual.

Creating Custom Windows PowerShell Automation Policies

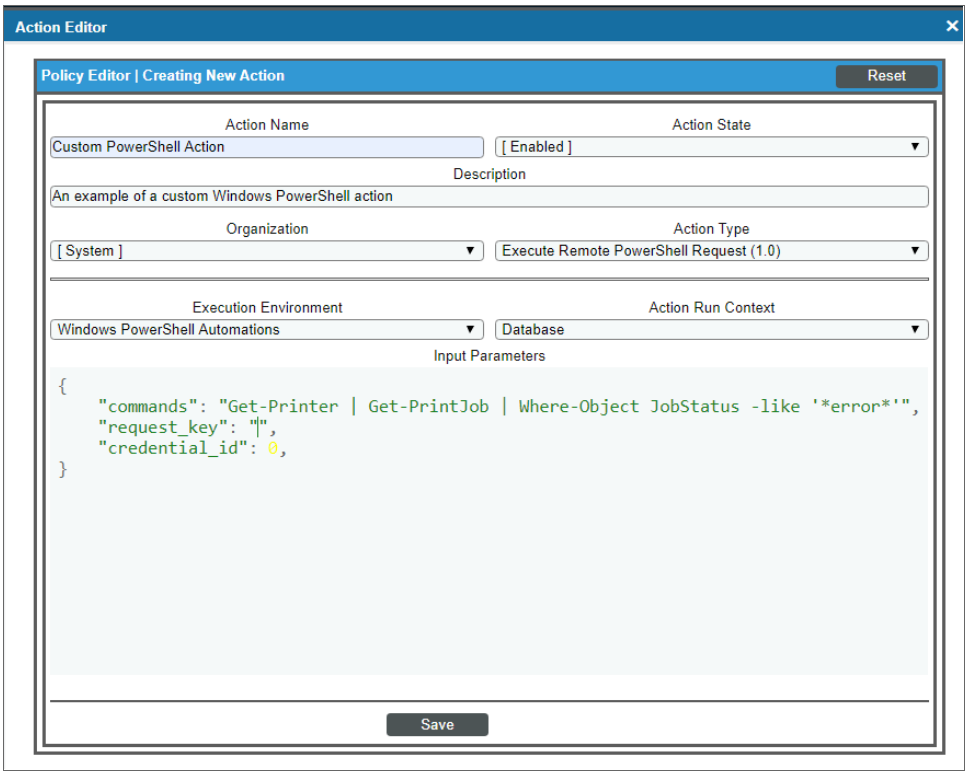
To create and customize Automation Policies for the *Windows PowerShell Automations* PowerPack, see the *Creating and Customizing Automation Policies* section.

Creating a Custom Action Policy

You can use the "Execute PowerShell Request" action type included with the Windows PowerShell Automations PowerPack to create custom automation actions that you can then use to build custom automation policies.

To create a custom action policy using the "Execute PowerShell Request" action type:

1. Navigate to the **Action Policy Manager** page (Registry > Run Book > Actions).
2. In the **Action Policy Manager** page, click the **[Create]** button.
3. The **Action Policy Editor** modal appears.



4. In the **Action Policy Editor** page, supply a value in each field.

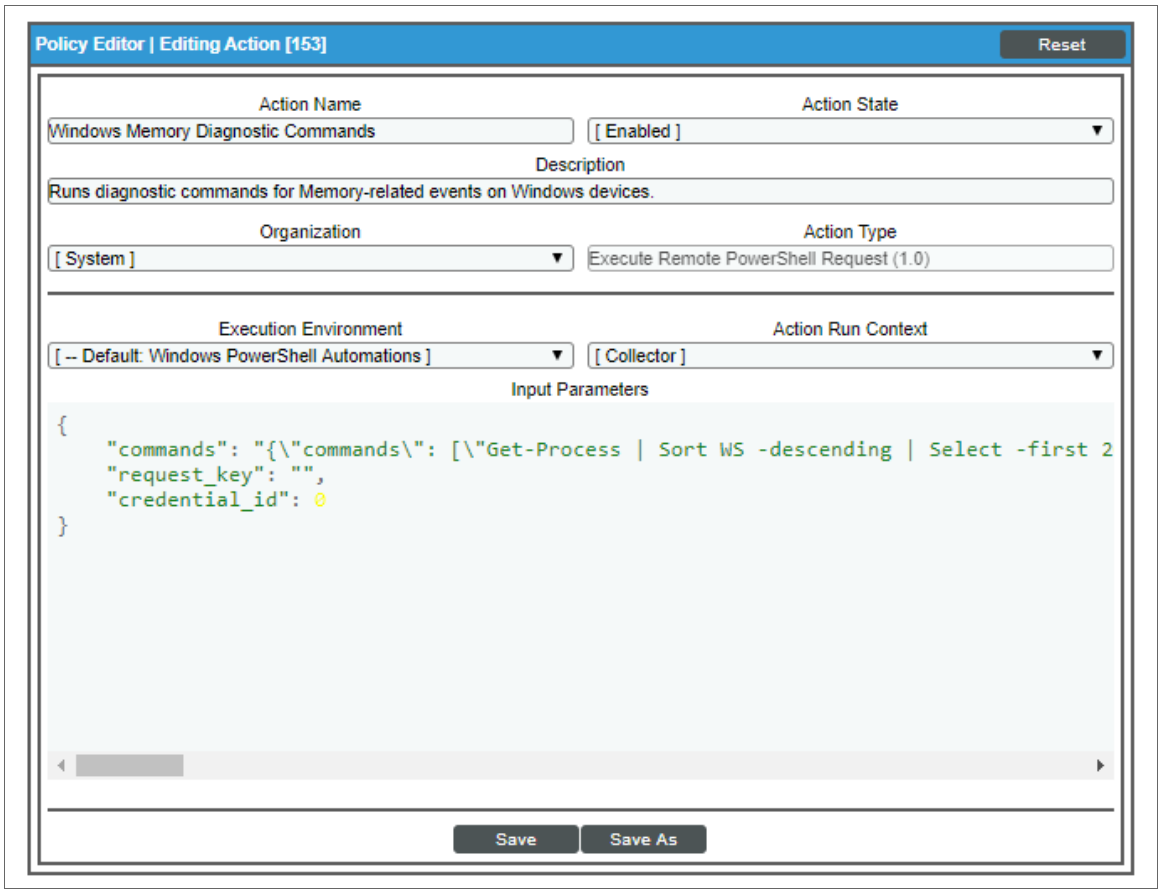
   - ***Action Name***. Specify the name for the action policy.
   - ***Action State***. Specifies whether the policy can be executed by an automation policy (enabled) or cannot be executed (disabled).
   - ***Description***. Allows you to enter a detailed description of the action.
   - ***Organization***. Organization to associate with the action policy.
   - ***Action Type***. Type of action that will be executed. Select the "Execute PowerShell Request (1.0)" action type (highlighted in the figure above).
   - ***Execution Environment***. Select from the list of available Execution Environments. The default execution environment is *System*.
   - ***Action Run Context***. Select *Database* or *Collector* as the context in which the action policy will run.
   - ***Input Parameters***. A JSON structure that specifies each input parameter. Each parameter definition includes its name, data type, and whether the input is optional or required for this Custom Action Type. For more information about the available input parameters, see the table in *Creating a New Windows PowerShell Automation Action*.

NOTE: Input parameters must be defined as a JSON structure.

6. Click **[Save]**. If you are modifying an existing action policy, click **[Save As]**. Supply a new value in the ***Action Name*** field, and save the current action policy, including any edits, as a new policy.

Customizing Automation Actions

The *Windows PowerShell Automations* PowerPack includes 5 automation actions that execute the "Execute PowerShell Request" action type to request diagnostic information or remediate an issue. You can specify the host and the options in a JSON structure that you enter in the ***Input Parameters*** field in the **Action Policy Editor** modal.



The following automation actions that use the "Execute PowerShell Request" action type are included in the *Windows PowerShell Automations* PowerPack. Compare the commands run with the example in the image above. For more information about input parameter fields, see the table in *Creating a New Windows PowerShell Automation Action*.

Action Name	Description	Commands Run
Windows CPU and Memory Diagnostic Commands	Runs diagnostic commands for CPU and Memory events on Windows devices	• `Get-Process | Sort CPU -descending | Select -first 20`

Action Name	Description	Commands Run
		• `Get-Process | Select-Object Name, ID, @{Name='ThreadCount';Expression = {$_.Threads.Count}} | Sort-Object -Property ThreadCount -Descending | Select -first 20` • `Get-Process | Sort WS -descending | Select -first 20` • `Get-CimInstance -Class Win32_PageFileUsage | Format-Table -Property Caption,Name,Status,Description,InstallDate, AllocatedBaseSize,PeakUsage,TempPageFile` • A command that collects the memory usage of running processes, where the memory usage is aggregated across all instances of each named process. The command is not listed here for clarity.
Windows CPU Diagnostic Commands	Runs diagnostic commands for CPU-related events on Windows devices	• `Get-Process | Sort CPU -descending | Select -first 20` • `Get-Process | Select-Object Name, ID, @{Name='ThreadCount';Expression = {$_.Threads.Count}} | Sort-Object -Property ThreadCount -Descending | Select -first 20`
Windows Disk I/O Diagnostic Commands	Runs diagnostic commands for Disk I/O events on Windows devices	• A command that collects the "IO Data Bytes per second" counter for each running process. The command takes 10 samples at 1-second intervals and returns the average of all samples for each process. The command is not listed here for clarity. • A command that collects the "IO Data Operations per second" counter for each running process. The command takes 10 samples at 1-second intervals and returns the average of all samples for each process. The command is not listed here for clarity.
Windows Get Largest Event Log Files	Gets the 20 largest Windows Event Log files.	• `Get-ChildItem C:\Windows\System32\winevt\Logs | Sort -Descending -Property length | Select -first 20`
Windows Get Largest Files on Disk	Gets the 20 largest files on the disk specified in the event.	• `Get-ChildItem %Y -r -erroraction 'silentlyContinue' | Sort -Descending -Property length | Select -first 20 | Select-Object FullName,@{Name='SizeMB';Expression= {[math]::Round($_.Length / 1MB,2)}}`
Windows Memory Diagnostic Commands	Runs diagnostic commands for Memory-related events on Windows	• `Get-Process | Sort WS -descending | Select -first 20`

Action Name	Description	Commands Run
	devices.	• `Get-CimInstance -Class Win32_PageFileUsage | Format-Table -Property Caption,Name,Status,Description,InstallDate, AllocatedBaseSize,PeakUsage,TempPageFile` • A command that collects the memory usage of running processes, where the memory usage is aggregated across all instances of each named process. The command is not listed here for clarity.
Windows Print Job Error Diagnostic Commands	Runs diagnostic commands for Print Job Error events on Windows devices.	• `Get-Printer | Get-PrintJob | Where-Object JobStatus -like '*error*'`

TIP: For more information about substitution variables, see *Appendix A*.

Creating a New Windows PowerShell Automation Action

You can create a new automation action that runs remote PowerShell requests using the supplied "Execute PowerShell Request" custom action type. To do this, select "Execute PowerShell Request" in the Action Type drop-down list when you create a new automation action. You can also use the existing automation actions in the PowerPack as a template by using the **[Save As]** option.

The Windows PowerShell automation actions accept the following parameters in JSON:

Parameter	Input type	Description
commands	string	Specifies a single command or a list of commands, in JSON format, to execute. You can use substitution variables in the commands.
request_key	string	**(Optional field)** **Default value**: empty The unique key for each instance (row) returned by the request. This unique key must be a property name, and the request must include that property (column) and return values from that property name (column). **Example**: Suppose you want to get the ID, number of cores, name, and maximum clock speed of every CPU installed on a Windows system, run the following command, where "DeviceID" is the request key. `Get-WmiObject -Class Win32_Processor -Property DeviceID, NumberOfCores, Name,`

Parameter	Input type	Description
		`MaxClockSpeed | Format-List DeviceID, NumberOfCores, Name, MaxClockSpeed`
credential_id	integer	**Default value**: 0 Specifies the credential_id to use for the connection. • If set to 0 (false), the custom action type will dynamically determine the credential. For more information, see *Authentication for Windows Devices*. • If set to an ID number, it maps to the credential ID specified. You can find credential IDs by going to System > Manage > Credentials.

Using Substitution Values. The commands input can contain substitution values that match the keys in EM7_VALUES.

TIP: For more information about substitution variables, see *Appendix A*.

For a description of all options that are available in Automation Policies, see the ***Run Book Automation*** manual.

Chapter 3

Microsoft Hyper-V Automation

Overview

This manual describes how to use the automation policies, automation actions, and custom action types found in the *Microsoft Hyper-V Automation* PowerPack. Installation of the *Windows PowerShell Automations* PowerPack is required before using the *Microsoft Hyper-V Automation* PowerPack.

> **NOTE:** ScienceLogic provides this documentation for the convenience of ScienceLogic customers. Some of the configuration information contained herein pertains to third-party vendor software that is subject to change without notice to ScienceLogic. ScienceLogic makes every attempt to maintain accurate technical information and cannot be held responsible for defects or changes in third-party vendor software. There is no written or implied guarantee that information contained herein will work for all third-party variants. See the End User License Agreement (EULA) for more information.

This chapter covers the following topics:

What is the Microsoft Hyper-V Automation PowerPack?

The *Microsoft Hyper-V Automation* PowerPack includes:

- A set of automation actions that run diagnostic commands on Hyper-V systems via PowerShell
- A set of automation policies that tie events from monitoring PowerPacks to the automation actions
- A dynamic device group for Hyper-V devices that is used to scope the automation policies

The Microsoft Hyper-V Automation actions are executed on the SL1 All-In-One Appliance or Data Collector.

In addition to using the standard content, you can use the content in the *Microsoft Hyper-V Automation* PowerPack to create your own automation policies that include the pre-defined actions that run different sets of diagnostic commands.

The *Microsoft Hyper-V Automation* PowerPack uses the supplied "Execute PowerShell Request" custom action type included with the *Windows PowerShell Automations* PowerPack.

Prerequisites

Before installing the Microsoft Hyper-V Automation PowerPack, you must perform the following actions:

- Install the *Microsoft: Hyper-V Server* PowerPack and configure it to monitor your Hyper-V device(s)
- Install version 103 or later of the *Windows PowerShell Automations* PowerPack
- Install version 102 or later of the *Datacenter Automation Utilities* PowerPack
- Install the Diag-V plug-in on your Hyper-V server. The plug-in is available here: https://gallery.technet.microsoft.com/scriptcenter/Diag-V-A-Hyper-V-0fe983e4

Installing the Microsoft Hyper-V Automation PowerPack

Before completing the steps in this manual, you must import and install the latest version of the *Microsoft Hyper-V Automation* PowerPack.

NOTE: The *Microsoft Hyper-V Automation* PowerPack requires SL1 version 8.10.0 or later. For details on upgrading SL1, see the appropriate SL1 Release Notes.

TIP: By default, installing a new version of a PowerPack overwrites all content from a previous version of that PowerPack that has already been installed on the target system. You can use the ***Enable Selective PowerPack Field Protection*** setting in the **Behavior Settings** page (System > Settings > Behavior) to prevent new PowerPacks from overwriting local changes for some commonly customized fields. (For more information, see the ***System Administration*** manual.)

IMPORTANT: The minimum required MySQL version is 5.6.0.

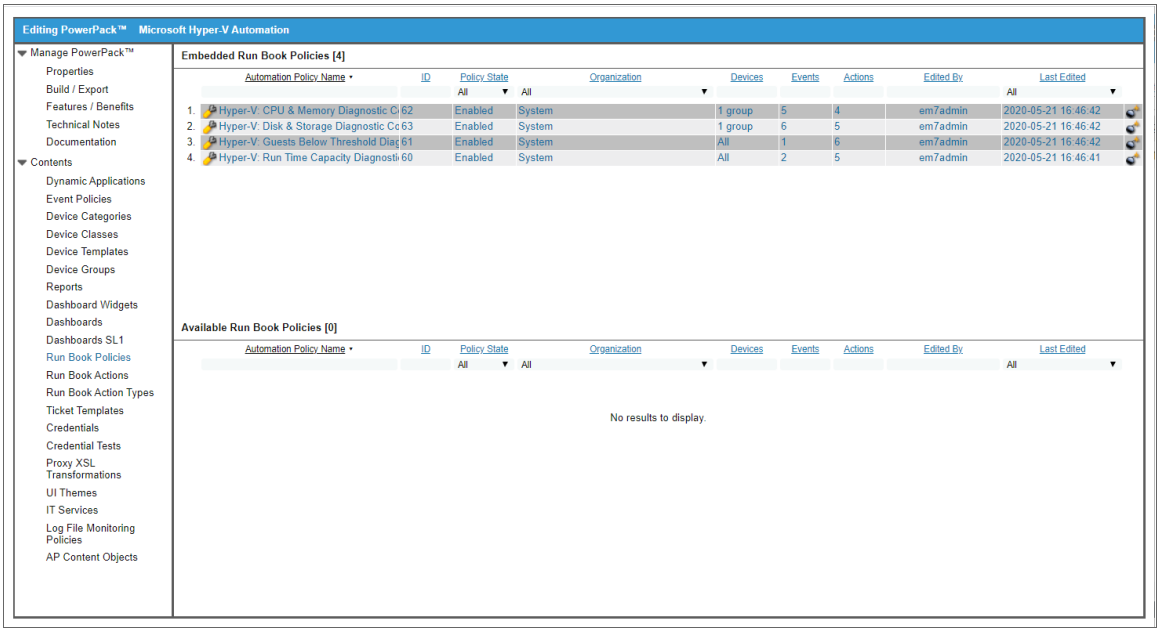
To download and install the PowerPack:

1. Search for and download the PowerPack from the **PowerPacks** page (Product Downloads > PowerPacks & SyncPacks) at the ScienceLogic Support Site.
2. In SL1, go to the **PowerPacks** page (System > Manage > PowerPacks).
3. Click the **[Actions]** button and choose *Import PowerPack*. The **Import PowerPack** dialog box appears.
4. Click **[Browse]** and navigate to the PowerPack file from step 1.
5. Select the PowerPack file and click **[Import]**. The **PowerPack Installer** modal displays a list of the PowerPack contents.
6. Click **[Install]**. The PowerPack is added to the **PowerPacks** page.

NOTE: If you exit the **PowerPack Installer** modal without installing the imported PowerPack, the imported PowerPack will not appear in the **PowerPacks** page. However, the imported PowerPack will appear in the **Imported PowerPacks** modal. This page appears when you click the **[Actions]** menu and select *Install PowerPack*.

Standard Automation Policies

The *Microsoft Hyper-V Automation* PowerPack includes four standard automation policies, shown in the following figure. Each policy triggers three automation actions that collect diagnostic data within a PowerShell session, and an action that formats the output in HTML. All of the automation actions use the same custom action type, "Execute PowerShell Request", which is supplied in the *Windows PowerShell Automations*PowerPack.



All of the standard automation policies are tied to included ScienceLogic SL1 events generated by the Dynamic Applications from the *Microsoft: Hyper-V Server* PowerPack.

Several of the automation actions use the substitution character feature of the "Execute PowerShell Request" custom action type. If an event variable is included in a command (such as "%Y" for the sub-entity name), the custom action type automatically replaces that variable with the value from the triggering event.

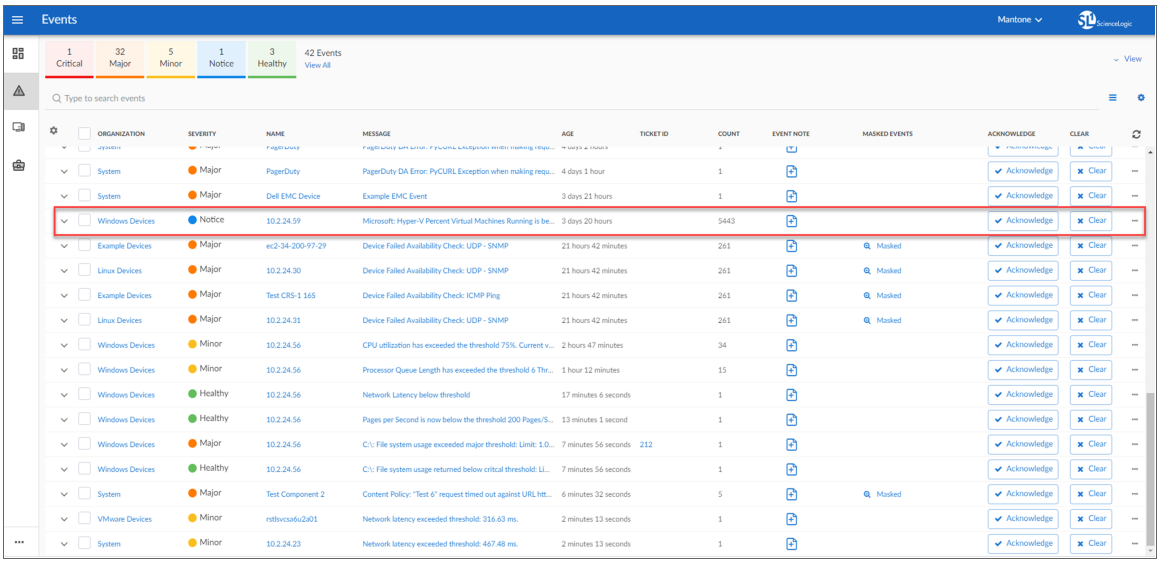
The following table shows the standard automation policies, their aligned events, and the automation actions that run in response to the events.

> **NOTE:** The aligned events are included as part of the *Microsoft: Hyper-V Server* PowerPack and are not installed with the SL1 platform. You must install the *Microsoft: Hyper-V Server* PowerPack to obtain these events.

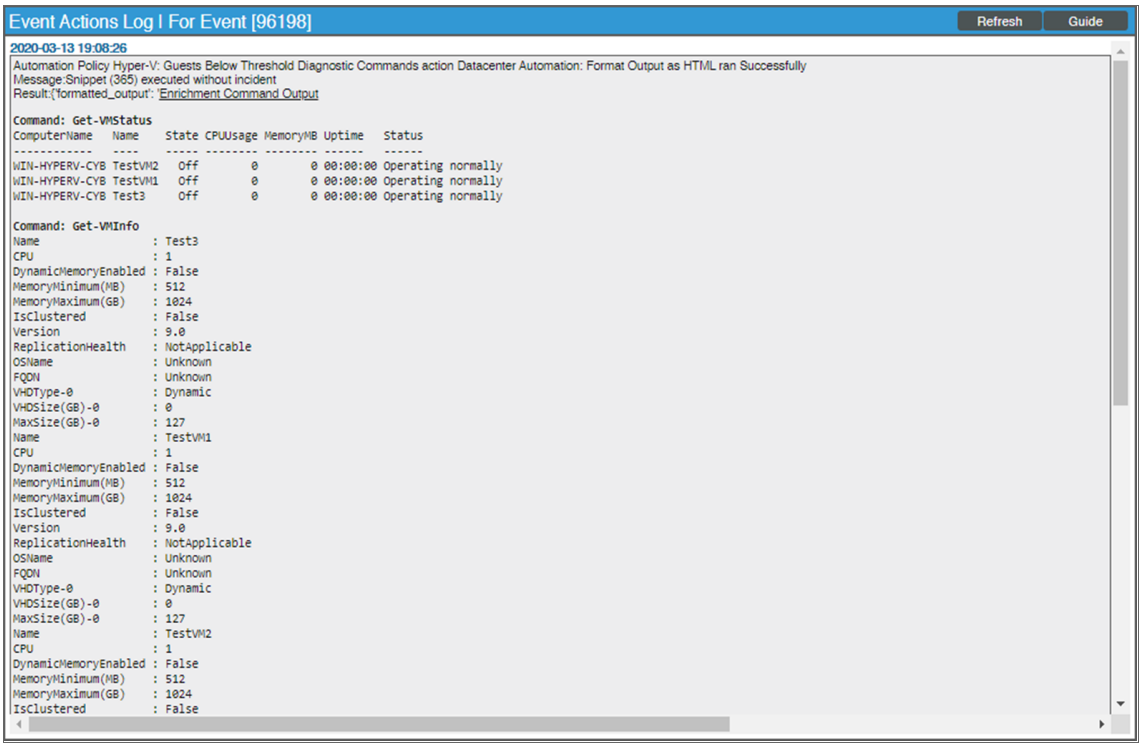
Automation Policy Name	Aligned Events	Automation Actions
Hyper-V: CPU & Memory Diagnostic Commands	• Microsoft: Windows CPU Utilization has exceeded the threshold • Microsoft: Windows Processor Queue Length exceeded the threshold • Microsoft: Windows Available Memory below threshold • Microsoft: Windows Pages per Second has exceeded threshold • Microsoft: Windows Paging File has exceeded threshold	• Automation Utilities: Calculate Memory Size for Each action • Hyper-V Guest Status Diagnostic Commands • Hyper-V Log Collection • Datacenter Automation: Format Output as HTML
Hyper-V: Disk & Storage Diagnostic Commands	• Microsoft: Windows Disk Transfer Time (Physical Disk) exceeded threshold • Microsoft: Windows % Disk Time (Logical Disk) exceeded threshold • Microsoft: Windows % Disk Time (Physical Disk) exceeded threshold • Microsoft: Windows Current Disk Queue Length (Physical Disk) exceeded threshold • Poller: File system usage exceeded (major) threshold • Poller: File system usage exceeded (critical) threshold	• Automation Utilities: Calculate Memory Size for Each action • Hyper-V Guest Replication Diagnostic Command • Hyper-V Guest Status Diagnostic Commands • Hyper-V Guest Storage Diagnostic Commands • Datacenter Automation: Format Output as HTML
Hyper-V: Guests Below Threshold Diagnostic Commands	• Microsoft: Hyper-V Percent VMs Running below threshold	• Automation Utilities: Calculate Memory Size for Each Action • Hyper-V Guest Replication Diagnostic Command

Automation Policy Name	Aligned Events	Automation Actions
		• Hyper-V Guest Status Diagnostic Commands • Hyper-V Guest Storage and Replication Diagnostic Commands • Hyper-V Log Collection • Datacenter Automation: Format Output as HTML
Hyper-V: Run Time Capacity Diagnostic Commands	• Microsoft: Hyper-V Percent Total Run Time has exceeded major threshold • Microsoft: Hyper-V Percent Total Run Time has exceeded minor threshold	• Automation Utilities: Calculate Memory Size for Each Action • Hyper-V Guest Status Diagnostic Commands • Hyper-V Allocation Diagnostic Commands • Hyper-V Log Collection • Datacenter Automation: Format Output as HTML

The following figure shows a memory event with a classification of "Major" appears on the **Events** page. Click the **[Actions]** button (···) for an event, and select *View Automation Actions* to see the automation actions triggered by the events.



The results shown for this event, in the Event Actions Log, include the automation policy that ran (shown at the top of the following figure), along with the automation actions (commands) that ran. Results for each command are also displayed. The following figure shows an example of this HTML output.



To learn more about which commands are executed by default for a given automation action, see *Customizing Actions*.

TIP: Although you can edit the automation policies described in this section, it is a best practice to use "Save As" to create a new automation action, rather than to customize the standard automation policies.

Credential for Hyper-V Automation

The *Microsoft Hyper-V Automation* PowerPack uses the same credential that you created for the *Windows PowerShell Automations* PowerPack. Refer to the *Creating a Credential* section for more information.

NOTE: If you have the *Microsoft: Windows Server* PowerPack installed and configured, you may skip this section.

For more information about configuring credentials in SL1, see the ***Discovery and Credentials*** manual.

Creating and Customizing Hyper-V Automation Policies

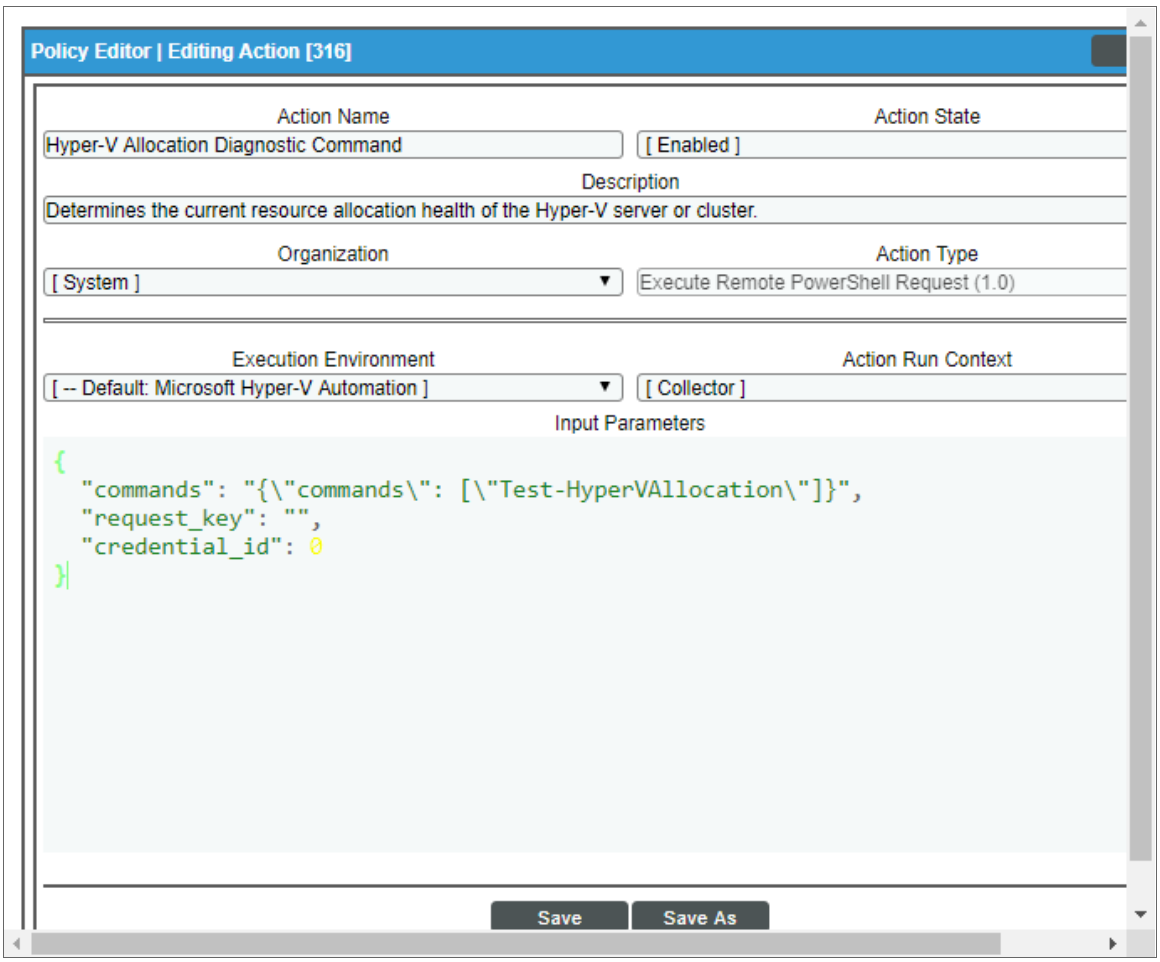
To create and customize Automation Policies for the *Microsoft Hyper-V Automation* PowerPack, see the *Creating and Customizing Automation Policies* section.

Creating a Custom Action Policy for Hyper-V

You can use the "Execute PowerShell Request" action type included with the *Windows PowerShell Automations* PowerPack to create custom automation actions that you can then use to build custom automation policies. To create a custom action policy, see the *Creating a Custom Action Policy* section.

Customizing Automation Actions

The *Microsoft Hyper-V Automation* PowerPack includes 2 automation actions that execute the "Execute PowerShell Request" action type to request diagnostic information or remediate an issue. You can specify the commands and the options in a JSON structure that you enter in the ***Input Parameters*** field in the **Action Policy Editor** modal.



The following automation actions that use the "Execute PowerShell Request" action type are included in the *Microsoft Hyper-V Automation* PowerPack. Compare the commands run with the example in the image above. For more information about input parameter fields,
see the table in *Creating a New Microsoft Hyper-V Automation Action*.

Action Name	Description	Commands Run
Hyper-V Allocation Diagnostic Command	Determines the current resource allocation health of the Hyper-V server or cluster.	• `Test-HyperVAllocation`
Hyper-V Guest Replication Diagnostic Command	Runs a diagnostic command related to Hyper-V guest replication	• `Get-VMReplicationStatus`
Hyper-V Guest Status Diagnostic Commands	Runs diagnostic commands to collect Hyper-V guest status and configuration information.	• `Get-VMStatus` • `Get-VMInfo | Format-Table`
Hyper-V Guest Storage Diagnostic Commands	Runs diagnostic commands related to Hyper-V Guest storage and replication.	• `Get-VMLocationPathInfo | Format-Table` • `Get-VMSharedVHDs | Format-Table`
Hyper-V Log Collection	Collects the most recent 25 log entries from the Hyper-V logs.	• `Get-HyperVLogInfo -StartDate ((Get-Date).addminutes(-10))`

TIP: For more information about substitution variables, see *Appendix A*.

Creating a New Microsoft Hyper-V Automation Action

You can create a new automation action that runs remote PowerShell requests using the supplied "Execute PowerShell Request" custom action type. To do this, refer to the *Creating a New Windows PowerShell Automation Action* section

For a description of all options that are available in Automation Policies, see the ***Run Book Automation*** manual.

Chapter 4

Configuring Device Credentials

This chapter describes how to configure the credentials required by the automation actions in the Microsoft Automations PowerPacks.

This chapter covers the following topics:

NOTE: ScienceLogic provides this documentation for the convenience of ScienceLogic customers. Some of the configuration information contained herein pertains to third-party vendor software that is subject to change without notice to ScienceLogic. ScienceLogic makes every attempt to maintain accurate technical information and cannot be held responsible for defects or changes in third-party vendor software. There is no written or implied guarantee that information contained herein will work for all third-party variants. See the End User License Agreement (EULA) for more information.

Creating a Credential

If you are creating a credential for the *Windows PowerShell Automations* PowerPack and do not have the *Microsoft: Windows Server* PowerPack installed, you must create a credential that includes the username and password to communicate with your Windows devices.

To prepare your Windows systems for monitoring, follow the instructions in *Configuring Windows Servers for Monitoring with PowerShell*.

> **NOTE:** If you have the *Microsoft: Windows Server* PowerPack installed and configured, you may skip this section.

To define a PowerShell credential in SL1:

1. Collect the information you need to create the credential:
    - The username and password for a user on the Windows device.
    - If the user is an Active Directory account, the hostname or IP address of the Active Directory server and the domain.
    - Determine if an encrypted connection should be used.
    - If you are using a Windows Management Proxy, the hostname or IP address of the proxy server.
2. Go to the **Credential Management** page (System > Manage > Credentials).
3. In the **Credential Management** page, click the **[Actions]** menu. Select ***Create PowerShell Credential***.
4. The **Credential Editor** page appears, where you can define the following fields:
    - ***Profile Name***. Name of the credential. Can be any combination of alphanumeric characters. This field is required.
    - ***Hostname/IP***. Hostname or IP address of the device from which you want to retrieve data. This field is required.
        - You can include the variable **%D** in this field. SL1 will replace the variable with the IP address of the device that is currently using the credential.
        - You can include the variable **%N** in this field. SL1 will replace the variable with the hostname of the device that is currently using the credential. If SL1 cannot determine the hostname, SL1 will replace the variable with the primary, management IP address for the current device.
        - You can include the prefix **HOST** or **WSMAN** before the variable **%D** in this field if the device you want to monitor uses a service principal name (for example, "HOST://%D" or "WSMAN://%D"). SL1 will use the WinRM service HOST or WSMan instead of HTTP and replace the variable with the IP address of the device that is currently using the credential.
    - ***Username***. Type the username for an account on the Windows device to be monitored or on the proxy server. This field is required.

NOTE: The user should not include the domain name prefix in the username for Active Directory accounts. For example, use "em7admin" instead of "MSDOMAIN\em7admin".

- ***Encrypted***. Select whether SL1 will communicate with the device using an encrypted connection. Choices are:
  - *yes*. When communicating with the Windows server, SL1 will use a local user account with authentication of type "Basic Auth". You must then use HTTPS and can use a Microsoft Certificate or a self-signed certificate.
  - *no*. When communicating with the Windows server, SL1 will not encrypt the connection.
- ***Port***. Type the port number used by the WinRM service on the Windows device. This field is automatically populated with the default port based on the value you selected in the ***Encrypted*** field. This field is required.
- ***Account Type***. Type of authentication for the username and password in this credential. Choices are:
  - *Active Directory*. On the Windows device, Active Directory will authenticate the username and password in this credential.
  - *Local*. Local security on the Windows device will authenticate the username and password in this credential.
- ***Timeout (ms)***. Type the time, in milliseconds, after which SL1 will stop trying to collect data from the authenticating server. For collection to be successful, SL1 must connect to the authenticating server, execute the PowerShell command, and receive a response within the amount of time specified in this field.
- ***Password***. Type the password for the account on the Windows device to be monitored or on the proxy server. This field is required.
- ***PowerShell Proxy Hostname/IP***. If you use a proxy server in front of the Windows devices you want to communicate with, type the fully-qualified domain name or the IP address of the proxy server in this field.
- ***Active Directory Hostname/IP***. If you selected Active Directory in the ***Account Type*** field, type the hostname or IP address of the Active Directory server that will authenticate the credential.
- ***Domain***. If you selected Active Directory in the ***Account Type*** field, type the domain where the monitored Windows device resides.

5. To save the credential, click the **[Save]** button. To clear the values you set, click the **[Reset]** button.

For more information about configuring credentials in SL1, see the ***Discovery and Credentials*** manual.

Chapter 5

Creating and Customizing Automation Policies

Overview

This chapter describes how to create automation policies using the automation actions in Microsoft Automation PowerPacks.

This chapter covers the following topics:

Prerequisites

Before you create an automation policy using the automation actions in *Microsoft Automation* PowerPacks, you must determine:

- Which set of commands you want to run on a monitored device when an event occurs. There are ten automation actions in the PowerPack that run the "Execute PowerShell Request" action type with different commands. You can also create your own automation actions using the custom action type supplied in the PowerPack.
- What event criteria you want to use to determine when the automation actions will trigger, or the set of rules that an event must match before the automation is executed. This can include matching only specific event policies, event severity, associated devices, and so on. For a description of all the options that are available in Automation Policies, see the ***Run Book Automation*** manual.

Creating an Automation Policy

To create an automation policy that uses the automation actions in *Microsoft Automation* PowerPacks, perform the following steps:

1. Go to the **Automation Policy Manager** page (Registry > Run Book > Automation).
2. Click **[Create]**. The **Automation Policy Editor** page appears.
3. Complete the following required fields:
    - ***Policy Name***. Enter a name for the automation policy.
    - ***Policy Type***. Select whether the automation policy will match events that are active, match when events are cleared, or run on a scheduled basis. Typically, you would select *Active Events* in this field.
    - **Policy State**. Specifies whether the policy will be evaluated against the events in the system. If you want this policy to begin matching events immediately, select *Enabled*.
    - **Policy Priority**. Specifies whether the policy is high-priority or default priority. These options determine how the policy is queued.
    - ***Organization***. Select one or more organizations to associate with the automation policy. The automation policy will execute only for devices in the selected organizations (that also match the other criteria in the policy). To configure a policy to execute for all organizations , select *System* without specifying individual devices to align to.

- ***Aligned Action***s. This field includes the actions from the PowerPack. To add an action to the ***Aligned Actions*** field, select the action in the ***Available Actions*** field and click the right arrow (>>). To re-order the actions in the ***Aligned Actions*** field, select an action and use the up arrow or down arrow buttons to change that action's position in the sequence.

> **NOTE:** You must have at least two Aligned Actions: one that runs the automation action and one that provides the output format. The actions providing the output formats are contained in the *Datacenter Automation Utilities* PowerPack, which is a prerequisite for running automations in this PowerPack.

> **NOTE:** If you are selecting multiple collection actions that use the "Execute PowerShell Request" action type, you may want to include the "Calculate Memory Size for Each Action" automation action, found in the *Datacenter Automation Utilities* PowerPack, in your automation policy.

4. To align the policy with the *Windows Automation* device group, which is supplied in the PowerPack, do the following:
   a. In the ***Align With*** drop-down menu, select "Device Groups".
   b. In the ***Available Device Groups*** field, select, the "Windows Automation" device group, and click the right arrow (>>).
5. Optionally, supply values in the other fields on this page to refine when the automation will trigger.
6. Click **[Save]**.

> **NOTE:** You can also modify one of the automation policies included with this PowerPack. Best practice is to use the **[Save As]** option to create a new, renamed automation policy, instead of customizing the standard automation policies.
>
> **NOTE:** If you modify one of the included automation policies and save it with the original name, the customizations in that policy will be overwritten when you upgrade the PowerPack unless you remove the association between the automation policy and the PowerPack before upgrading.

Customizing an Automation Policy

To customize an automation policy:

1. Go to the **Automation Policy Manager** page (Registry > Run Book > Automation).
2. Search for the automation policy you want to edit and click the wrench icon ( ) for that policy . The **Automation Policy Editor** page appears.
3. Complete the following fields as needed:

- ***Policy Name***. Type a new name for the automation policy to avoid overwriting the default policy.
- ***Policy Type***. Select whether the automation policy will match events that are active, match when events are cleared, or run on a scheduled basis. Typically, you would select *Active Events* in this field.
- ***Policy State***. Specifies whether the policy will be evaluated against the events in the system. If you want this policy to begin matching events immediately, select *Enabled*.
- **Policy Priority**. Specifies whether the policy is high-priority or default priority. These options determine how the policy is queued.
- ***Aligned Actions***. This field includes the actions from the PowerPack. You should see "Execute Remote PowerShell Request" action in this field. To add an action to the ***Aligned Actions*** field, select the action in the ***Available Actions*** field and click the right arrow (**>>**). To re-order the actions in the ***Aligned Actions*** field, select an action and use the up arrow or down arrow buttons to change that action's position in the sequence.

> **NOTE:** You must have at least two Aligned Actions: one that runs the automation action and one that provides the output format. The actions providing the output formats are contained in the *Datacenter Automation Utilities* PowerPack, which is a prerequisite for running automations in this PowerPack.

- ***Organization***. Select the organization that will use this policy.

4. To align the policy with the *Windows Automation* device group, which is supplied in the PowerPack, do the following:
   a. In the ***Align With*** drop-down menu, select "Device Groups".
   b. In the ***Available Device Groups*** field, select, the "Windows Automation" device group, and click the right arrow (>>).
5. Optionally, supply values in the other fields on this page to refine when the automation will trigger.
6. Click **[Save]**.

> **NOTE:** You can also modify one of the automation policies included with this PowerPack. Best practice is to use the **[Save As]** option to create a new, renamed automation policy, instead of customizing the standard automation policies.
>
> **NOTE:** If you modify one of the included automation policies and save it with the original name, the customizations in that policy will be overwritten when you upgrade the PowerPack unless you remove the association between the automation policy and the PowerPack before upgrading.

6. Optionally, supply values in the other fields on the **Automation Policy Editor** page to refine when the automation will trigger.
7. Click **[Save As]**.

Removing an Automation Policy from a PowerPack

After you have customized a policy from a *PowerPack*, you might want to remove that policy from that PowerPack to prevent your changes from being overwritten if you update the PowerPack later. If you have the license key with author's privileges for a PowerPack or if you have owner/administrator privileges with your license key, you can remove content from a PowerPack.

To remove content from a PowerPack:

1. Go to the **PowerPack Manager** page (System > Manage > PowerPacks).
2. Find the PowerPack. Click its wrench icon ( ).
3. In the **PowerPack Properties** page, in the navigation bar on the left side, click **Run Book Policies**.
4. In the **Embedded Run Book Polices** pane, locate the policy you updated, and click the bomb icon ( ) for that policy. The policy will be removed from the PowerPack and will now appear in the bottom pane.

Chapter 6

Windows PowerShell User-Initiated Automations

Overview

This manual describes how to use the automation policies found in the *Windows PowerShell User-Initiated Automation* PowerPack

NOTE: This PowerPack is available with a ScienceLogic SL1 Standard solution. Contact your ScienceLogic Customer Success Manager or Customer Support to learn more.

NOTE: ScienceLogic provides this documentation for the convenience of ScienceLogic customers. Some of the configuration information contained herein pertains to third-party vendor software that is subject to change without notice to ScienceLogic. ScienceLogic makes every attempt to maintain accurate technical information and cannot be held responsible for defects or changes in third-party vendor software. There is no written or implied guarantee that information contained herein will work for all third-party variants. See the End User License Agreement (EULA) for more information.

This chapter covers the following topics:

What is the Windows PowerShell User-Initiated Automation PowerPack?

The *Windows PowerShell User-Initiated Automation* PowerPack includes automation policies that you can use to run Windows diagnostic commands from the SL1 event console , using Event Tools. This PowerPack is supplemental to the *Windows PowerShell Automations* PowerPack and is not meant for standalone use.

In addition to using the standard content, you can customize the automation policies, or you can create your own automation policies using any available automation actions.

Installing the Windows PowerShell User-Initiated Automation PowerPack

Before completing the steps in this manual, you must import and install the latest version of the *Windows PowerShell Automations* PowerPack and the *Microsoft: Windows Server* PowerPack.

NOTE: The *Windows PowerShell User-Initiated Automation* PowerPack requires SL1 version 10.1.0 or later. For details on upgrading SL1, see the appropriate SL1 Release Notes.

WARNING: You must also install the *Datacenter Automation Utilities* PowerPack, which provides the output formats for the automation actions included in this PowerPack.

TIP: By default, installing a new version of a PowerPack overwrites all content from a previous version of that PowerPack that has already been installed on the target system. You can use the ***Enable Selective PowerPack Field Protection*** setting in the **Behavior Settings** page (System > Settings > Behavior) to prevent new PowerPacks from overwriting local changes for some commonly customized fields. (For more information, see the ***System Administration*** manual.)

IMPORTANT: The minimum required MySQL version is 5.6.0.

To download and install the PowerPack:

1. Search for and download the PowerPack from the **PowerPacks** page (Product Downloads > PowerPacks & SyncPacks) at the ScienceLogic Support Site.
2. In SL1, go to the **PowerPacks** page (System > Manage > PowerPacks).
3. Click the **[Actions]** button and choose *Import PowerPack*. The **Import PowerPack** dialog box appears.
4. Click **[Browse]** and navigate to the PowerPack file from step 1.
5. Select the PowerPack file and click **[Import]**. The **PowerPack Installer** modal displays a list of the

PowerPack contents.

6. Click **[Install]**. The PowerPack is added to the **PowerPacks** page.

> **NOTE:** If you exit the **PowerPack Installer** modal without installing the imported PowerPack, the imported PowerPack will not appear in the **PowerPacks** page. However, the imported PowerPack will appear in the **Imported PowerPacks** modal. This page appears when you click the **[Actions]** menu and select *Install PowerPack*.

Standard Automation Policies

The *Windows PowerShell User-Initiated Automation* PowerPack includes standard automation policies that trigger automation actions that will run Windows diagnostic commands from the SL1 event console.

The automation policies available in this release of the PowerPack are tied to included ScienceLogic SL1 events generated by the Dynamic Applications from the *Microsoft: Windows Server* PowerPack.

The automation policies are of Policy Type, "User Initiated". This means that for an event that matches the criteria, you can run these automation policies from the **Event Console**.

For these automation policies to be visible from the Event Tools in the Event's drawer, the following three things must be true between the event and the automation policy configuration:

- ***Organization***. The organization associated with the event must match the organization configured in the automation policy. Policies in the "System" organization match all organizations.
- ***Aligned Devices***. The device for which the event is triggered must be configured as a Aligned Device in the automation policy.
- ***Aligned Event***. The event must match one of the Aligned Events configured in the automation policy.

The following table shows the automation policies, their aligned events, and the automation actions that run in response to the events.

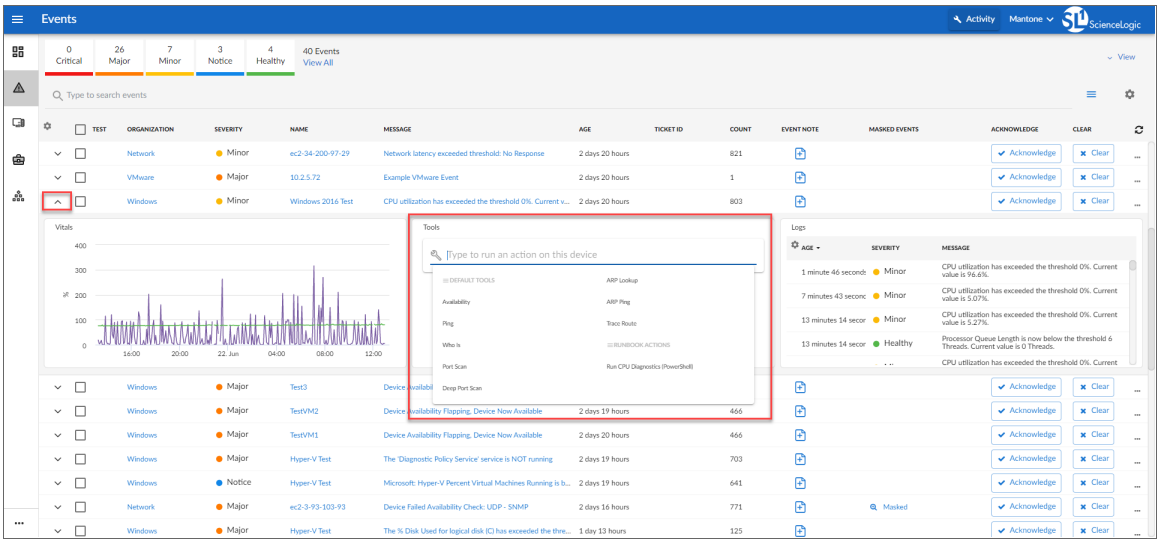
> **NOTE:** The aligned events are included as part of the *Microsoft: Windows Server* PowerPack and are not installed with the SL1 platform. You must install the PowerPack to obtain these events.

Automation Policy Name	Aligned Events	Automation Action
Run CPU & Memory Diagnostics (PowerShell)	• Microsoft: Windows Disk Transfer Time (Physical Disk) exceeded threshold	• Execute Remote PowerShell Request [101]: Windows CPU and Memory Diagnostic Commands • Datacenter Automation: Format Output as HTML
Run CPU Diagnostics (PowerShell)	• Microsoft: Windows CPU Utilization has exceeded the threshold	• Execute Remote PowerShell Request [101]: Windows CPU Diagnostic Commands

Automation Policy Name	Aligned Events	Automation Action
	• Microsoft: Windows Processor Queue Length exceeded the threshold	• Datacenter Automation: Format Output as HTML
Run Disk I/O Diagnostics (PowerShell)	• Microsoft: Windows % Disk Time (Logical Disk) exceeded threshold • Microsoft: Windows % Disk Time (Physical Disk) exceeded threshold • Microsoft: Windows Current Disk Queue Length (Physical Disk) exceeded threshold	• Execute Remote PowerShell Request [101]: Windows Disk I/O Diagnostic Commands • Datacenter Automation: Format Output as HTML
Run Memory Diagnostics (PowerShell)	• Microsoft: Windows Available Memory below threshold • Microsoft: Windows Pages per Second has exceeded threshold • Microsoft: Windows Paging File Usage has exceeded threshold	• Execute Remote PowerShell Request [101]: Windows Memory Diagnostic Commands • Datacenter Automation: Format Output as HTML
Run Print Job Error Diagnostics (PowerShell)	• Microsoft: Windows Print Job Errors exceeded threshold	• Execute Remote PowerShell Request [101]: Windows Print Job Error Diagnostic Commands • Datacenter Automation: Format Output as HTML

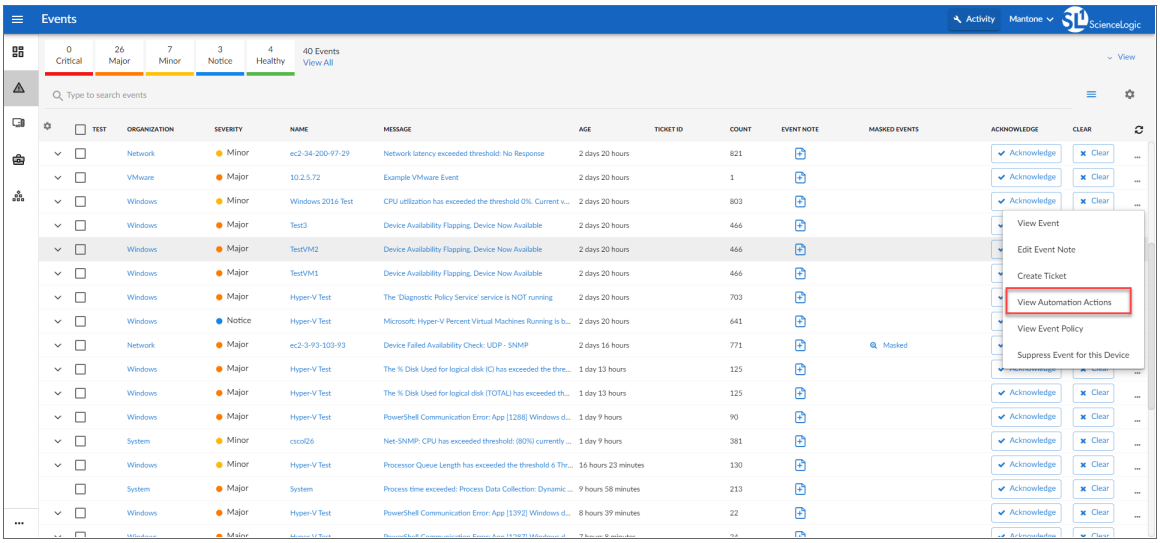
Running a User Initiated Automation Policy

To run a user initiated automation policy, open the drawer for the event and click in the Tools section. Any available user initiated automation policy will be available to run on demand.



Viewing Automation Actions for an Event

The following figure shows a VMware event with major criticality on the **Events** page. Click the **[Actions]** button (···) for an event, and select *View Automation Actions* to see the automation actions triggered by the events.



The results shown for this event, in the **Event Actions Log**, include the automation policy that ran (shown at the top of the following figure), along with the collected data. The following figure shows an example of this output.

Event Actions Log | For Event [3162] Refresh Guide

2020-06-19 20:40:25

Automation Policy Windows PowerShell: Run CPU Diagnostic Commands action Datacenter Automation: Format Output as HTML ran Successfully
Message:Snippet (50) executed without incident
Result:{'formatted_output': 'Enrichment Command Output

```
Command: Get-Process | Sort CPU -descending | Select -first 20 | Format-Table -AutoSize
Handles NPM(K)    PM(K)    WS(K)     CPU(s)    Id SI ProcessName
------- ------    -----    -----     ------    -- -- -----------
    592     86   234868   242436 150,890.66 2016  0 MsMpEng
1811628    110  3664872  1118976 123,800.61  656  0 svchost
    895      0      124      136  63,996.13    4  0 System
    545     69   248324    84744  50,117.42 2464  0 sqlservr
    608     26   373664   364152  11,050.31 2668  0 mysqld
    378     23    30860    42524  10,569.52 3612  0 WmiPrvSE
   1064     62    20692    27928   7,909.75 1096  0 svchost
    456     20    18780    23076   6,648.56 3568  0 WmiPrvSE
    329     21    10688    10208   4,228.13 1904  0 vmtoolsd
   1134     22     7988    16804   4,081.55  588  0 lsass
    444     17    16000    16272   4,010.13  300  0 svchost
    901     38   290228   151448   3,606.55  644  0 svchost
    303     10     5580    10148   3,240.00  576  0 services
     92      8     4956     1812   2,169.42 2676  0 conhost
    535     19    51760    52796   1,696.86  736  0 svchost
     93      8     4984     1564   1,237.72 2724  0 conhost
    440     15     2032     2280     906.08  360  0 csrss
    501     18     5876    15068     695.72  672  0 svchost
    142     12     2040     3968     519.63 1440  0 svchost
    146     14  1145132     3324     469.41 1824  0 mysqld

Command: Get-Process | Select-Object Name, ID, @{Name='ThreadCount';Expression ={$_.Threads.Count}} | Sort-Object -Property ThreadCount -Descending | Select -first 20
Name         Id ThreadCount
----         -- -----------
System        4         107
svchost     656          64
sqlservr   2464          50
mysqld     2668          42
svchost    1096          32
mysqld     1824          30
svchost     644          27
svchost     892          24
MsMpEng    2016          24
svchost     324          22
svchost     912          18
powershell 1112          18
svchost     300          16
svchost     672          14
```

NOTE: To learn more about which logs are collected by default for a given automation action, see the *Customizing Windows PowerShell Actions* section.

TIP: Although you can edit the automation policy described in this section, it is a best practice to use "Save As" to create a new automation policy, rather than to customize the standard automation policies.

Appendix

A

Run Book Variables

Overview

This appendix defines the different variables you can use when creating an action policy.

Use the following menu options to navigate the SL1 user interface:

- To view a pop-out list of menu options, click the menu icon ( ).
- To view a page containing all of the menu options, click the Advanced menu icon (•••).

This appendix covers the following topics:

This chapter covers the following topics:

Run Book Variables

You can include variables when creating an action policy. These variables are listed in the table below.

- In an action policy of type ***Send an Email Notification***, you can include one or more of these variables in the fields ***Email Subject*** and ***Email Body***.
- In an action policy of type ***Send an SNMP Trap***, you can include one or more of these variables in the ***Trap OID*** field, ***Varbind OID*** field, and the ***Varbind Value*** field.
- In an action policy of type ***Create a New Ticket***, you can include one or more of these variables in the ***Description*** field or the ***Note*** field of the related Ticket Template.
- In an action policy of type ***Send an SNMP Set***, you can include one or more of these variables in the ***SNMP OID*** field and the ***SNMP Value*** field.
- In an action policy of type ***Run A Snippet***, you can access variables from the global dictionary ***EM7_VALUES***.
- In a policy of type ***Execute an SQL Query***, you can include one or more of these variables in the ***SQL Query*** field.

Variable	Source	Description
%A	Account	Username
%N	Action	Automation action name
%g	Asset	Asset serial
%h	Asset	Device ID associated with the asset
%i (lowercase "eye")	Asset	Asset Location
%k	Asset	Asset Room
%K	Asset	Asset Floor
%P	Asset	Asset plate
%p	Asset	Asset panel
%q	Asset	Asset zone
%Q	Asset	Asset punch
%U	Asset	Asset rack
%u	Asset	Asset shelf
%v	Asset	Asset tag
%w	Asset	Asset model
%W	Asset	Asset make
%m	Automation	Automation policy note
%n	Automation	Automation policy name
%F	Dynamic Alert	Alert ID for a Dynamic Application Alert
%I (uppercase "eye")	Dynamic Alert	For events with a source of "dynamic", this variable contains the index value from SNMP. For events with a source of "syslog" or "trap", this variable contains

Variable	Source	Description
		the value that matches the ***Identifier Pattern*** field in the event definition.
%T	Dynamic Alert	Value returned by the Threshold function in a Dynamic Application Alert.
%V	Dynamic Alert	Value returned by the Result function in a Dynamic Application Alert.
%L	Dynamic Alert	Value returned by the label variable in a Dynamic Application Alert.
%a	Entity	IP address
%_category_id	Entity	Device category ID associated with the entity in the event.
%_category_ name	Entity	Device category name associated with the entity in the event.
%_class_id	Entity	Device class ID associated with the entity in the event.
%_class_name	Entity	Device class description associated with the entity in the event.
%_parent_id	Entity	For component devices, the device ID of the parent device.
%_parent_ name	Entity	For component devices, the name of the parent device.
%_root_id	Entity	For component devices, the device ID of the root device.
%_root_name	Entity	For component devices, the name of the root device.
%1 (one)	Event	Entity type. Possible values are: • 0. Organization • 1. Device • 2. Asset • 4. IP Network • 5. Interface • 6. Vendor • 7. Account • 8. Virtual Interface • 9. Device Group • 10. IT Service • 11. Ticket
%2	Event	Sub-entity type. Possible values for organizations are: • 9. News feed Possible values for devices are: • 1. CPU • 2. Disk • 3. File System • 4. Memory

Variable	Source	Description
		• 5. Swap • 6. Component • 7. Interface • 9. Process • 10. Port • 11. Service • 12. Content • 13. Email
%4	Event	Text string of the user name that cleared the event.
%5	Event	Date/time when event was deleted.
%6	Event	Date/time when event became active.
%7	Event	Event severity (1-5), for compatibility with previous versions of SL1. 1=critical, 2=major, 3=minor, 4=notify, 5=healthy. **NOTE**: When referring to an event, %7 represents severity (for previous versions of SL1). When referring to a ticket, %7 represents the subject line of an email used to create a ticket.
%c	Event	Event counter
%d	Event	Date/time when last event occurred.
%D	Event	Date/time of first event occurrence.
%e	Event	Event ID
%H	Event	URL link to event
%M	Event	Event message
%s	Event	severity (0 - 4). 0=healthy, 1=notify, 2=minor, 3=major, 4=critical.
%S	Event	Severity (HEALTHY - CRITICAL)
%_user_note	Event	Current note about the event that is displayed on the **Events** page.
%x	Event	Entity ID
%X	Event	Entity name
%y	Event	Sub-entity ID
%Y	Event	Sub-entity name
%Z	Event	Event source (Syslog - Group)
%z	Event	Event source (1 - 8)
%_ext_ticket_ref	Event	For events associated with an external Ticket ID, this variable contains the external Ticket ID.
%3	Event Policy	Event policy ID
%E	Event Policy	External ID from event policy

Variable	Source	Description
%f	Event Policy	Specifies whether event is stateful, that is, has an associated event that will clear the current event. 1 (one)=stateful; 0 (zero)=not stateful.
%G	Event Policy	External Category
%R	Event Policy	Event policy cause/action text
%_event_policy_name	Event Policy	Name of the event policy that triggered the event.
%B	Organization	Organization billing ID
%b	Organization	Impacted organization
%C	Organization	Organization CRM ID
%o (lowercase "oh")	Organization	Organization ID
%O (uppercase "oh")	Organization	Organization name
%r	System	Unique ID / name for the current SL1 system
%7	Ticket	Subject of email used to create a ticket. If you specify this variable in a ticket template, SL1 will use the subject line of the email in the ticket description or note text when SL1 creates the ticket. **NOTE**: When referring to a ticket, %7 represents the subject line of an Email used to create a ticket. When referring to an event, %7 represents severity (for previous versions of SL1).
%t	Ticket	Ticket ID
%J	Ticket	Description field from the SL1 ticket.

Appendix B

Configuring Windows Servers for Monitoring with PowerShell

Overview

The following sections describe how to configure Windows Server 2022, 2019, 2016, 2012, or 2012 R2 for monitoring by SL1 using PowerShell:

This chapter covers the following topics:

Prerequisites

Before configuring PowerShell, ensure the following:

- Forward and Reverse DNS should be available for the target Windows server from the SL1 Data Collector. Port 53 to the domain's DNS server should thus be available.
- When using an Active Directory user account as the SL1 credential, port 88 on the Windows Domain Controller, for the Active Directory domain, should be open for Kerberos authentication.
- If encrypted communication between the SL1 Data Collector and monitored Windows servers is desired, port 5986 on the Windows server should be open for HTTPS traffic. If unencrypted communications is being used, then port 5985 on the Windows server should be opened for HTTP traffic
- If multiple domains are in use, ensure that they are mapped in the [domain_realm] section of the Kerberos krb5.conf file on the Linux operating system of the SL1 collector appliance.

Configuring PowerShell

To monitor a Windows Server using PowerShell Dynamic Applications, you must configure the Windows Server to allow remote access from SL1. To do so, you must perform the following general steps:

1. *Configure a user account* that SL1 will use to connect to the Windows Server. The user account can either be a local account or an Active Directory account.

TIP: For ease of configuration, ScienceLogic recommends using an Active Directory account that is a member of the local Administrators group on the Windows Server.

2. *Configure a Server Authentication Certificate* to encrypt communication between SL1 and the Windows Server.
3. *Configure Windows Remote Management*.
4. Optionally, *configure a Windows server as a Windows Management Proxy*.

NOTE: If you are configuring multiple Windows servers for monitoring by SL1, you can apply these settings using a Group Policy.

5. Optionally, you can *increase the number of PowerShell Dynamic Applications that can run simultaneously* against a single Windows server.

Step 1: Configuring the User Account for the ScienceLogic Platform

To enable SL1 to monitor Windows servers, you must first configure a user account on a Windows Server that SL1 can use to make PowerShell requests. You will include this user account information when creating the PowerShell credential that SL1 uses to collect data from the Windows Server.

To configure the Windows Server user account that SL1 can use to make PowerShell requests, complete one of the following options:

- **Option 1**: *Create an Active Directory Account with Administrator access*
- **Option 2**: *Create a local user account with Administrator access*
- **Option 3**: *Create a non-administrator user account*

TIP: For ease-of-configuration, ScienceLogic recommends creating an Active Directory user account.

After creating your Windows Server user account, depending on your setup and the servers you want to monitor, you might also need to configure the user account for remote PowerShell access to the following server types:

- *Microsoft Exchange Server*
- *Hyper-V Servers*

Option 1: Creating an Active Directory Account with Administrator Access

For each Windows server that you want to monitor with PowerShell or WinRM, you can create an Active Directory account that is a member of the local Administrators group on each server. For instructions, consult Microsoft's documentation. On Windows Domain Controller servers, you can use a domain account that is not in the Domain Administrators group by following the configuration instructions for *Option 3: Creating a Non-Administrator User Account*.

After creating your Active Directory account:

- If you use SL1 to monitor Microsoft Exchange Servers, you must *configure the user account for remote PowerShell access to Microsoft Exchange Server*.
- If you use SL1 to monitor Hyper-V Servers, you must *configure the user account for remote PowerShell access to the Hyper-V Servers*.
- Otherwise, ***you can skip the remainder of this section and*** *proceed to Step 3*.

Option 2: Creating a Local User Account with Administrator Access

If you have local Administrator access to the servers you want to monitor and are monitoring Windows Server 2016 or Windows Server 2012, you can alternatively create a local user account with membership in the Administrators group instead of an Active Directory account. For instructions, consult Microsoft's documentation.

WARNING: This method does not work for Windows Server 2008.

After creating your local user account with Local Administrator access:

- If you use SL1 to monitor Microsoft Exchange Servers, you must ***configure the user account for remote PowerShell access to Microsoft Exchange Server***.
- If you use SL1 to monitor Hyper-V Servers, you must ***configure the user account for remote PowerShell access to the Hyper-V Servers***.
- Otherwise, ***you can skip the remainder of this section and proceed to Step 2***.

Option 3: Creating a Non-Administrator User Account

If you do not have Local Administrator access to the servers that you want to monitor with PowerShell or WinRM, or if the monitored Windows server is a Domain Controller that will not be in the local Administrators group, then you must first create a domain user account or create a local user account on the Windows Server. For instructions, consult Microsoft's documentation.

After creating your domain user account or local user account:

- You must configure the Windows servers to allow that non-administrator user access. To do so, ***follow the steps in this section***.
- If you use SL1 to monitor Microsoft Exchange Servers, you must also ***configure the user account for remote PowerShell access to Microsoft Exchange Server***.
- If you use SL1 to monitor Hyper-V Servers, you must also ***configure the user account for remote PowerShell access to the Hyper-V Servers***.

To configure Windows Servers to allow access by your non-administrator user account:

1. Start a Windows PowerShell shell with ***Run As Administrator*** and execute the following command:

   ```
   winrm configsddl default
   ```

2. On the **Permissions for Default** window, click the **[Add]** button, and then add the non-administrator user account.
3. Select the *Allow* checkbox for the ***Read (Get, Enumerate, Subscribe)*** and ***Execute (Invoke)*** permissions for the user, and then click **[OK]**.
4. Access the Management console. To do this:
   - In Windows Server 2016 and 2012, right-click the Windows icon, click **[Computer Management]**, and then expand **[Services and Applications]**.
5. Right-click on **[WMI Control]** and then select *Properties*.
6. On the **WMI Control Properties** window, click the **[Security]** tab, and then click the **[Security]** button.
7. Click the **[Add]** button, and then add the non-administrator user or group in the **Select Users, Service Accounts, or Groups** dialog, then click **[OK]**.

8. On the **Security for Root** window, select the user o group just added, then in the **Permissions** section at the bottom of the window, select the ***Allow*** checkbox for the *Execute Methods*, *Enable Account*, and *Remote Enable* permissions.
9. Under the **Permissions** section of the **Security for Root** window, click the **[Advanced]** button.
10. In the **Advanced Security Settings** window, double-click on the user account or group you are modifying.
11. On the **Permission Entry** window, in the ***Type*** field, select *Allow*.
12. In the ***Applies to*** field, select *This namespace and subnamespaces*.
13. Select the ***Execute Methods***, ***Enable Account***, and ***Remote Enable*** permission checkboxes, and then click **[OK]** several times to exit the windows opened for setting WMI permissions.
14. Restart the WMI Service from services.msc.

NOTE: To open services.msc, press the Windows + R keys, type "services.msc", and then press Enter.

15. **If this is a member server**, go to the Management console, go to System Tools > Local Users and Groups > Groups. Right-click on ***Performance Monitor Users***, then select *Properties*.
16. **If this is on a domain controller**, go to the Server Manager, go to the **Tools** menu, and click **Active Directory Users and Computers**. Locate the **Builtin** folder. Inside the **Builtin** folder right-click ***Performance Monitor Users***, and then select *Properties*.
17. On the **Performance Monitor Users Properties** window, click the **[Add]** button.
18. In the ***Enter the object names to select*** field, type the non-administrator domain user or group name, and then click **[Check Names]**.
19. Select the user or group name from the list and then click **[OK]**.
20. In the **Performance Monitor Users Properties** window, click **[OK]**.
21. Perform steps 15-20 for the **Event Log Readers** user group and again for the **Distributed COM Users** user group, the R**emote Management Users** user group, and if it exists on the server, the **WinRMRemoteWMIUsers__** user group.
22. If you intend to use encrypted communications between the SL1 collector host and your monitored Windows servers, each Windows server must have a digital certificate installed that has "Server Authentication" as an Extended Key Usage property. You can create a self-signed certificate for WinRM by executing the following command:

```
$Cert = New-SelfSignedCertificate -CertstoreLocation
Cert:\LocalMachine\My -DnsName "myHost"
```

24. Add an HTTPS listener by executing the following command:

```
New-Item -Path WSMan:\LocalHost\Listener -Transport HTTPS -Address * -
CertificateThumbPrint $Cert.Thumbprint -Force
```

NOTE: This command should be entered on a single line.

25. Ensure that your local firewall allows inbound TCP connections on port 5986 if you are going to use encrypted communications between the SL1 collector(s) and the Windows server, or port 5985 if you will be using unencrypted communications between the two. You may have to create a new rule on Windows Firewall if one does not already exist.

Optional: Configuring the User Account for Remote PowerShell Access to Microsoft Exchange Server

If you use SL1 to monitor Microsoft Exchange Servers:

1. Follow the steps in the section *Configuring the User Account for SL1*.
2. Add the new user account to the "Server Management" Exchange security group in Active Directory.
3. The user account will then be able to connect to the relevant WinRM endpoint to use cmdlets installed with the Exchange Management Shell. For example, this will give the user account access to the cmdlet "Get-ExchangeServer".

Optional: Configuring the User Account for Remote PowerShell Access to Hyper-V Servers

To use PowerShell Dynamic Applications to monitor a Hyper-V server, you must:

- Create a user group in Active Directory
- Add the user account you will use to monitor the Hyper-V server to the group
- Set the session configuration parameters on the Hyper-V Server
- Set the group permissions on the Hyper-V Server
- Create a PowerShell credential using the new user account

Creating a User Group and Adding a User in Active Directory

To create a group in Active Directory and add a user:

1. In Active Directory, in the same DC as the Hyper-V host you want to monitor, in the OU called ***Users***, create a group. For example, we called our group ***PSSession Creators***.
2. Add a user that meets the requirements for monitoring a Windows server via PowerShell to the group. This is the user that you will specify in the PowerShell credential.

NOTE: For details on using Active Directory to perform these tasks, consult Microsoft's documentation.

Setting the Session Configuration Parameters and Group Permissions

To set the Session Configuration and the Group Permissions on the Hyper-V Server:

1. Login to the Hyper-V server.
2. Open a PowerShell session. Enter the following command:

```
Set-PSSessionConfiguration -ShowSecurityDescriptorUI -Name
Microsoft.PowerShell
```

3. When prompted, select **A**.
4. The **Permissions** dialog appears.
5. In the **Permissions** dialog, supply values in the following fields:
   - ***Group or user names***. Select the name of the group you created in Active Directory.
   - ***Permissions for group***. For ***Full Control (All Operations)***, select the *Allow* checkbox.
6. Click the **[OK]** button.

Creating a PowerShell Credential

To create a PowerShell credential using the new user account, follow the instructions in the *Creating a PowerShell Credential* section.

Optional: Configuring the User Account for Access to Windows Failover Cluster

To configure Windows Servers to allow access to your Windows Failover Cluster:

1. Start a Windows PowerShell shell with ***Run As Administrator*** and execute the following command:

```
'Grant-ClusterAccess -User <domain>\<user> -ReadOnly'
```

Step 2: Configuring a Server Authentication Certificate

ScienceLogic highly recommends that you encrypt communications between SL1 and the Windows Servers you want it to monitor.

If you have created a ***local account on the Windows Server that uses Basic Auth*** and that account will allow communication between SL1 and the Windows server, the best practice for security is to enable HTTPS to support encrypted data transfer and authentication. To do this, you must configure WinRM to listen for HTTPS requests. This is called configuring an HTTPS listener.

> **NOTE:** For details on configuring WinRM on your Windows servers to use HTTPS, see https://support.microsoft.com/en-us/help/2019527/how-to-configure-winrm-for-https.

The sections below describe how to configure a Server Authentication Certificate on the Windows Server. This is only one task included in configuring an HTTPS listener. However, not all users need to configure a Server Authentication Certificate. You can find out if your Windows computer has a digital certificate installed for Server Authentication by running `'Get-ChildItem -Path Cert:\LocalMachine\My -EKU "*Server Authentication*"'` from a PowerShell command shell.

To support encrypted data transfer and authentication between SL1 and the servers, one of the following must be true:

- Your network ***includes a Microsoft Certificate server***. In this scenario, you should work with your Microsoft administrator to get a certificate for your Windows Server instead of configuring a self-signed Server Authentication Certificate. ***You can skip this section and*** *proceed to Step 3*.
- Your network ***does not include a Microsoft Certificate server***. In this scenario, you must configure a self-signed Server Authentication Certificate on the Windows Server that you want to monitor with SL1 using one of the following methods:
    - **Option 1**: *Use the Microsoft Management Console*.
    - **Option 2**: If your Windows Server includes Windows Software Development Kit (SDK), you can *use the makecert tool*.
    - **Option 3**: If you are running PowerShell 4.0 or later, you can *use the New-SelfSignedCertificate and Export-PfxCertificate commands*.

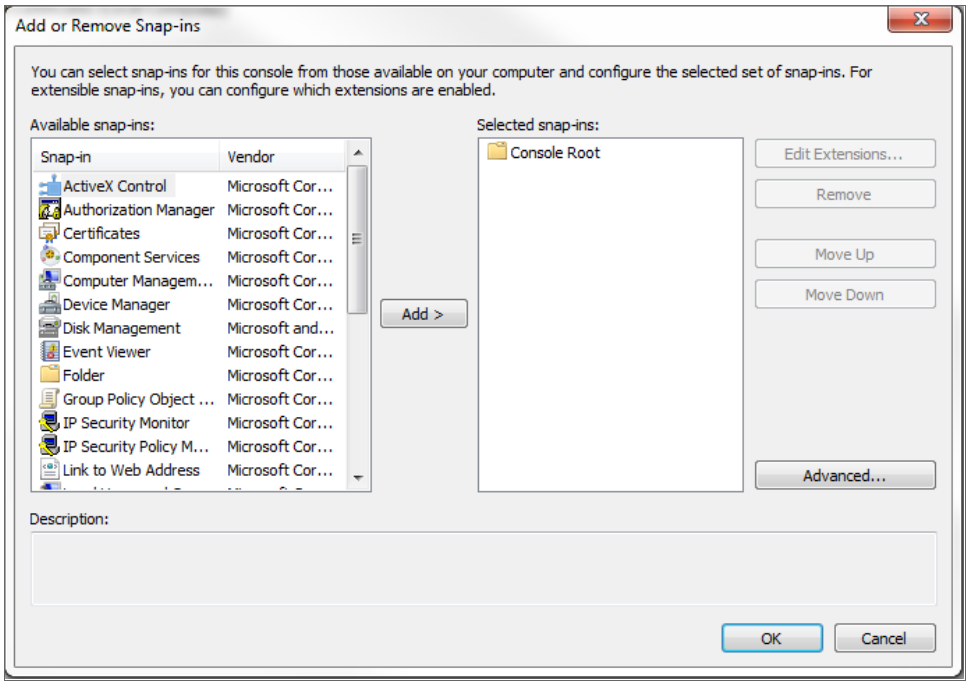
> **NOTE:** If you have created an Active Directory user account on the Windows Server to allow communication between SL1 and the server, Active Directory will use Kerberos and AES-256 encryption to ensure secure authentication.

Option 1: Using the Microsoft Management Console to Create a Self-Signed Authentication Certificate

To use the **Microsoft Management Console** to create a self-signed certificate:

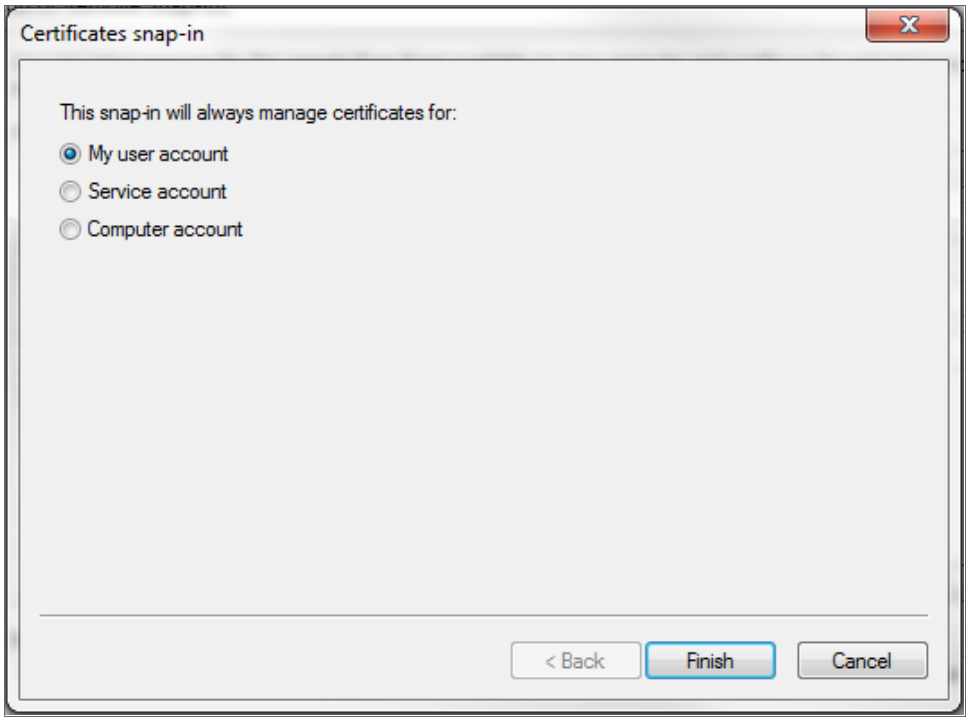
1. Log in to the Windows Server that you want to monitor with SL1.
2. In the Start menu search bar, enter "mmc" to open a **Microsoft Management Console** window.

3. Select **[File]**, then *Add/Remove Snap-Ins*. The **Add or Remove Snap-ins** window is displayed:



4. In the ***Available snap-ins*** list, select *Certificates*.

5. Click the **[Add >]** button. The **Certificates snap-in** window is displayed:



6. Select *Computer account*.
7. Click the **[Next >]** button.
8. Click the **[Finish]** button.
9. In the **Add or Remove Snap-ins** window, click the **[OK]** button.
10. In the left pane of the **Microsoft Management Console** window, navigate to Console Root > Certificates (Local Computer) > Personal.
11. Right-click in the middle pane and select *All Tasks > Request New Certificate…*. The **Certificate Enrollment** window is displayed.
12. Click the **[Next]** button. The **Select Certificate Enrollment Policy** page is displayed.
13. Select *Active Directory Enrollment Policy*.
14. Click the **[Next]** button. The **Request Certificates** page is displayed.
15. Select the ***Computer*** checkbox.
16. Click the **[Enroll]** button.
17. After the certificate is installed, click the **[Finish]** button.

Option 2: Using the MakeCert Tool to Create a Self-Signed Authentication Certificate

If your Windows system includes Windows Software Development Kit (SDK), you can use the MakeCert tool that is included in the kit to create a self-signed certificate. For information on the MakeCert tool, or for details about creating a self-signed certificate with MakeCert and installing the certificate in the Trusted Root Certificate Authorities store, see the Microsoft documentation.

Option 3: Using PowerShell Commands to Create a Self-Signed Authentication Certificate

If your Windows system includes PowerShell 4.0 or later, you can use the following PowerShell commands to create a self-signed certificate:

- You can use the **New-SelfSignCertificate** command to create a self-signed certificate. For information on **New-SelfSignCertificate**, see the Microsoft documentation.
- You can use the **Export-PfxCertificate** command to export the private certificate. For information on the **Export-PfxCertificate**, see the Microsoft documentation.

Step 3: Configuring Windows Remote Management

To provide SL1 remote access to the Windows Servers you want to monitor, you must configure Windows Remote Management.

> **NOTE:** This step is required regardless of the user account type that SL1 will use to connect to the Windows Server.

There are three ways to configure Windows Remote Management:

- **Option 1**: *Use the script provided by ScienceLogic.*
- **Option 2**: *Manually perform the configuration*.
- **Option 3**: *Use a group policy*.

Option 1: Using a Script to Configure Windows Remote Management

ScienceLogic provides a PowerShell script in a .zip file in the PowerPack download folder that automates configuration of Windows Remote Management and permissions required for the user account that will be used in the SL1 credential. The script configures all of the base Windows permissions required, except for opening up Windows Firewall ports for HTTP and/or HTTPS traffic. The configuration performed by the script is useful primarily for running collection with the **Microsoft: Windows Server**, **Microsoft: Windows Server Event Logs**, and **Microsoft: SQL Server Enhanced** PowerPacks. (Microsoft: SQL Server Enhanced requires further instance-specific permissions. See the ***Monitoring SQL Servers*** manual for more information.

To use the PowerShell script, perform the following steps:

1. When you download the *Microsoft: Windows Server* PowerPack from the ScienceLogic Support site, a .zip file for the ***WinRM Configuration Wizard Script (winrm_configuration_wizard.ps1)*** will be in the folder with the PowerPack's EM7PP file.
2. Unzip the downloaded file.
3. Using the credentials for an account that is a member of the Administrator's group, log in to the Windows server you want to monitor. You can log in directly or use Remote Desktop to log in.
4. Copy the PowerShell script named ***winrm_configuration_wizard*** to the Windows server that you want to monitor with SL1.
5. Right-click on the PowerShell icon and select ***Run As Administrator***.
6. At the PowerShell prompt, navigate to the directory where you copied the PowerShell script named ***winrm_configuration_wizard***.
7. At the PowerShell prompt, enter the following to enable execution of the script:

```
Set-ExecutionPolicy -ExecutionPolicy Unrestricted -Scope Process
-Force
```

> **NOTE**: The execution policy setting persists only during the current PowerShell session.

8. After the warning text, select Y.

> **NOTE**: If your Windows configuration requires further steps to allow execution of the script, PowerShell will display prompts. Follow the prompts.

9. To run the script with interactive dialogs, enter the following at the PowerShell prompt:

```
.\winrm_configuration_wizard.ps1 -user <domain>\<username>
```

> **NOTE:** If you have run the script previously and set HTTPS listeners, make sure you have deleted any previous HTTPS listeners with the following command: `winrm delete winrm/config/Listener?Address=*+Transport=HTTPS`

The user account you wish to use for SL1 collection must be specified with the `-user` command-line argument regardless of other arguments used. You can obtain the full help for the PowerShell configuration script by entering the following:

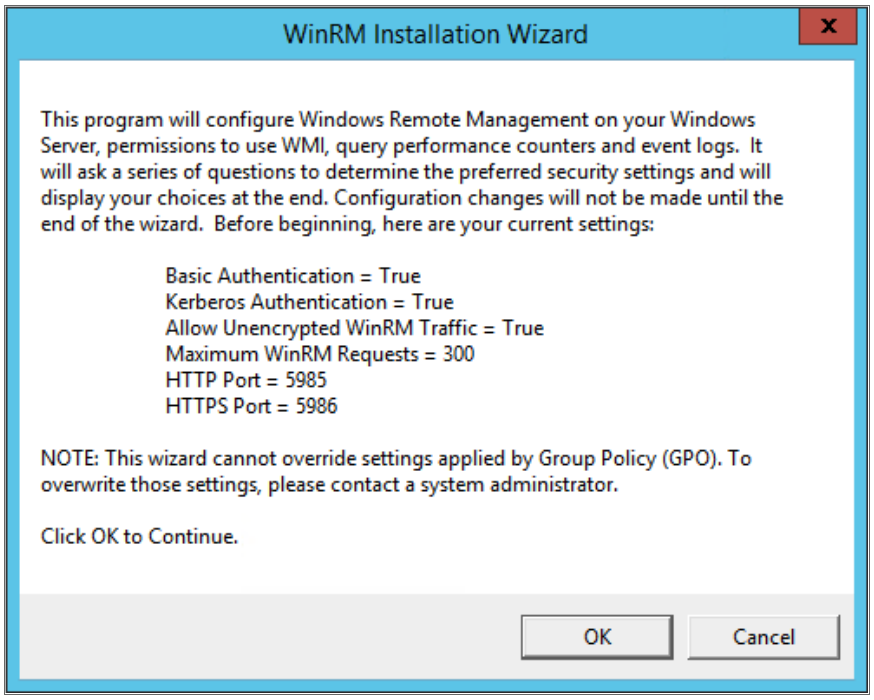
```
help .\winrm_configuration_wizard.ps1 -full
```

The most common way to run the script is silently:

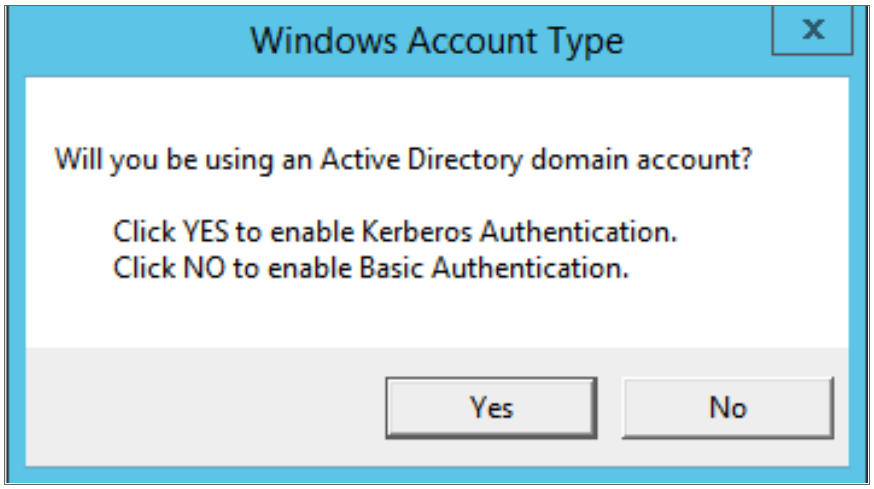
```
.\winrm_configuration_wizard.ps1 -user <domain>\<username> -
silent
```

> **NOTE**: If you have multiple certificates installed on your server, running the script with the `-silent` flag will by default use the first certificate it encounters for your HTTP/HTTPS listeners. To set a specific certificate, run the script without the `-silent` flag and use the WinRM Installation Wizard.

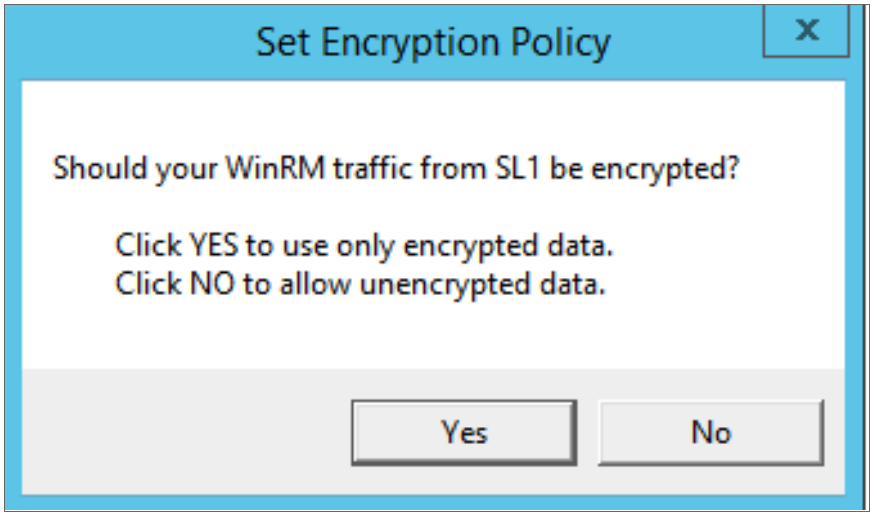
10. If you start the script without using the `-silent` command-line argument, the **WinRM Installation Wizard** modal appears. Click **[OK]**.



11. The **Windows Account Type** modal appears. Select the appropriate choice for your environment.

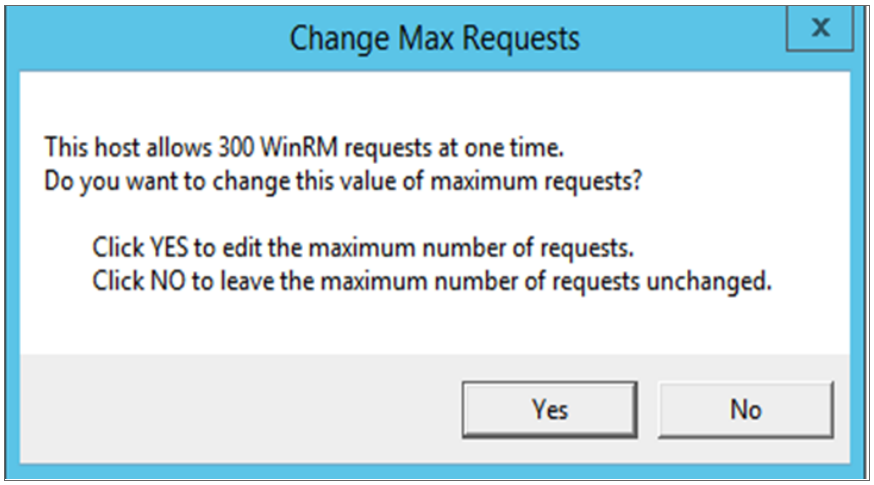


12. The **Set Encryption Policy** modal appears. Select the appropriate choice for your environment.

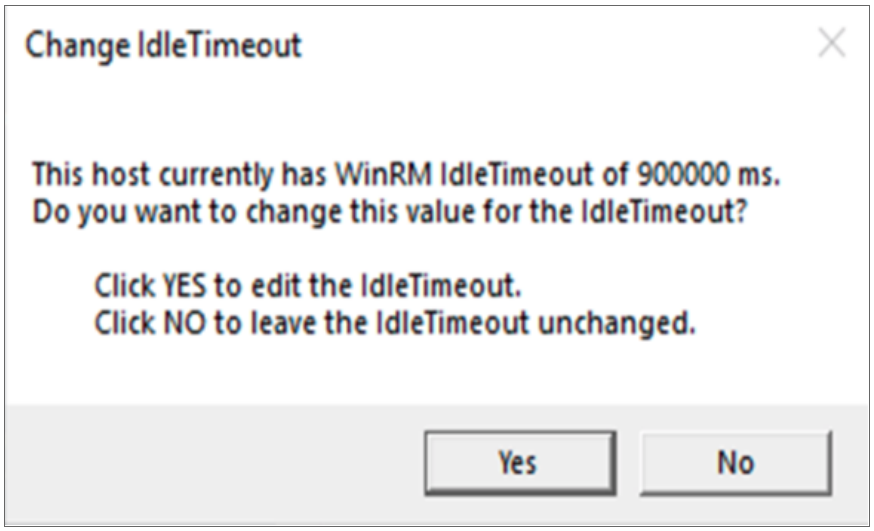


- ***Click YES to us only encrypted data***. Click Yes to configure an HTTPS listener for using encrypted communications between the SL1 collectors and the Windows server. Setting up an HTTPS listener requires a digital certificate with Server Authentication EKU to be available on the server. For information on creating a self-signed certificate, see *Configuring a Server Authentication Certificate*.
- ***Click NO to allow unencrypted data***. For communication between SL1 collectors and the Windows server, if unencrypted traffic is allowed, an HTTP listener will be configured for communication.

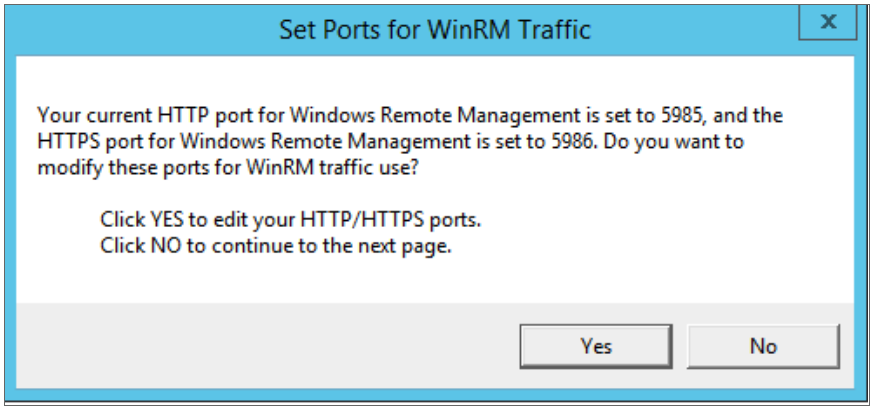
13. The **Change Max Requests** modal appears. Click **[Yes]**.



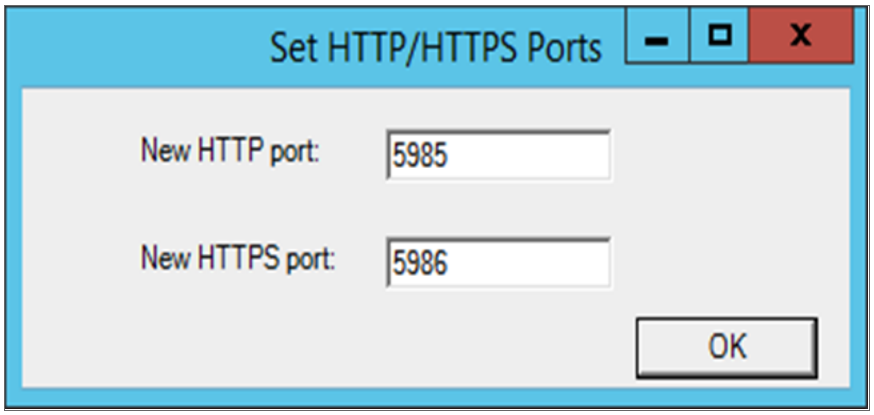
14. The **Change IdleTimeout** modal appears. If you would like to change the value of **IdleTimeout**, click **[Yes]**. If you click **[Yes]**, the **Set WinRM IdleTimeout** modal appears. Enter the new value in the field and click **[OK]**.



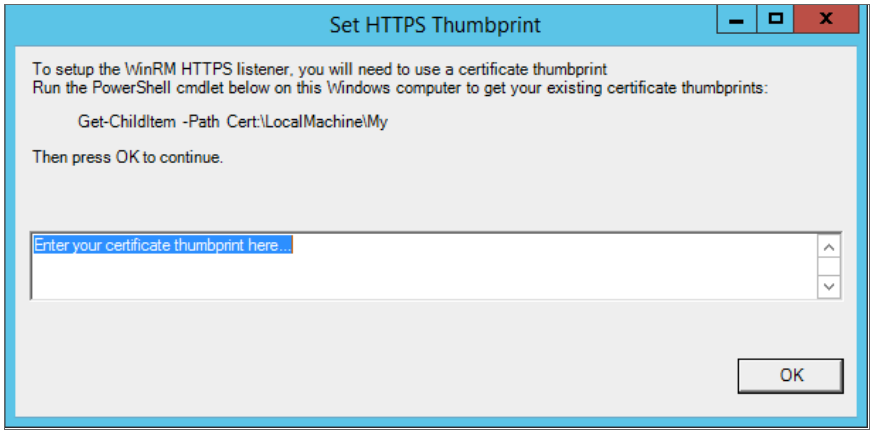
15. The **Set Ports for WinRM Traffic** modal appears, and it shows the current settings for the HTTP and HTTPS ports. If you want to make a change to these, click **[YES]**; otherwise, click **[NO]** to continue.



16. Choose which port values you would like SL1 to use when communicating with the Windows server.

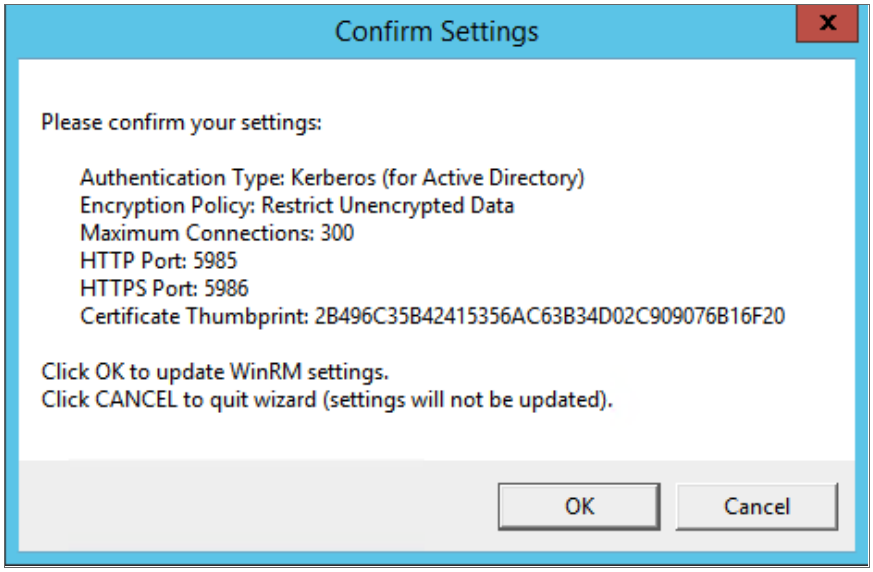


17. The **Set HTTPS Thumbprint** modal appears. Enter the information for your certificate thumbprint, which is used to create an HTTPS listener, then click **[OK]**.

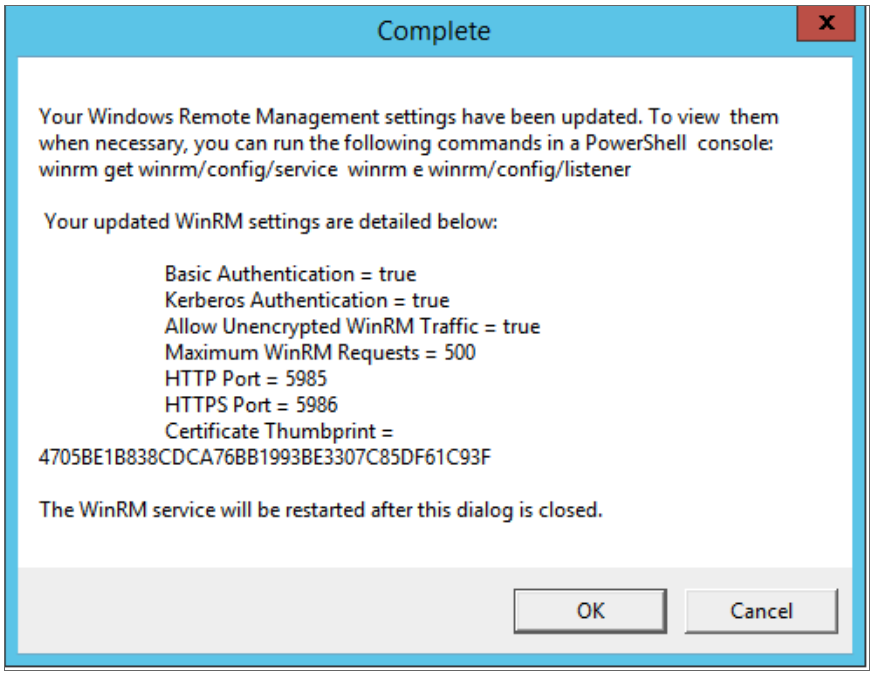


> **NOTE:** If the certificate structure for your certificate thumbprint is incomplete or incorrect, an error message appears indicating that the WinRM client cannot process the request. If you think you made an error, click **[OK]** and try to correct it. Otherwise, contact a system administrator for help.

18. The **Confirm Settings** modal appears. If the settings are as you specified, click **[OK]**.



19. The **Complete** modal appears. If the settings are correct, click **[OK]**.



20. Exit the PowerShell session.

Option 2: Manually Configuring Windows Remote Management

To configure a Windows server for monitoring via PowerShell directly, perform the following steps:

1. Log in to the server with an account that is a member of the local Administrators group, or a Domain Administrator's account if on a Windows server with the Domain Controller role installed.
2. Ensure that your local firewall allows inbound TCP connections on port 5986 if you are going to use encrypted communications between the SL1 Data Collectors and the Windows server, or port 5985 if you will be using unencrypted communications between the two. You may have to create a new rule on Windows Firewall if one does not already exist.
3. Right-click on the PowerShell icon in the taskbar or the **Start** menu, and select *Run as Administrator*.
4. Execute the following command:

```
Get-ExecutionPolicy
```

5. If the output is "Restricted", execute the following command:

```
Set-ExecutionPolicy RemoteSigned
```

6. Enter "Y" to accept.
7. Execute the following command:

```
winrm quickconfig
```

8. Enter "Y" to accept.
9. If you are configuring this Windows server for encrypted communication, execute the following command:

```
winrm quickconfig -transport:https
```

10. Enter "Y" to accept.
11. Execute the following command:

```
winrm get winrm/config
```

The output should look like this (additional lines indicated by ellipsis):

```
Config
  ...
  Client
    ...
    Auth
      Basic = true
      ...
      Kerberos = true
      ...
    ...
  Service
    ...
    AllowUnencrypted = false
    ...
    DefaultPorts
      HTTP = 5985
      HTTPS = 5986
    ...
    AllowRemoteAccess = true
  Winrs
    AllowRemoteShellAccess = true
    ...
```

12. In the Service section, if the parameter ***AllowRemoteAccess*** is set to *false*, execute the following command:

> **NOTE:** This setting does not appear for all versions of Windows. If this setting does not appear, no action is required.

```
Set-Item WSMan:\Localhost\Service\AllowRemoteAccess -value true
```

13. In the Winrs section, if the parameter ***AllowRemoteShellAccess*** is set to *false*, execute the following command:

```
Set-Item WSMan:\Localhost\Winrs\AllowRemoteShellAccess -value true
```

14. If you are configuring this Windows server for unencrypted communication and the parameter ***AllowUnencrypted*** (in the Service section) is set to *false*, execute the following command:

```
Set-Item WSMan:\Localhost\Service\AllowUnencrypted -value true
```

15. If you are configuring this Windows server for unencrypted communication, verify that "HTTP = 5985" appears in the DefaultPorts section.

> **NOTE:** ScienceLogic recommends using encrypted communication, particularly if you are also using an Active Directory account. Using an Active Directory account for encrypted authentication enables you to use Kerberos ticketing for authentication.

16. If you are configuring this Windows server for encrypted communication, verify that "HTTPS = 5986" appears in the DefaultPorts section.
16. If you are using an Active Directory account to communicate with this Windows server and in the Auth section, the parameter ***Kerberos*** is set to *false*, execute the following command:

```
Set-Item WSMan:\Localhost\Service\Auth\Kerberos -value true
```

> **NOTE:** ScienceLogic recommends using an Active Directory account.

17. If you are using a local account to communicate with this Windows server and in the Auth section, the parameter ***Basic*** is set to *false*, execute the following command:

```
Set-Item WSMan:\Localhost\Service\Auth\Basic -value true
```

18. IdleTimeout is set to 7200000 milliseconds (2 hours) by default. If an issue occurs with scheduled PowerShell monitoring and a process remains on a Windows device, it will therefore remain for up to 2 hours before being removed. To reduce the IdleTimeout and have Windows shut down idle WinRM processes after a shorter time period, execute the following command:

```
winrm s winrm/config/winrs '@{IdleTimeout="600000"}'
```

This command will change the timeout to 10 minutes (600000 ms).

> **NOTE:** When changing IdleTimeout, ensure that no other applications or utilities need a higher timeout for WinRM sessions.

Option 3: Using a Group Policy to Configure Windows Remote Management

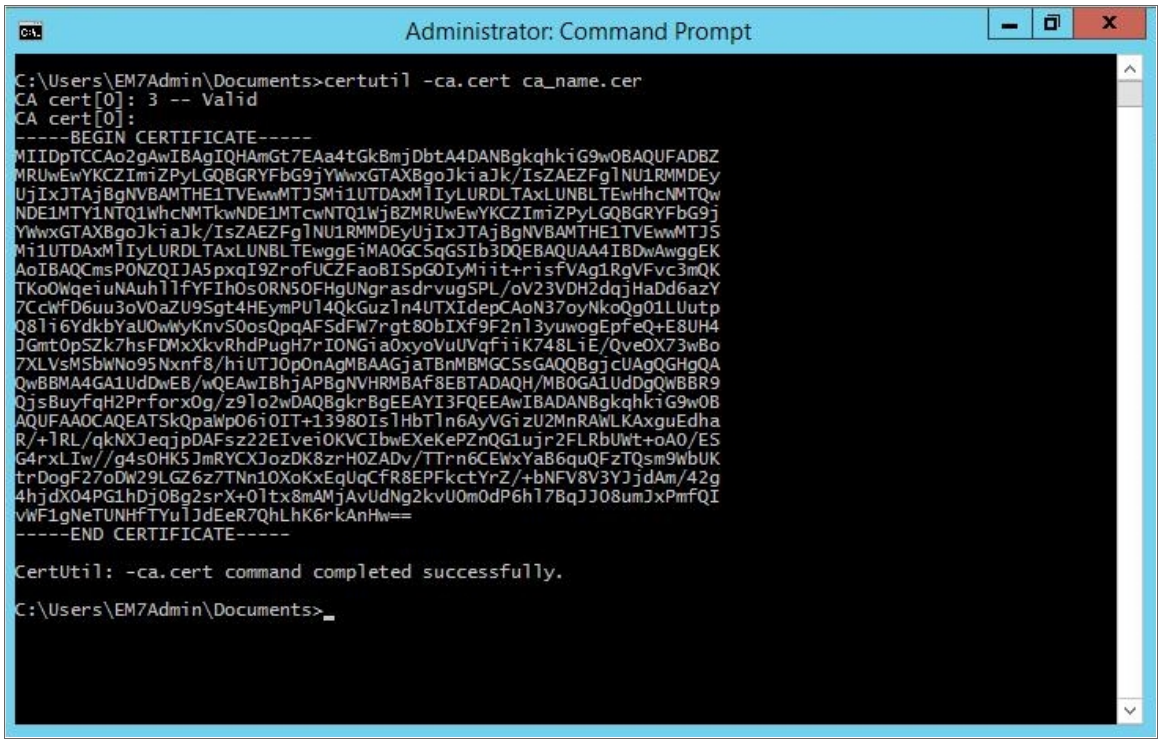
You can use a group policy object (GPO) to configure the following Windows Remote Management settings on Windows Server 2012 or Windows Server 2016:

- A registry key to enable Local Account access to Windows Remote Management
- Firewall rules
- Certificates
- HTTP and HTTPS listeners, including authentication and encryption settings
- Service start and recovery settings

To create the group policy object, perform the following steps:

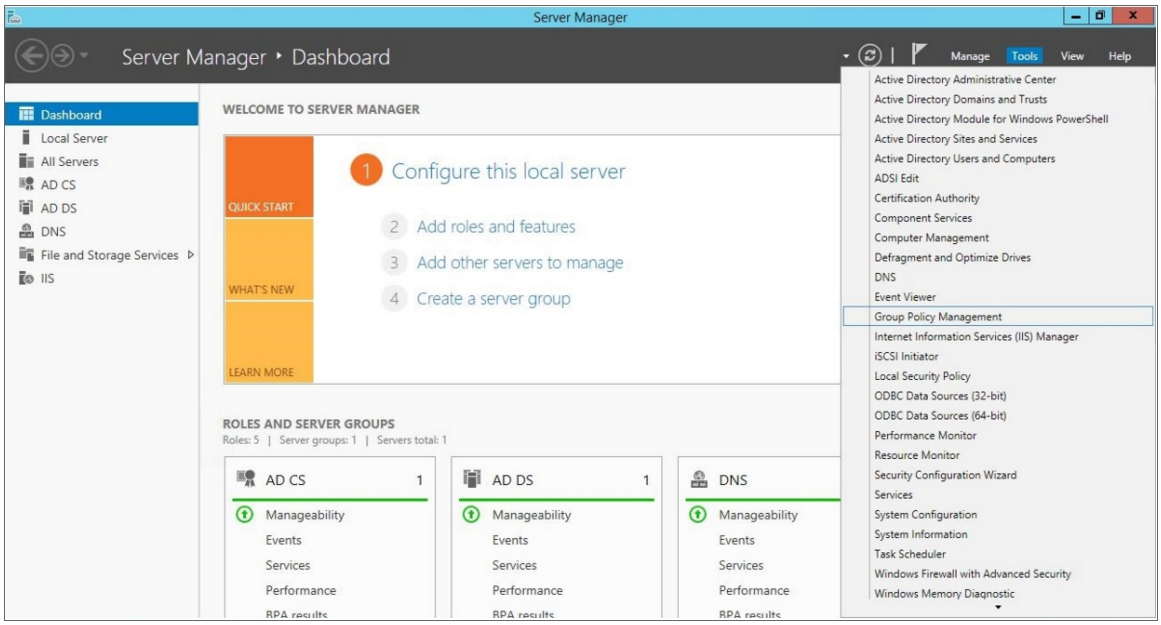
1. Log in to the CA server as an administrator.
2. Right-click on the PowerShell icon in the taskbar and select *Run as Administrator*.
3. At the PowerShell prompt, use the change directory (CD) command to navigate to a folder where you can create new files.
4. Save the root Certification Authority certificate to the local directory by executing the following command:

```
certutil.exe –ca.cert ca_name.cer
```

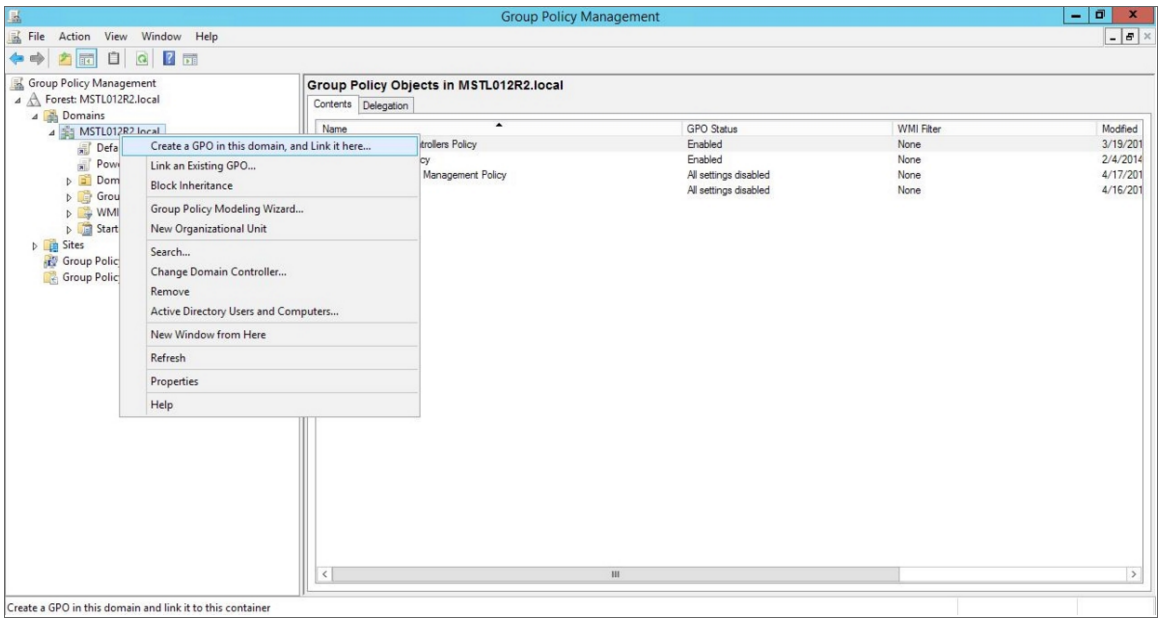


TIP: You will import this certificate into the new group policy in step 21.

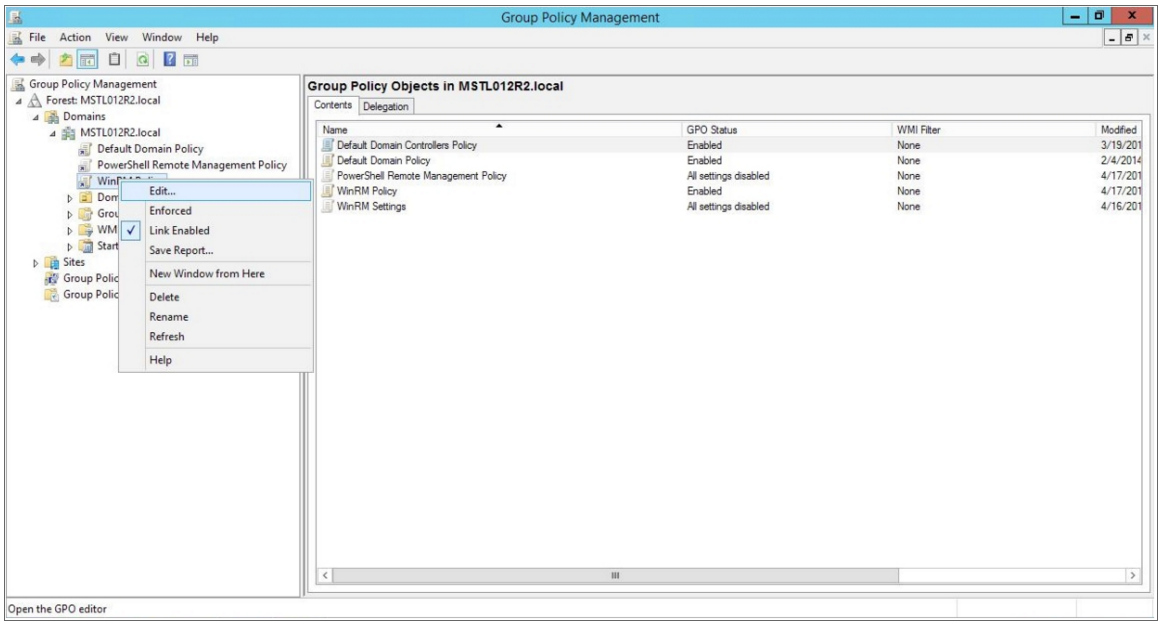
5. Exit the command prompt.
6. Log in to a domain controller in your Active Directory forest and navigate to the System Manager dashboard.
7. Click the **Tools** menu, then select *Group Policy Management*.



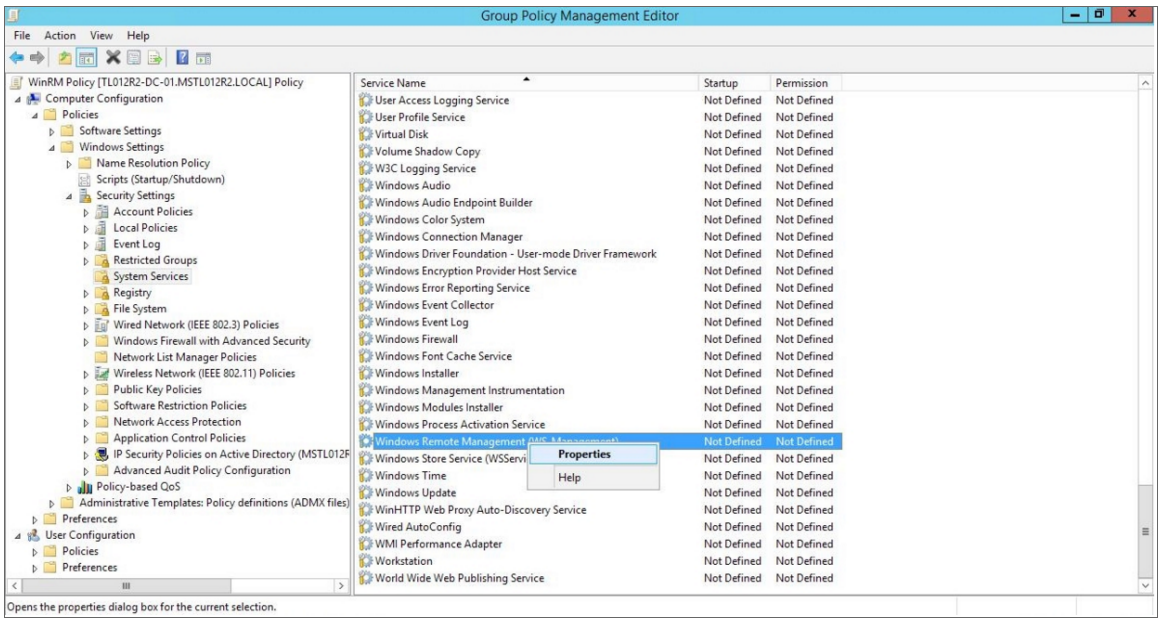
8. On the **Group Policy Management** page, in the left panel, right-click the domain name where you want the new group policy to resideand then select *Create a GPO in this domain and Link it here*.



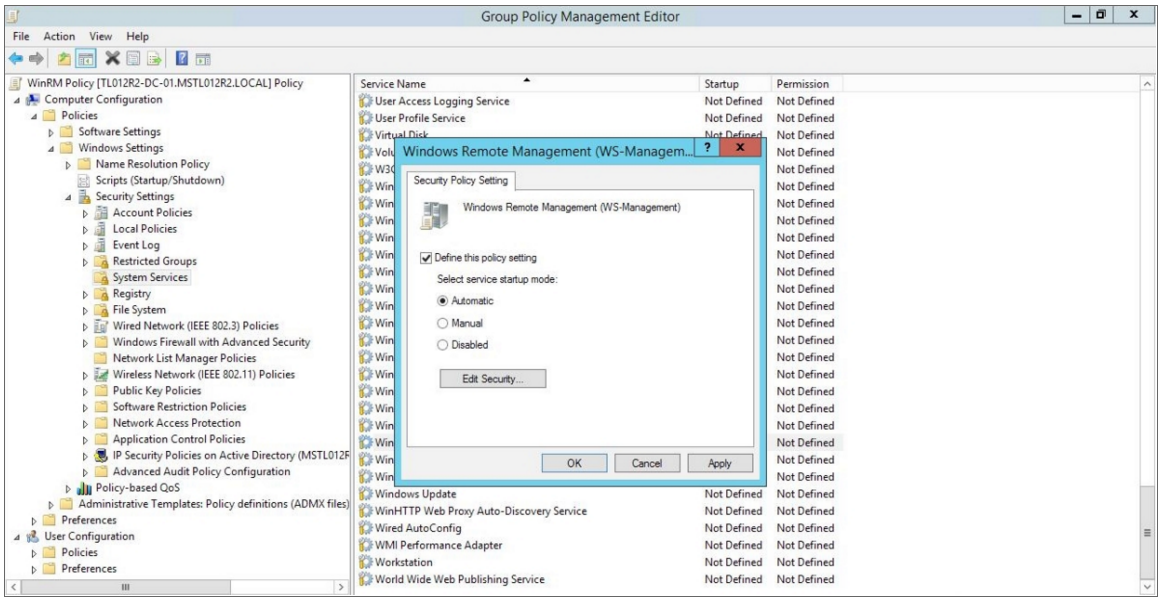
9. In the left panel, right-click the new group policy and select *Edit*. The **Group Policy Management Editor** page for the new Windows Remote Management group policy appears.



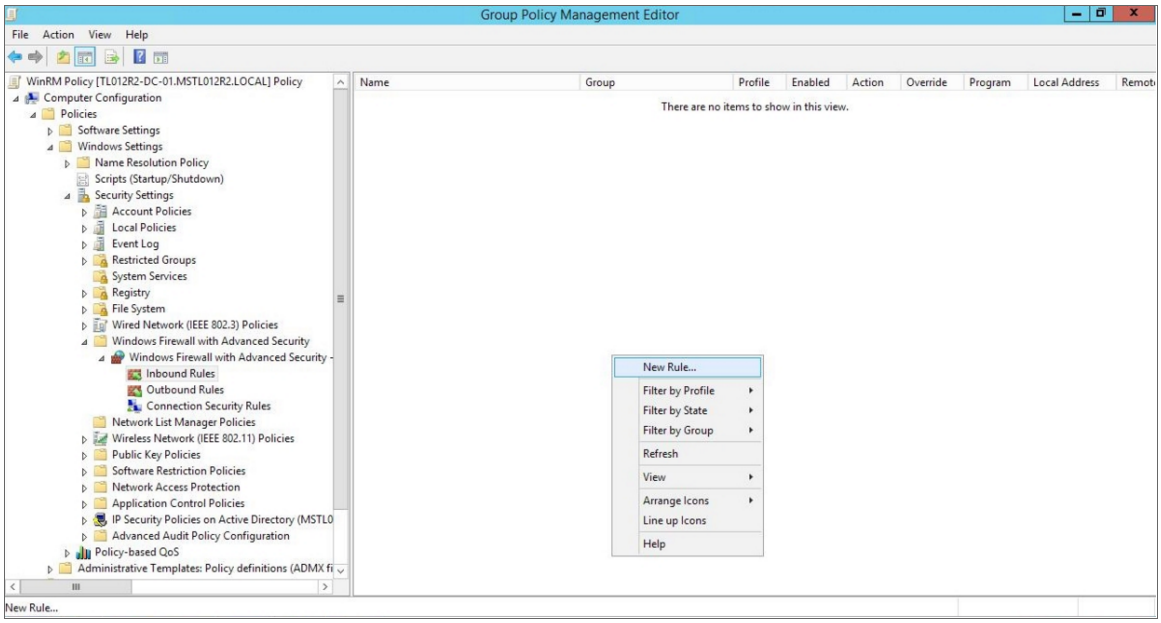
10. In the left panel, navigate to **Computer Configuration > Policies > Windows Settings > Security Settings > System Services**. In the right panel, locate the **Windows Remote Management (WS-Management)** service. Right-click the service, then select *Properties*.



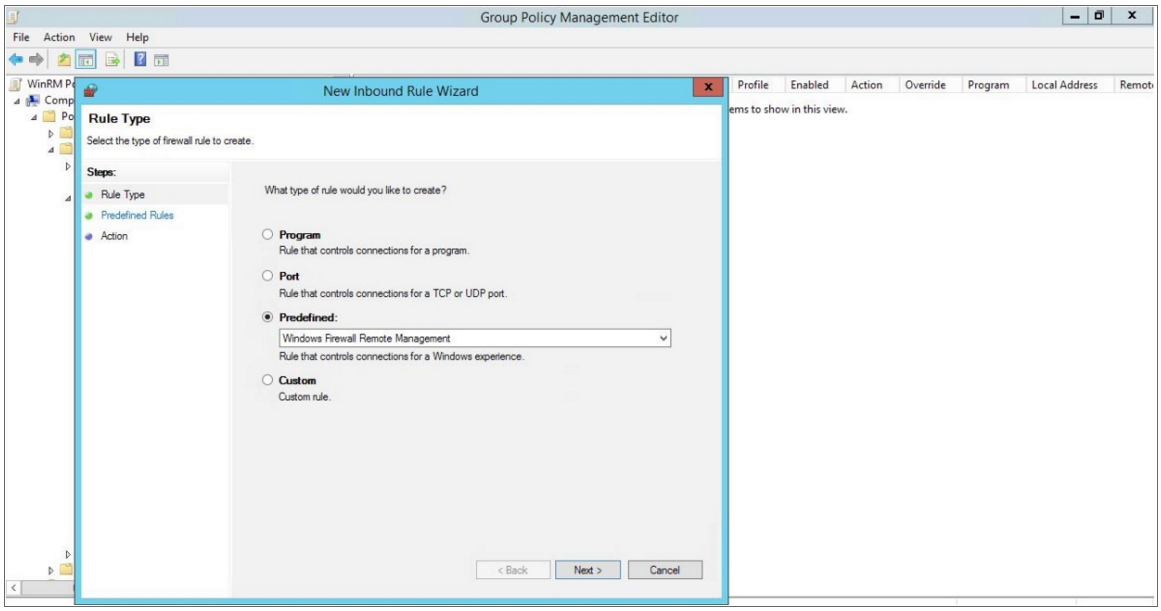
11. The **Windows Remote Management (WS-Management)** modal page appears. Select the **Define this policy setting** check box and the **Automatic** radio button, then click **[OK]**.



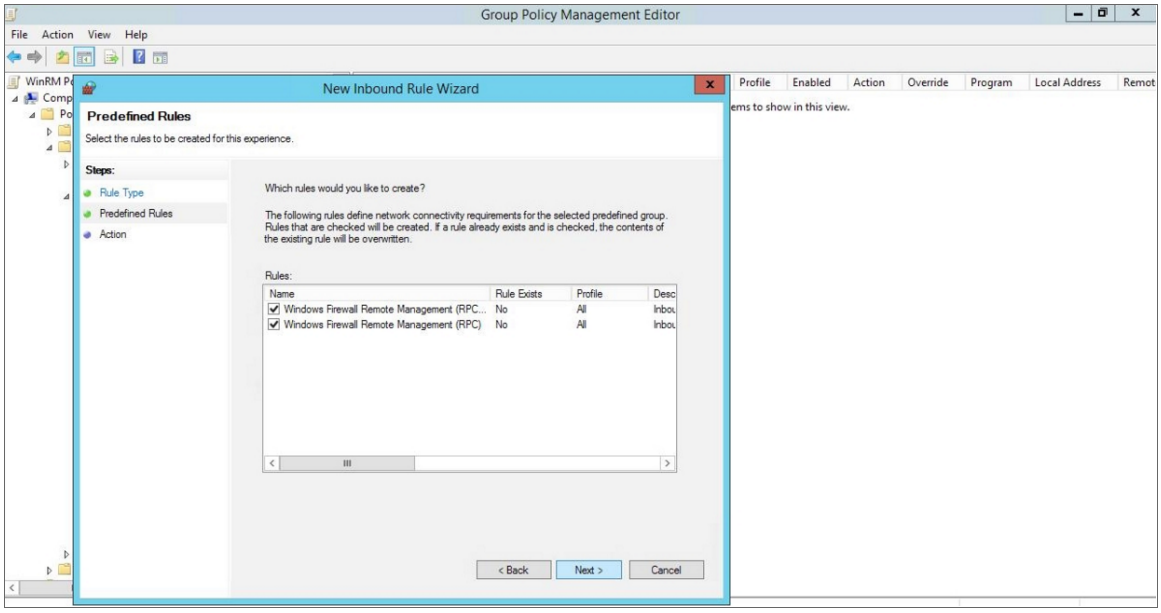
12. In the left panel of the **Group Policy Management Editor** page, navigate to **Computer Configuration > Policies > Windows Settings > Security Settings > Windows Firewall with Advanced Security > Windows Firewall with Advanced Security - LDAP > Inbound Rules**. In the right panel, right-click and select *New Rule*.



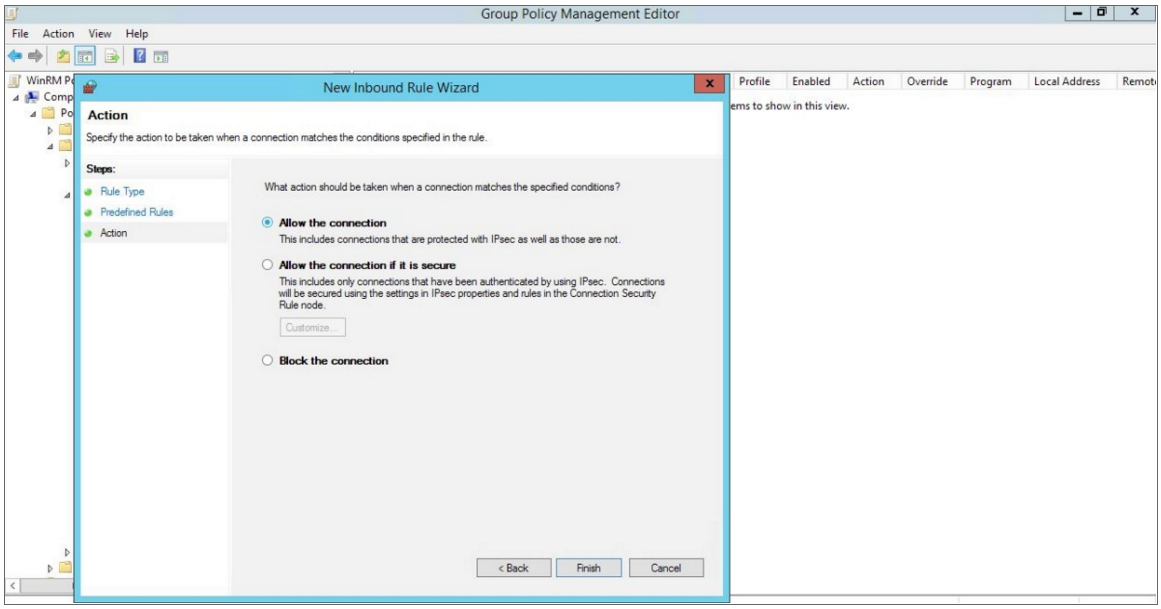
13. The **New Inbound Rule Wizard** modal page appears. Click the **Predefined** radio button, select *Windows Firewall Remote Management* from the list, and then click **[Next]**.



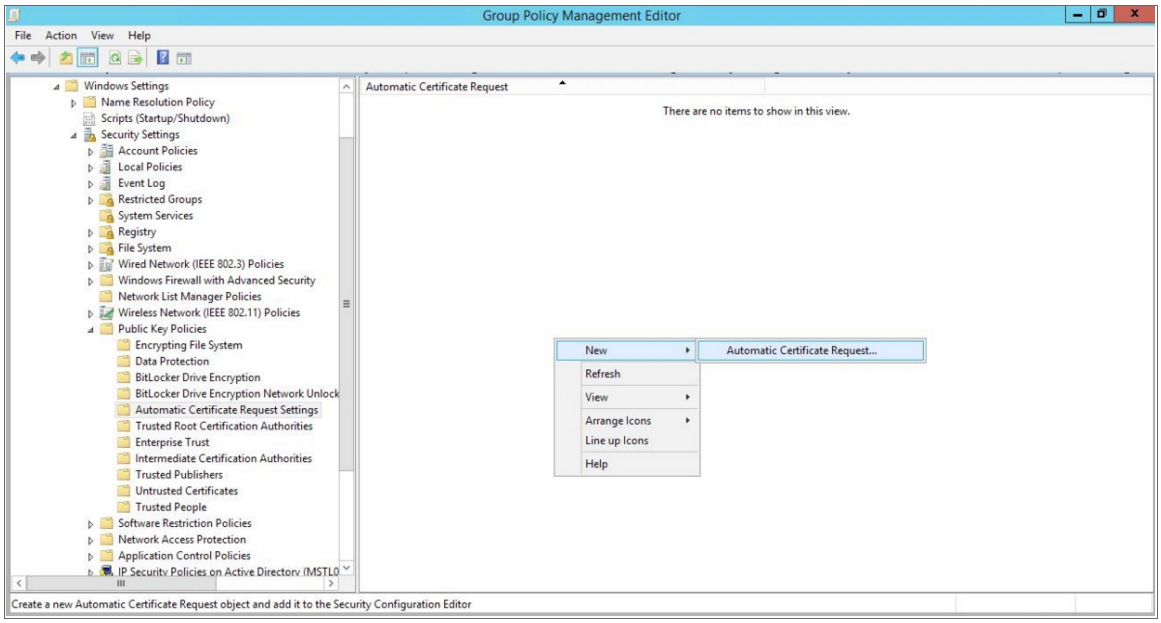
14. Select the *Windows Firewall Remote Management (RPC)* and *Windows Firewall Remote Management (RPC-EPMAP)* check boxes, then click **[Next]**.



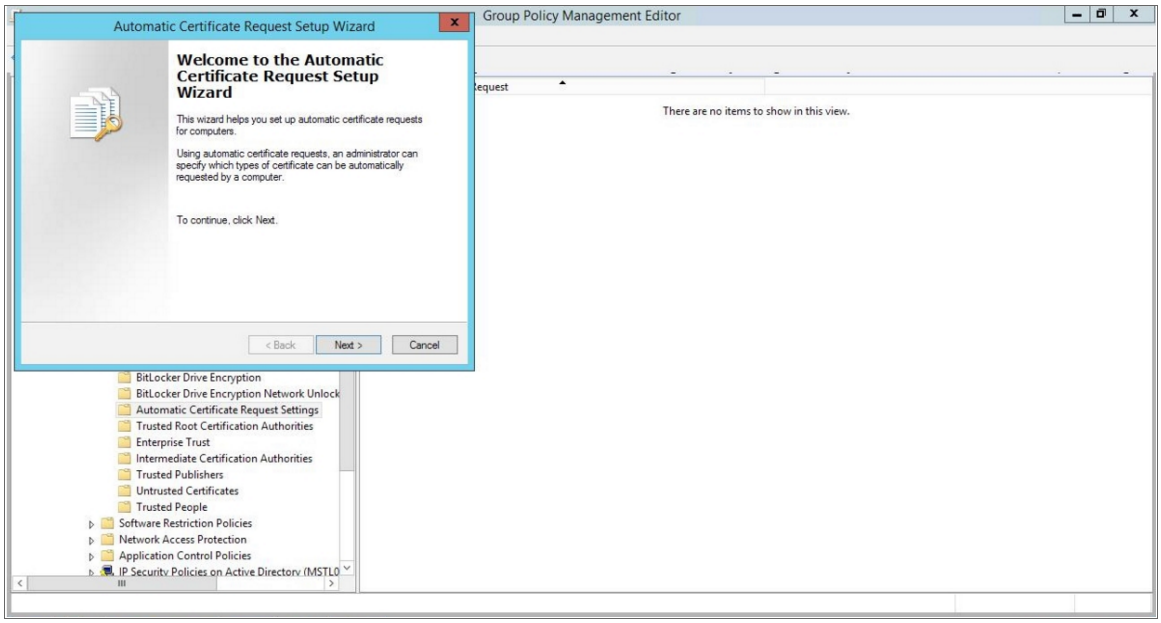
15. Select the *Allow the connection* radio button, then click **[Finish]**.



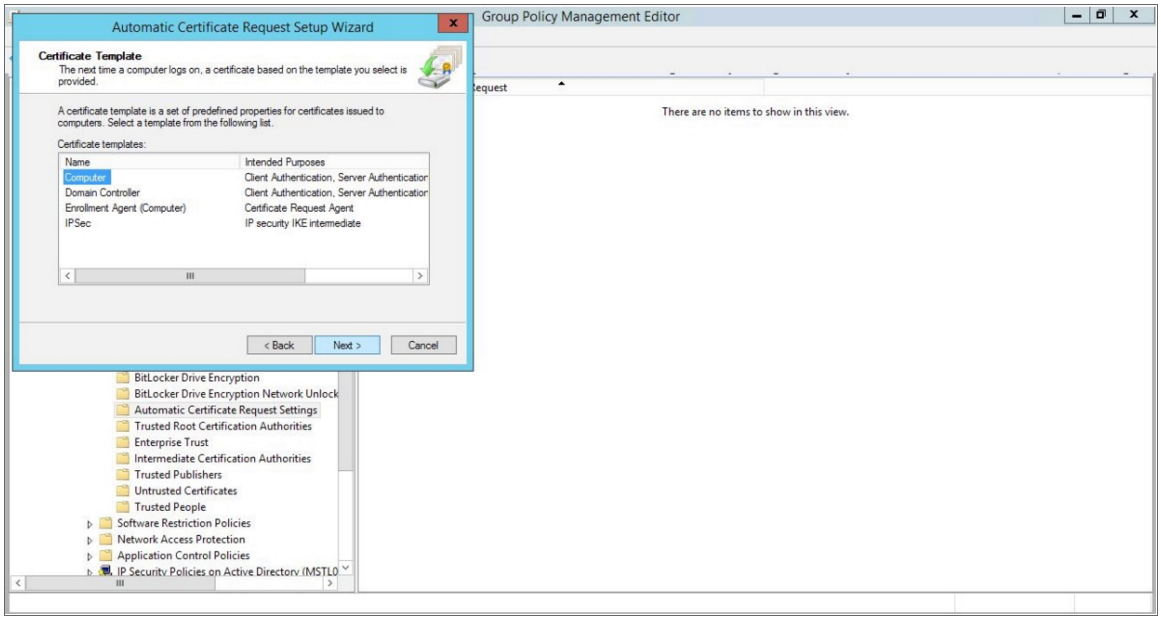
16. In the left panel of the **Group Policy Management Editor** page, navigate to **Computer Configuration > Policies > Windows Settings > Security Settings > Public Key Policies > Automatic Certificate Request Settings**. In the right panel, right-click and select *New > Automatic Certificate Request*.



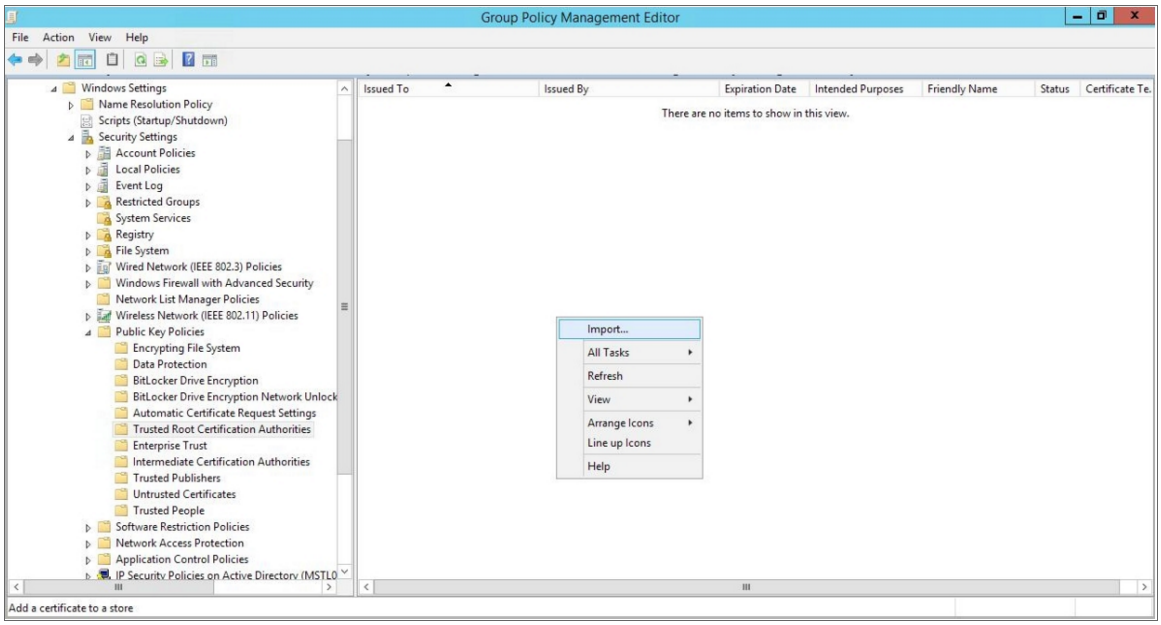
17. The **Automatic Certificate Request Setup Wizard** modal page appears. Click **[Next]**.



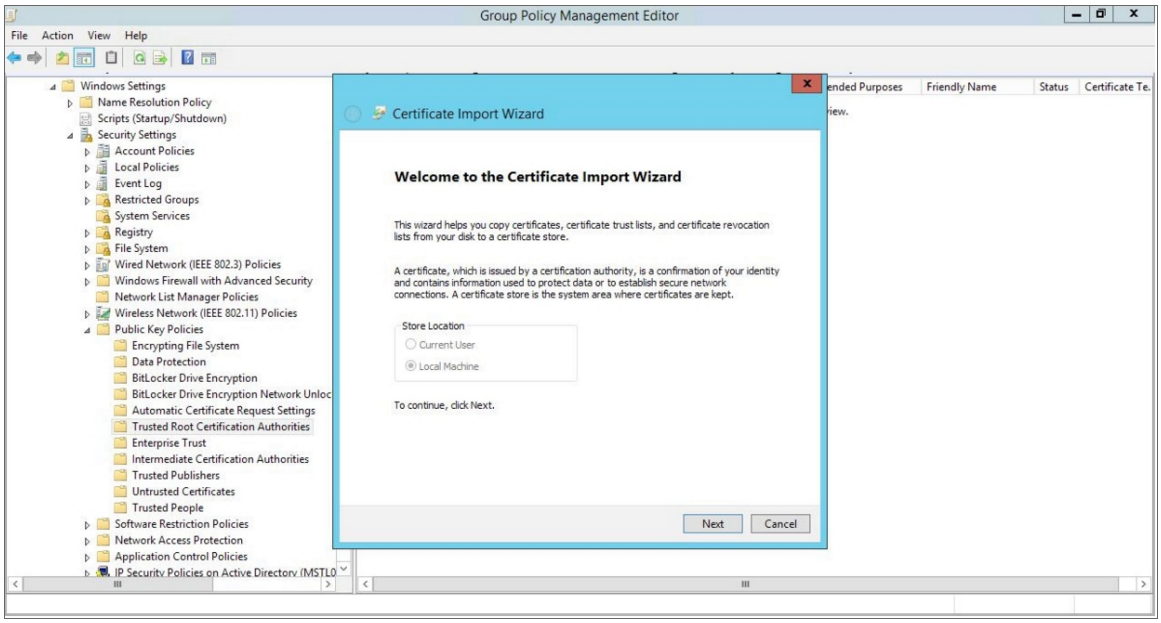
18. Select the *Computer* certificate template. Click **[Next]**, and then click **[Finish]**.



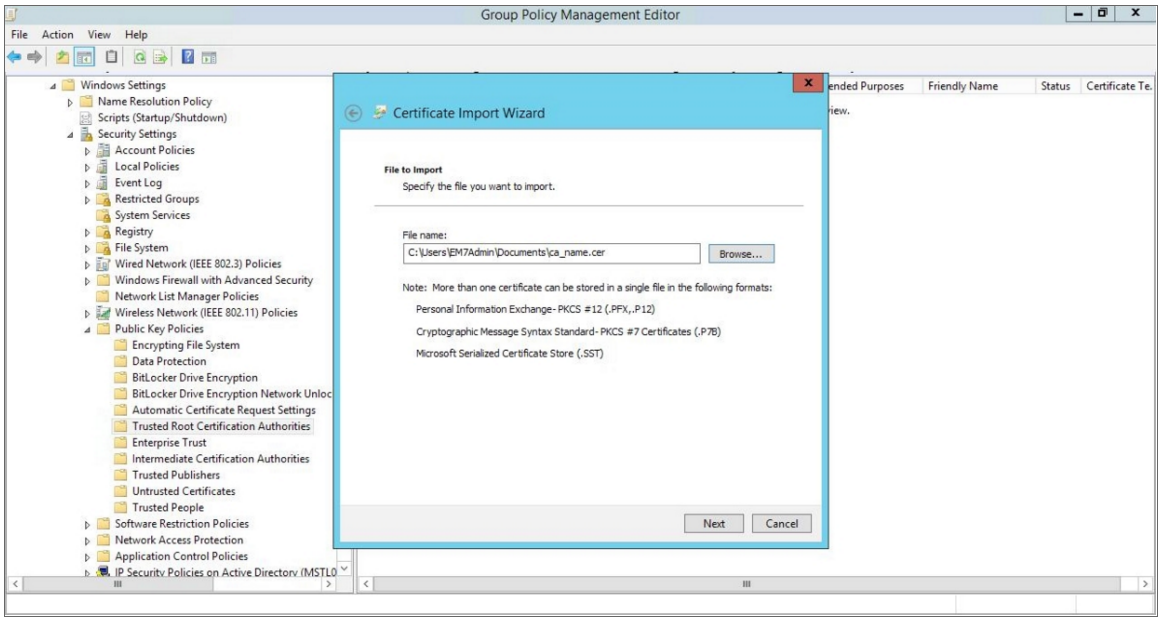
19. In the left panel of the **Group Policy Management Editor** page, navigate to **Computer Configuration > Policies > Windows Settings > Security Settings > Public Key Policies > Trusted Root Certification Authorities**. In the right panel, right-click and select *Import*.



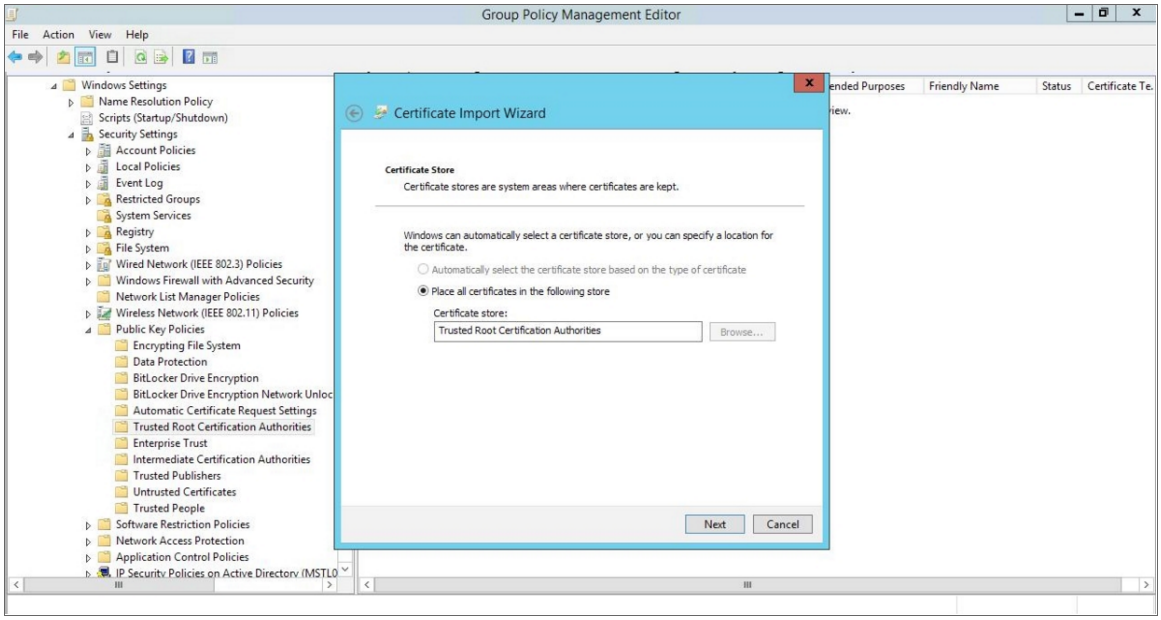
20. The **Certificate Import Wizard** modal page appears. Click **[Next]**.



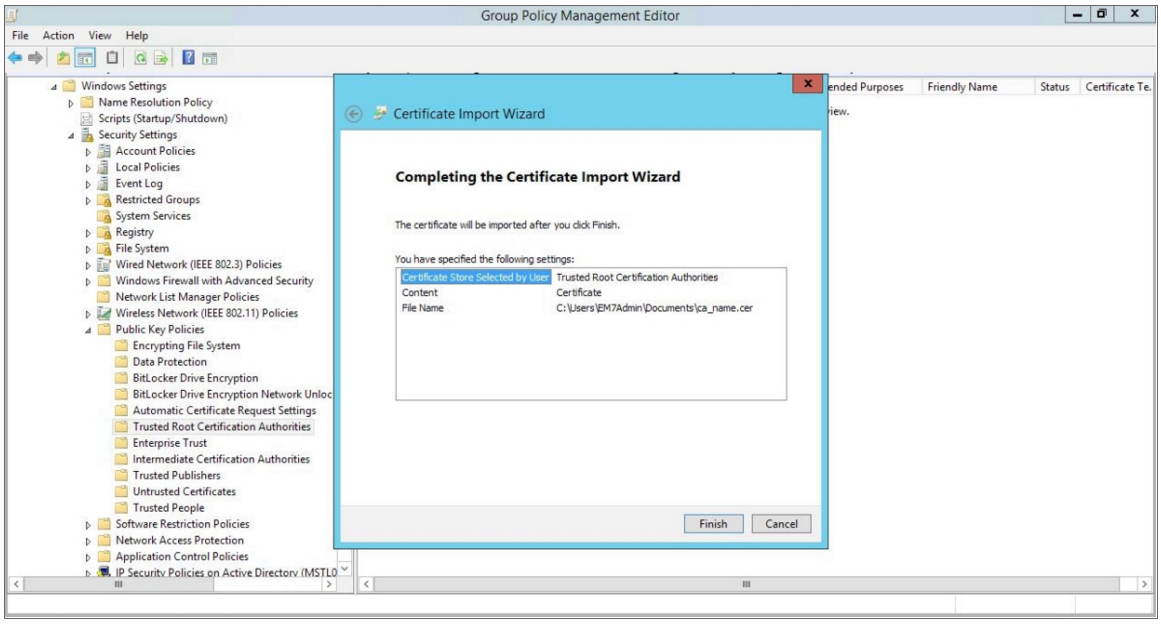
21. Browse to the Certification Authority certificate that you saved to your local directory in step 4, then click **[Next]**.



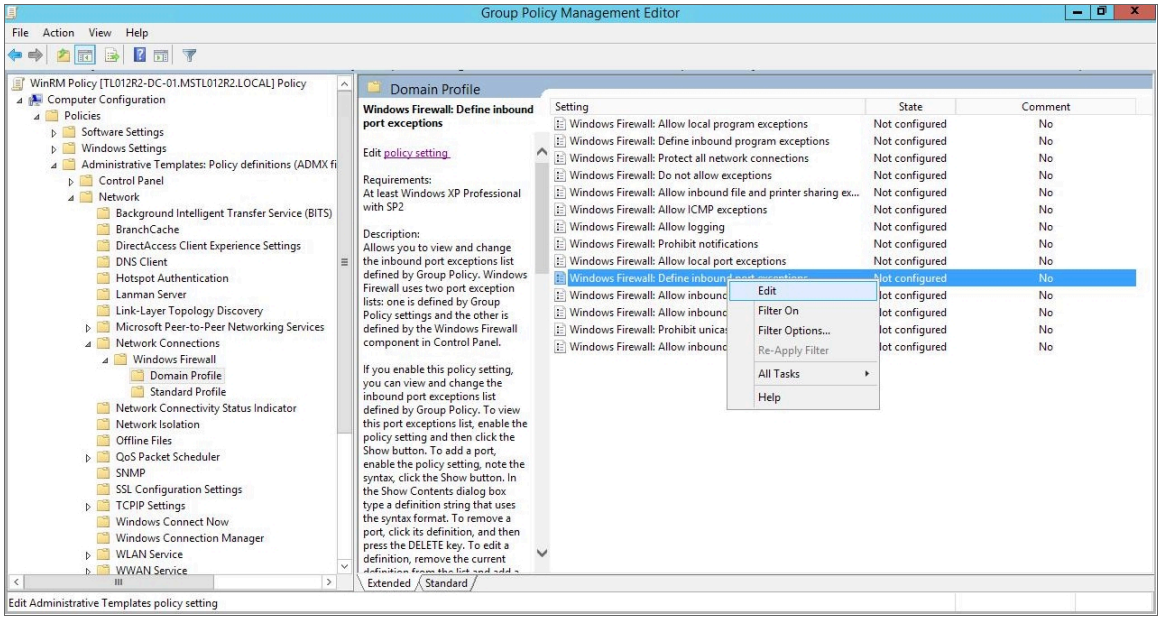
22. Select the **Place all certificates in the following store** radio button, then select the *Trusted Root Certification Authorities* certificate store and click **[Next]**.



23. Click **[OK]** to confirm that the certificate was successfully imported, and then click **[Finish]**.

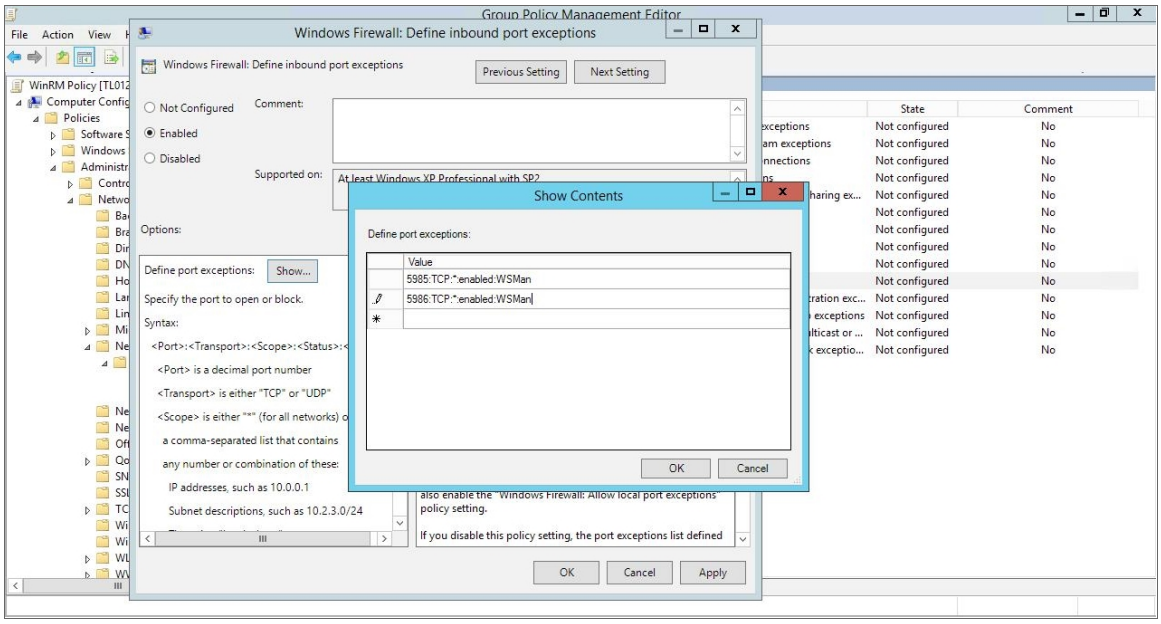


24. In the left panel of the **Group Policy Management Editor** page, navigate to **Computer Configuration > Policies > Administrative Templates > Network > Network Connections > Windows Firewall > Domain Profile**. In the right panel, right-click **Windows Firewall: Define inbound port exceptions** and select *Edit*.



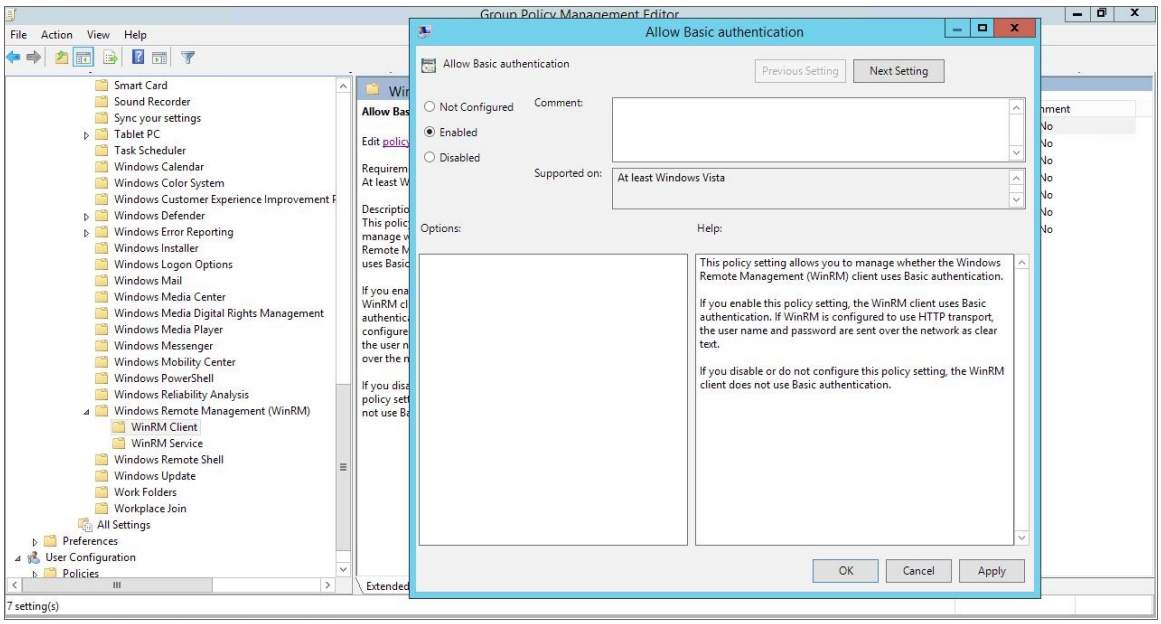
25. The **Windows Firewall: Define inbound port exceptions** modal page appears. Under **Options**, click **[Show]**.

26. The **Show Contents** modal page appears. Enter the following values:



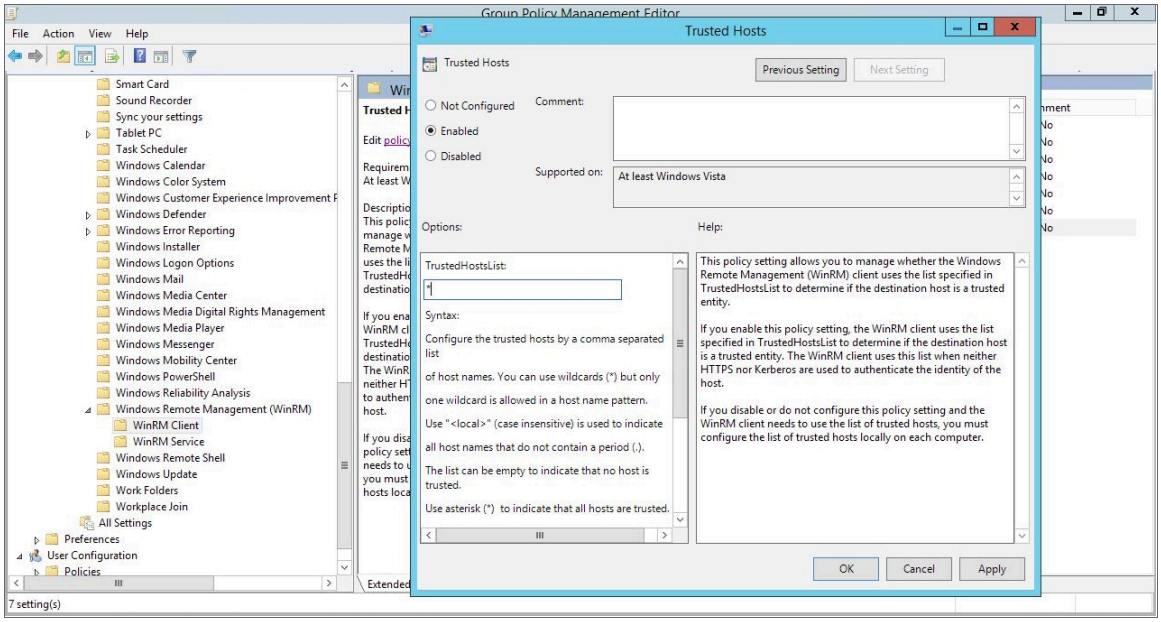
- 5985:TCP:*:enabled:WSMan
- 5986:TCP:*:enabled:WSMan

27. Click **[OK]**, then click **[OK]** again.
28. In the left panel of the **Group Policy Management Editor** page, navigate to **Computer Configuration > Policies > Administrative Templates > Windows Components > Windows Remote Management (WinRM) > WinRM Client**. In the right panel, double-click the **Allow Basic authentication** setting.

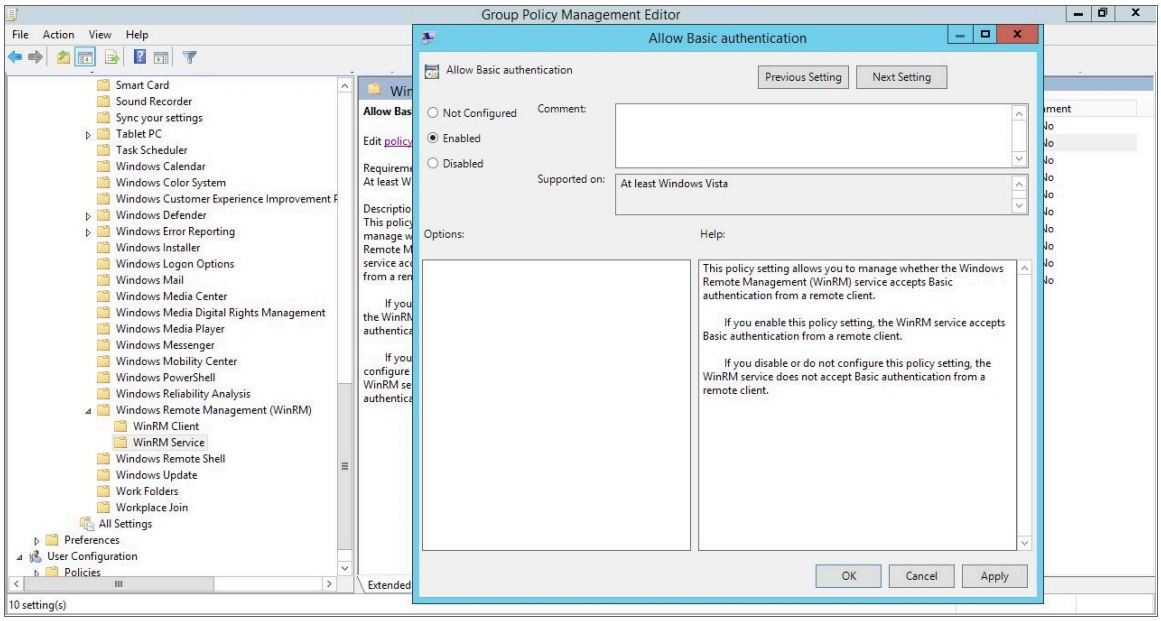


29. Select the **Enabled** radio button, then click **[OK]**.

30. Repeat steps 28 and 29 for the **Allow unencrypted traffic** setting.
31. Double-click the **Trusted Hosts** setting. Select the **Enabled** radio button, enter an asterisk (*) in the **TrustedHostsList** field (under **Options**), and then click **[OK]**.



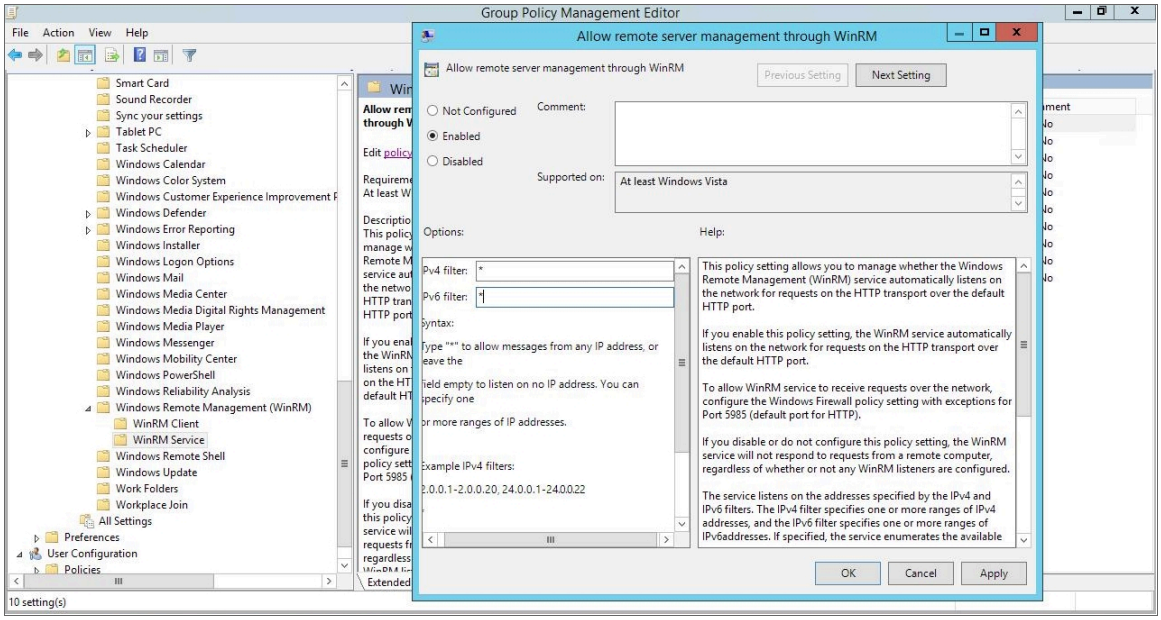
32. In the left panel of the **Group Policy Management Editor** page, navigate to **Computer Configuration > Policies > Administrative Templates > Windows Components > Windows Remote Management (WinRM) > WinRM Service**. In the right panel, double-click the **Allow Basic authentication** setting.



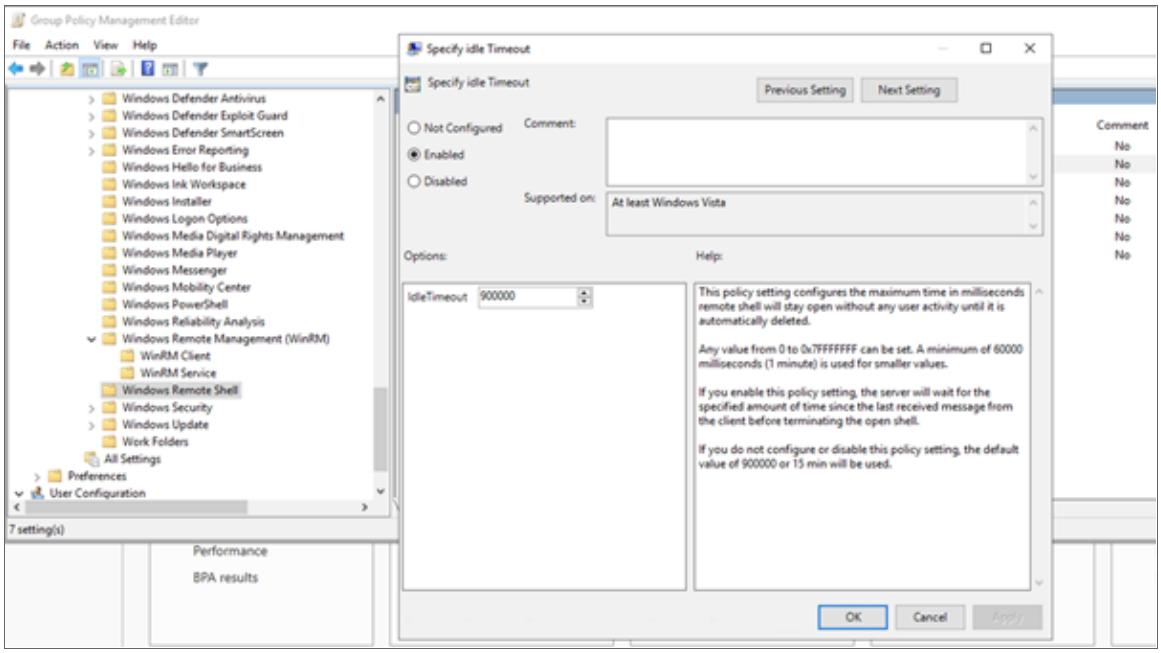
33. Select the **Enabled** radio button, then click **[OK]**.

34. Repeat steps 32 and 33 for the **Allow unencrypted traffic** setting.

35. Double-click the **Allow remote server management through WinRM** setting. Select the **Enabled** radio button, enter an asterisk (*) in the **Pv4 filter** and **Pv6 filter** fields (under **Options**), and then click **[OK]**.



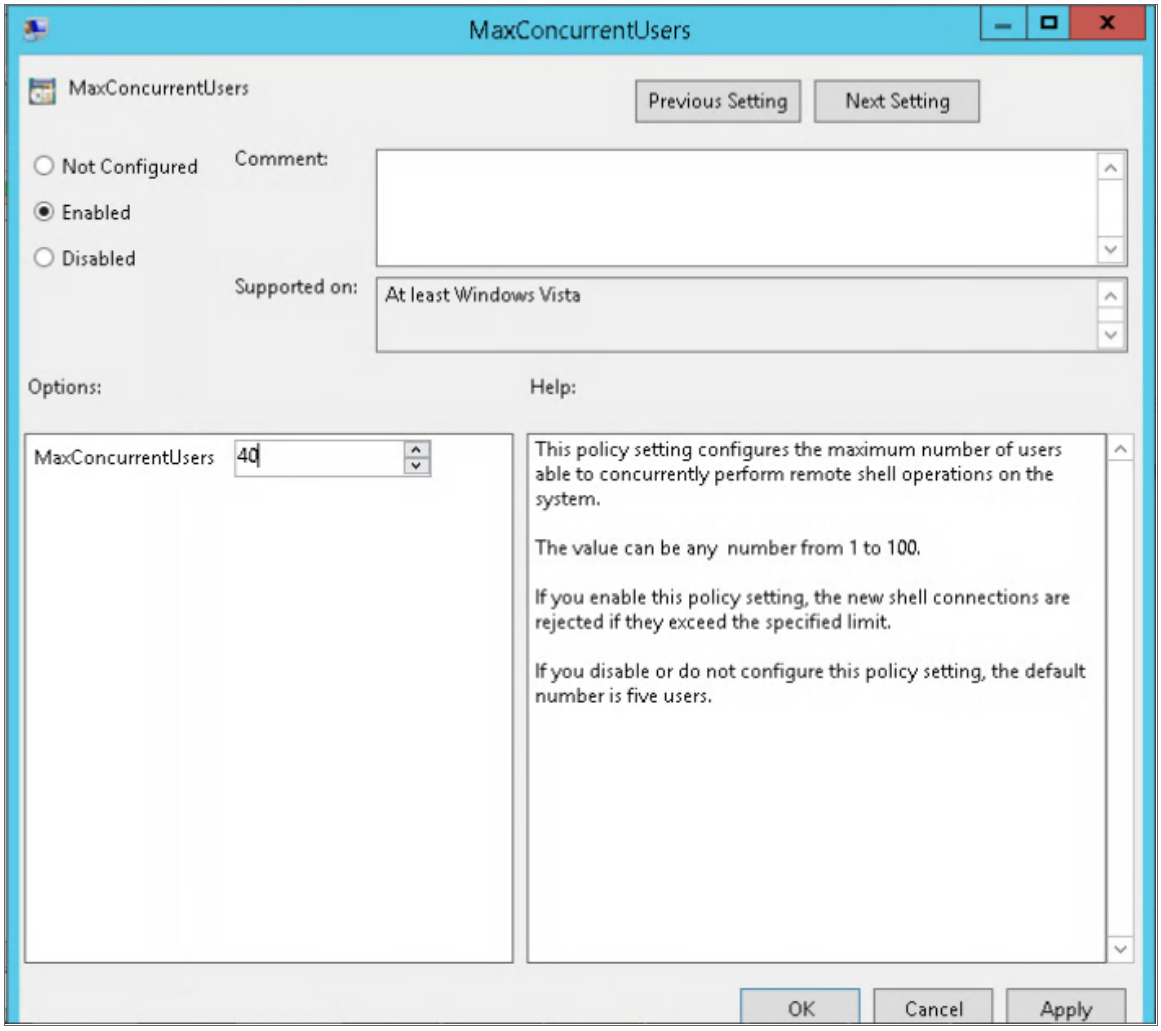
36. In the left panel of the **Group Policy Management Editor** page, navigate to **Computer Configuration > Policies > Administrative Templates: Policy Definitions > Windows Components > Windows Remote Shell**. In the right panel, double-click on **Specify Idle Timeout**:



Adjust the setting to meet your requirements. Using the value of 900000 in the image will set the timeout to 15 minutes. Once you have entered your timeout value in milliseconds, click the ***Enabled*** radio button and then click **[OK]**.

> **NOTE:** When changing IdleTimeout, ensure that no other applications or utilities need a higher timeout for WinRM sessions.

37. In the **Windows Remote Shell** folder, in the right panel, double-click on **MaxConcurrentUsers**:



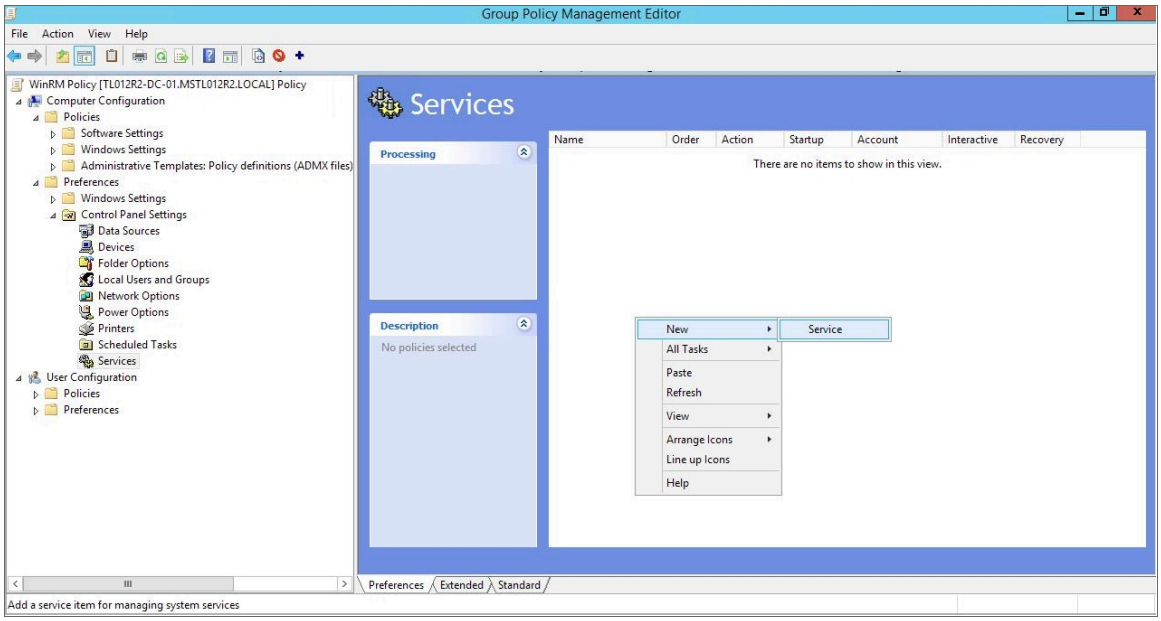
Enter "40" in the ***MaxConcurentUsers*** field. Once you have entered your value, click the ***Enabled*** radio button and then click **[OK]**.

38. **You can skip this step if you already have a group policy in place for this setting.** In the left panel of the **Group Policy Management Editor** page, navigate to **Computer Configuration > Preferences > Windows Settings > Registry**. In the right panel, right-click and select *New > Registry Item*. In the **New Registry Properties** modal page, edit the values in one or more of the following fields:

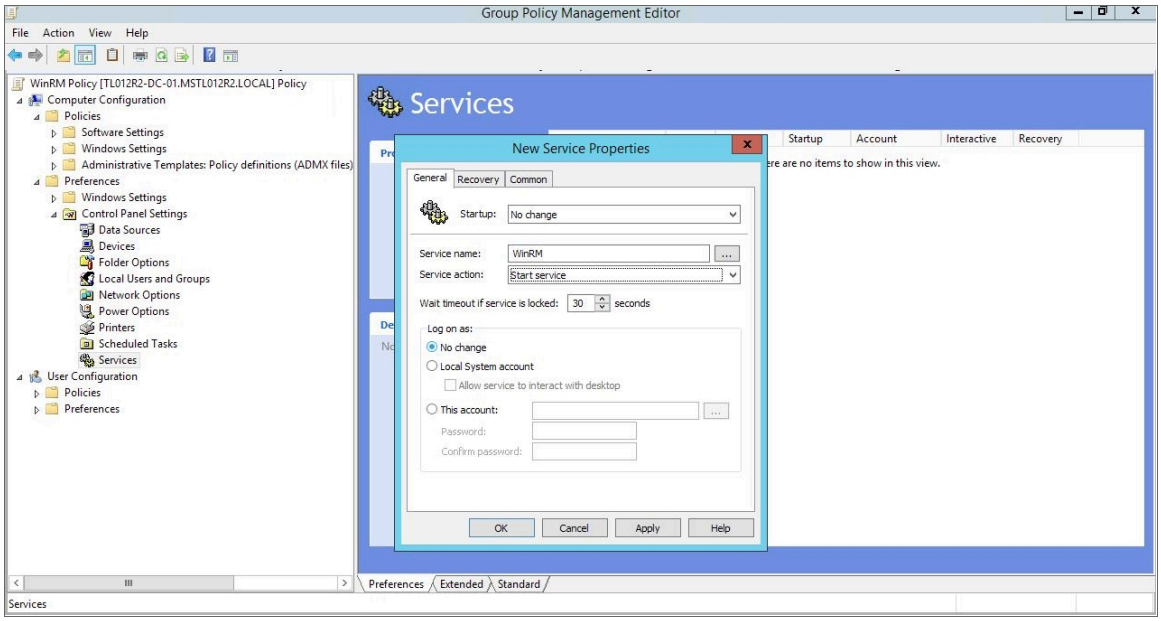
> **NOTE:** This step is required only if the user account is **not** a domain account and **not** the built-in local administrator account.

- ***Action***. Select *Create*.
- ***Hive***. Select *HKEY_LOCAL_MACHINE*.
- ***Key Path***. Enter "SOFTWARE\Microsoft\Windows\CurrentVersion\policies\system".
- ***Value name***. Enter "LocalAccountTokenFilterPolicy".
- ***Value type***. Enter "REG_DWORD".
- ***Value data***. Enter "1".
- ***Base***. Select *Decimal*.

39. In the left panel of the **Group Policy Management Editor** page, navigate to **Computer Configuration > Preferences > Control Panel Settings > Services**. In the right panel, right-click and select *New > Service*.

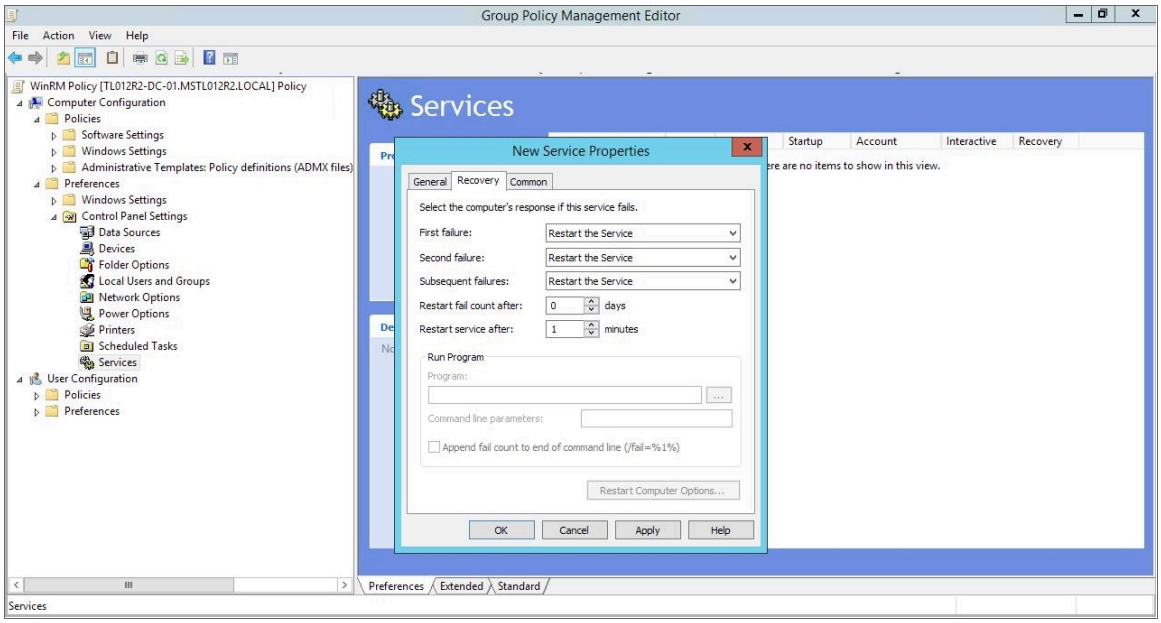


40. In the **New Service Properties** modal page, edit the values in one or more of the following fields:



- ***Startup***. Select *No change*.
- ***Service name***. Enter "WinRM".
- ***Service action***. Select *Start service*.
- ***Wait timeout if service is locked***. Select *30* seconds.
- ***Log on as***. Select *No change*.

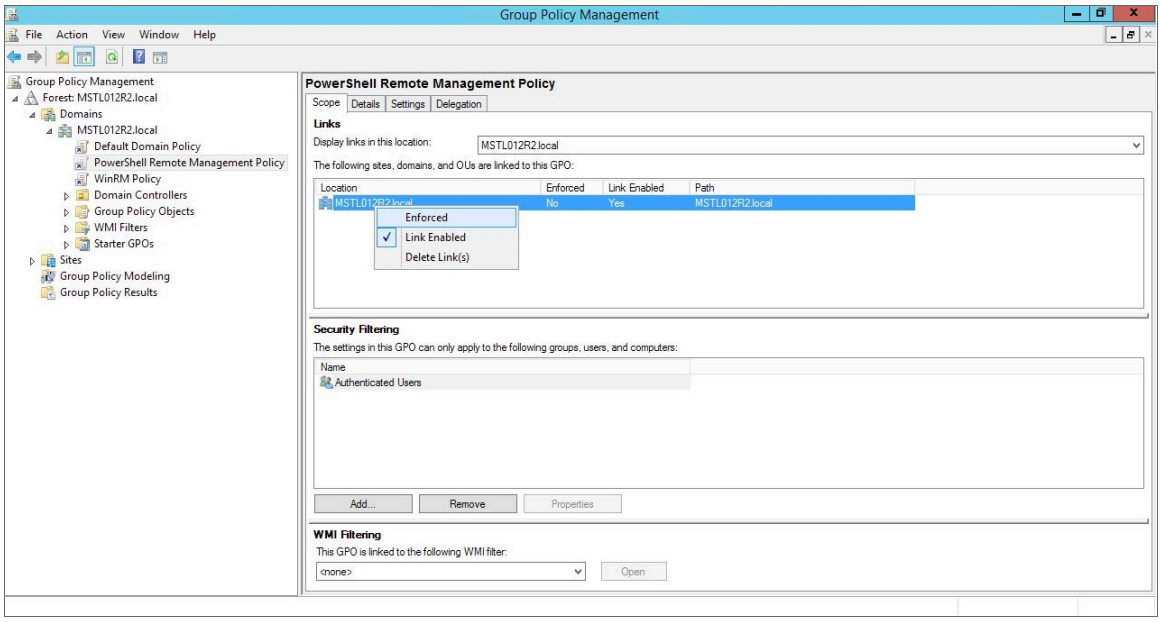
41. Click the **[Recovery]** tab, then edit the values in one or more of the following fields:



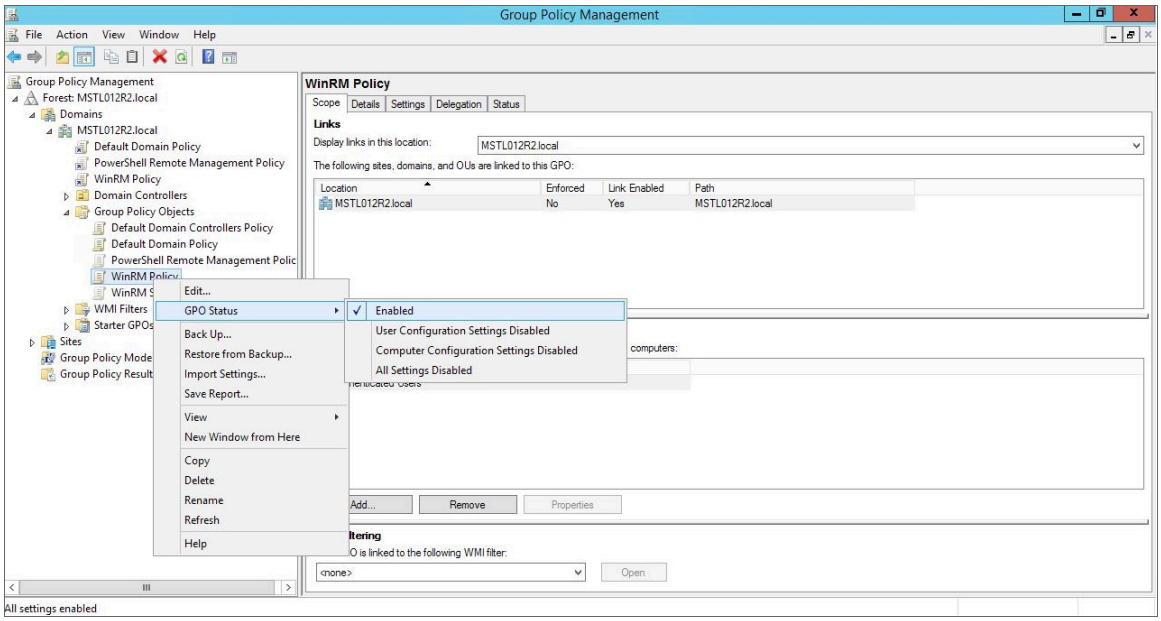
- ***First failure***. Select *Restart the Service*.

- ***Second failure***. Select *Restart the Service*.
- ***Subsequent failures***. Select *Restart the Service*.
- ***Restart fail count after***. Select *0* days.
- ***Restart service after***. Select *1* minute.

42. Click the **[OK]** button.
43. To enforce your group policy, in the left panel of the **Group Policy Management Editor** page, navigate to **Forest > Domains > [your local domain] > PowerShell Remote Management Policy**. In the **PowerShell Remote Management Policy** panel on the right, right-click the local domain name under ***The following sites, domains, and OUs are linked to this GPO*** and select *Enforced*.



44. To enable your group policy, in the left panel of the **Group Policy Management Editor** page, navigate to **Forest > Domains > [your local domain] > Group Policy Objects > WinRM Policy**. Right-click **WinRM Policy**, then select *GPO Status > Enabled*.



Configuring an HTTPS Listener with GPO Configuration

If you are using an HTTPS listener, you cannot create the listener and start it on the monitored device within group policy object (GPO) configuration. This can be done by using a startup script or an immediate task in the group policy, or by running a command manually or on the remote management tool on the device to be monitored. This command needs to be run only once as the HTTPS listener will automatically start once configured.

To perform this configuration within the group policy, perform the following steps:

1. Run the following command on the device you want to monitor:

```
winrm quickconfig -transport:https -force
```

This command will select the first available certificate enabled for server authentication. If you have multiple, valid server authentication certificates installed on your device, you will need to specify the thumbprint of the certificate and use the following command instead:

```
New-Item -Path WSMan:\LocalHost\Listener -Transport HTTPS -Address *
-CertificateThumbPrint "<CertThumbprint>" -Force
```

NOTE: The thumbprint should not contain spaces.

Using Forward and Reverse DNS for Windows Remote Management

When using Active Directory accounts for PowerShell monitoring, Kerberos and Windows Remote Management (WinRM) are used to connect to Windows devices and execute PowerShell code on those devices. Kerberos is

used to request a ticket for authentication to the Windows device, and WinRM is used to execute code on the Windows device.

In a Windows Active Directory configuration, Kerberos needs to be able to communicate with the target Windows device and the Active Directory Domain Controller to verify credentials and issue a ticket for authentication. Kerberos refers to a Windows Domain as a "realm" and an Active Directory Server as a "kdc" (Key Distribution Center).

For this process, it is important that forward and reverse lookup is working for all systems involved. Forward lookup translates a host to an IP address; reverse lookup translates an IP address to a host.

This can be managed through DNS, where a forward lookup is handled through an "A" record in a forward lookup zone, and reverse lookup through a "PTR" record in a reverse lookup zone. A utility such as "nslookup" will work correctly only if the DNS record (a PTR record, in this case) is present.

Where DNS is not available or reliable, it is possible to use the hosts file (`/etc/hosts`) instead. SL1 uses Python, which in turn can use the hosts file to provide both forward and reverse lookup. However, this approach means a higher level of server management because the hosts files on multiple Data Collector servers would need to be kept in sync. Additionally, where Concurrent PowerShell is used, the hosts files within the Docker containers would need to be updated.

Without a reliable forward and reverse lookup mechanism in place, Kerberos may not be able to validate credentials and issue a ticket for access to a Windows Device, which in turn would mean that access over WinRM to the device would be rejected.

Step 4: (Optional) Configuring a Windows Management Proxy

If SL1 cannot execute PowerShell requests directly on a Windows server, you can optionally configure an additional Windows server to act as a proxy for those PowerShell requests. To use a proxy, you must configure at least two Windows servers:

- A target server that SL1 cannot communicate with directly.
- A proxy server that SL1 will communicate with to execute PowerShell requests on the target server.

> **NOTE:** When monitoring a Windows device using a proxy, the account specified in the credentials is used to access both the proxy server and the target device. This account must have the correct access rights to be used on both servers. If multiple Active Directory domains are used, a trust relationship must be in place that allows the specified account access to the servers in both domains.

To configure the target and proxy servers, perform the following steps:

1. Configure a user account that SL1 will use to connect to the proxy server and the proxy server will use to connect to the target server. The user account can either be a local account or an Active Directory account; however, the user account must have the same credentials on the target and proxy servers and be in the Local Administrator's group on both servers.
2. If you have created a local user account on the Windows Server instead of an Active Directory account, you must configure encrypted communication between SL1 and the Windows server. To do this, you must *configure a Server Authentication certificate*.

3. *Configure Windows Remote Management* on the target server and the proxy server.
4. Log in to the proxy server as an administrator.
5. Open the PowerShell command window.
6. Right-click on the PowerShell icon in the taskbar and select *Run as Administrator*.
7. Execute one of the following commands on the proxy server to allow the proxy server to trust one or more target servers:
   - To allow the proxy server to trust all servers (not recommended), execute the following command:

     ```
     Set-Item WSMan:\Localhost\Client\TrustedHosts -value *
     ```

   - To allow the proxy server to trust only specific target servers, execute the following command, inserting a list that includes the IP address for each target server. Separate the list of IP addresses with commas.

     ```
     Set-Item WSMan:\Localhost\Client\TrustedHosts -value <comma-delimited-list-of-target-server-IPs>
     ```

NOTE: The following step is required only if the user account is **not** a domain account and **not** the built-in local administrator account.

8. Execute the following command on the proxy server to configure the LocalAccountTokenFilterPolicy:

   ```
   New-ItemProperty "HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System" -Name "LocalAccountTokenFilterPolicy" -Value 1 -PropertyType "DWORD"
   ```

NOTE: If the proxy server is in a different Windows domain (domain A) than the target servers (domain B), and the proxy server uses a user account from Active Directory, and Active Directory is in the same Windows domain as the target servers (domain B), you must perform the following to allow the proxy server to send PowerShell commands to the target servers:

- On the domain controller for each domain (domain A and domain B), create new forward-lookup zones and reverse-lookup zones that allow name resolution to work between the two domains.
- On the domain controller for each domain (domain A and domain B), create a non-transitive realm trust between the two domains.
- Login to the proxy server and add the Active Directory account (from domain A) to the Local Administrator's group for the proxy server. You should be able to select the account on the proxy server after you create the non-transitive realm trust between the two domains.

Step 5: (Optional) Increasing the Number of PowerShell Dynamic Applications That Can Run Simultaneously

You can optionally execute a series of commands that will allow SL1 to increase the default maximum number of PowerShell Dynamic Applications that can run simultaneously.

To do so:

1. Determine the number of Dynamic Applications that will be used to monitor the Windows server. Multiply this number by three.
2. Open a PowerShell command prompt. Log in as an Administrator.
3. At the prompt, execute the following commands:

```
Set-Item WSMan:\Localhost\Shell\MaxShellsPerUser -value <number you calculated in step 1>
```

```
Set-Item WSMan:\Localhost\Service\MaxConcurrentOperationsPerUser -value <number you calculated in step 1>
```

```
Restart-Service WinRM
```

4. Repeat these steps on each Windows server that will be monitored by SL1.

Optional PowerShell CLI Parameters

You can use the following parameters in PowerShell for the associated reasons:

- ***-NoProfile***. Does not load the PowerShell profile.
- ***-NoLogo***. Hides the copyright banner at startup.
- ***-NonInteractive***. Does not present an interactive prompt to the user.

To enable concurrent PowerShell collection to use one of these parameters:

1. Go to the **Database Tool** page (System > Tools > DB Tool).
2. If this row does not already exist in the `master.system_custom_config` table, enter the following in the ***SQL Query*** field:

```
INSERT INTO master.system_custom_config (`powershell_prefix_setting`, `<PREFIX INTEGER>`)
```

where:

<PREFIX> is an integer that represents one of the prefix values described above. The integers are as follows:

- **0**. Disabled
- ***1***. -NoProfile
- ***2***. -NoLogo
- ***3***. -NoProfile and -NoLogo
- ***4***. -NonInteractive
- ***7***. -NoProfile, -NoLogo, and -NonInteractive

For example, if a user wanted to configure their PowerShell Data Collector to not load their PowerShell profile, they would enter the following into the ***SQL Query*** field:

```
INSERT INTO master.system_custom_config (`powershell_prefix_setting`,
`1`)
```

3. If this row already exists in the `master.system_custom_config` table, enter the following in the ***SQL Query*** field:

```
UPDATE master.system_custom_config SET field_value = 1 WHERE field =
`powershell_prefix_setting`
```

4. After you have entered the command in the ***SQL Query*** field, click the **[Go]** button. Your changes will be picked up with the next batch of jobs that are processed.

ScienceLogic

800-SCI-LOGIC (1-800-724-5644)

International: +1-703-354-1010