Splunk PowerShell Resource Kit

Version

Table of Contents

[Document Information 4](#_Toc312358717)

[Copyright 4](#_Toc312358718)

[Version 4](#_Toc312358719)

[Document Conventions 4](#_Toc312358720)

[Getting Started with the Splunk PowerShell Resource Kit 6](#_Toc312358721)

[Notes 6](#_Toc312358722)

[Verify the Splunk PowerShell Resource Kit Installation 6](#_Toc312358723)

[Import the Splunk Module 7](#_Toc312358724)

[Query the Splunk Module Features 7](#_Toc312358725)

[Get Help for a Splunk Module Cmdlet 8](#_Toc312358726)

[Disable SSL Certificate Validation 11](#_Toc312358727)

[Create a Default Splunk Connection 12](#_Toc312358728)

[Check and Manage Splunk Services 14](#_Toc312358729)

[Test Active Directory Objects for Splunk Services 14](#_Toc312358730)

[Query the Status of Splunk Services on a Set of Hosts 14](#_Toc312358731)

[Manage Splunk Services on a Set of Hosts 17](#_Toc312358732)

[Restart Splunk 18](#_Toc312358733)

[Search Splunk 19](#_Toc312358734)

[View Raw Event Data 19](#_Toc312358735)

[View Event Data in a Table 20](#_Toc312358736)

[Specify Alternate Credentials for a Splunk Search 22](#_Toc312358737)

[Manage Splunk Server Classes 24](#_Toc312358738)

[Retrieve a List of Server Classes 24](#_Toc312358739)

[Retrieve a List of Deployment Clients 25](#_Toc312358740)

[Create a New Server Class 25](#_Toc312358741)

[Remove a Server Class 26](#_Toc312358742)

[Add Hosts from Active Directory to Server Class White List 26](#_Toc312358743)

[Add Hosts from an Active Directory Organizational Unit to Server Class White List 27](#_Toc312358744)

[Add Hosts from an Active Directory Group to Server Class White List 28](#_Toc312358745)

[Add List of Hosts from Splunk Search to Server Class White List 29](#_Toc312358746)

[Add a List of VM Host Names from HyperV to a Server Class White List 30](#_Toc312358747)

[Phase 2 32](#_Toc312358748)

[Apply an Input Configuration to Multiple Forwarders across all Hosts in an AD Site. 32](#_Toc312358749)

[Apply an Output Configuration to Multiple Forwarders across all Hosts in an AD Site. 33](#_Toc312358750)

[Apply an Application Configuration to Multiple Forwarders across all Hosts in an AD Site. 33](#_Toc312358751)

[Install a Splunk Application to Multiple Forwarders across all Hosts in an AD site. 34](#_Toc312358752)

[Apply an Input Configuration to Multiple Forwarders across all Hosts in an AD OU. 35](#_Toc312358753)

[Apply an Output Configuration to Multiple Forwarders across all Hosts in an AD OU. 36](#_Toc312358754)

[Apply an Application Configuration to Multiple Forwarders across all Hosts in an AD OU. 37](#_Toc312358755)

[Install a Splunk Application to Multiple Forwarders across all Hosts in an AD OU. 38](#_Toc312358756)

[Apply an Input Configuration to Multiple Forwarders across all Hosts Active in the Domain. 39](#_Toc312358757)

[Apply an Output Configuration to Multiple Forwarders across all Hosts Active in the Domain. 40](#_Toc312358758)

[Apply an Application Configuration to Multiple Forwarders across all Hosts Active in the Domain. 41](#_Toc312358759)

[Install a Splunk Application to Multiple Forwarders across all Hosts Active in the Domain. 42](#_Toc312358760)

[Run Tasks that Impact Multiple Hosts in Parallel. 43](#_Toc312358761)

[Forward Splunk alerts to SCOM. 45](#_Toc312358762)

[Forward SCOM alerts to Splunk. 45](#_Toc312358763)

# Document Information

This section describes aspects of this document.

## Copyright

Copyright © 2011 Splunk, All Rights Reserved.

## Version

This document is currently version , published .

The examples in this documentation are based on version of the Splunk PowerShell Module.

### Revision History

|  |  |  |
| --- | --- | --- |
| Date | Author | Revisions |
| 12/11/2011 | Jim Christopher | Added new recipes for Phase 2 sprint. |
| 08/03/2011 | Shell Consulting | Merged changes from John Cervelli. Rev'ed document version to 1.0.1. |
| 08/01/2011 | Shell Consulting | Initial version. Added cookbook examples for Splunk status, search, and server class management.  Based on Splunk module version 1.0.0.0. |

## Document Conventions

The following typographical conventions are used throughout this document:

Plain text

This is used for general text.

Code example

This is used to convey a PowerShell session. Lines that represent user commands are marked with a generic prompt indicator (PS >). Lines of PowerShell output are not prefaced with a prompt indicator.

In some cases, a single command will exceed the line length of the page. The PowerShell line continuation character (`) is used to indicate that a command continues to the next line.

In cases where PowerShell output has been truncated to fit the particular example, missing output is depicted by ellipsis (…).

Constant width

This indicates command names, options, parameters, variable names, URIs, or external programs.

**Constant width bold**

This indicates commands that must be typed in verbatim.

In some cases, a single command will exceed the line length of the page. The PowerShell line continuation character (`) is used to indicate that a command continues to the next line.

Cases where this typeface is wrapped in brackets ([ … ]) represent an optional element; the brackets are not part of the element and are not meant to be typed into PowerShell.

Constant width italic

This represents placeholders that should be replaced with user-specific values.

Cases where this typeface is wrapped in brackets ([ … ]) represent an optional element; the brackets are not part of the element and are not meant to be typed into PowerShell.

# Getting Started with the Splunk PowerShell Resource Kit

The Splunk PowerShell Resource Kit enables IT administrators to manage their Splunk topology, configure Splunk internals, and engage the Splunk search engine from their PowerShell session.

For instance, here are a few of the tasks enabled by the resource kit; each of these tasks can be completed using a single line of PowerShell:

* Determine or change the status of Splunk services across a set of Splunk servers in parallel;
* Force one or more Splunk servers to reload their configuration, in parallel;
* Deploy multiple Splunk forwarders to all active hosts in a Windows domain;
* Retrieve a list of Splunk server classes, optionally filtered by last deployment client connection time, associated applications, or matching patterns;
* Issue a Splunk search and format the retrieved events as a table, a list, or in a windowed grid view.

It is strongly recommended that the user read the complete section (starting on page 5 and ending on page 12) before skipping to specific administration recipes in the remainder of the document.

## Notes

Most of the examples in this document leverage a feature of the Splunk module described in on page 11. This is done mostly for example brevity, since most of the parameter values would simply be repeated in each example.

In addition, your Splunk environment may require you to as described on page 11. Note that a future version of the Splunk PowerShell Resource Kit will include the ability to manage certificates from your own Certificate Authority for use in symmetric key validation.

## Verify the Splunk PowerShell Resource Kit Installation

### Problem

You want to verify that the Splunk PowerShell Resource Kit is installed on your local computer.

### Solution

Use the built-in Get-Module PowerShell cmdlet to check the list of available modules.

PS > Get-Module –ListAvailable –Name Splunk

ModuleType Name ExportedCommands

---------- ---- ----------------

Script Splunk {Set-SplunkLicensePool, ...

### Discussion

The Splunk PowerShell Resource Kit is distributed as a PowerShell module named Splunk. If the module is properly installed on your computer, it will appear in the list of available modules. This list is obtained using the standard Get-Module cmdlet with the –ListAvailable parameter.

If the module is not installed, or if it is installed incorrectly, the Get-Module command for this recipe will output nothing. If this occurs, check that the Splunk module is available in your PowerShell module path.

### See Also

* **Get-Help about\_modules**
* **Get-Help Get-Module**
* Import the Splunk Module on page 6.

## Import the Splunk Module

### Problem

You want to import the Splunk module into your existing PowerShell session to enable Splunk management and search features.

### Solution

Use the built-in Import-Module PowerShell cmdlet to import the Splunk module.

PS > Import-Module –Name Splunk

### Discussion

The Splunk PowerShell Resource Kit is distributed as a PowerShell module named “Splunk.” In order to make use of the module features, this module must be imported into your current PowerShell session.

The Import-Module cmdlet takes the name of the module to import as the –Name parameter. Once this Import-Module cmdlet is executed, the Splunk PowerShell Resource Kit cmdlets will be available during your PowerShell session.

### See Also

* **Get-Help about\_modules**
* **Get-Help Import-Module**

## Query the Splunk Module Features

### Problem

You want to see what commands are available in the Splunk module.

### Solution

Use the Splunk module command Get-Splunk to retrieve a list of available module cmdlets.

PS > Get-Splunk

CommandType Name Definition

----------- ---- ----------

Function Add-SplunkLicenseFile ...

Function Add-SplunkLicensePool ...

Function Connect-Splunk ...

Function ConvertFrom-SplunkTime param($TimeAccessed)...

### Discussion

The Get-Splunk cmdlet returns a list of all cmdlets available in the Splunk module.

If you are interested in commands that apply only to specific objects or actions, you can use the optional –Noun or –Verb parameters, respectively. These parameters accept wildcard patterns that match against the names of the Splunk module cmdlets:

PS > Get-Splunk -Noun \*licensegroup

CommandType Name Definition

----------- ---- ----------

Function Get-SplunkLicenseGroup ...

Function Set-SplunkLicenseGroup ...

PS > Get-Splunk –Verb remove

CommandType Name Definition

----------- ---- ----------

Function Remove-SplunkConnectionObject ...

Function Remove-SplunkLicenseFile ...

Function Remove-SplunkLicensePool ...

### See Also

* **Get-Help about\_modules**
* **Get-Help Import-Module**
* **Get-Help Get-Splunk**

## Get Help for a Splunk Module Cmdlet

### Problem

You want to read the documentation for a Splunk module cmdlet.

### Solution

Use the standard PowerShell Get-Help cmdlet, passing the name of the Splunk module cmdlet in the –Name parameter:

Get-Help –Name CommandName

This example retrieves documentation for the Get-Splunkd cmdlet:

PS > Get-Help –Name Get-Splunkd

NAME

Get-Splunkd

SYNOPSIS

Gets the values set for the targeted Splunk instance.

SYNTAX

Get-Splunkd [[-ComputerName] [<String>]] [[-Port] [<Int32>]] [[-Protocol]

[<String>]] [[-Timeout] [<Int32>]] [[-Credential] [<PSCredential>]]

[<CommonParameters>]

DESCRIPTION

Gets the values set for the targeted Splunk instance. These are the

settings found in the Splunk web interface Manager > System settings >

General settings

RELATED LINKS

Get-Splunkd

REMARKS

To see the examples, type: "get-help Get-Splunkd -examples".

For more information, type: "get-help Get-Splunkd -detailed".

For technical information, type: "get-help Get-Splunkd -full".

### Discussion

The Splunk module contains detailed documentation for each of its cmdlets. Using Get-Help with the –Name parameter outputs the terse documentation shown in the above example.

If you wish to view more detailed documentation, use the –Detailed parameter to show more verbose cmdlet documentation:

PS > Get-Help –Name Get-Splunkd –Detailed

NAME

Get-Splunkd

SYNOPSIS

Gets the values set for the targeted Splunk instance.

SYNTAX

Get-Splunkd [[-ComputerName] [<String>]] [[-Port] [<Int32>]] [[-Protocol]

[<String>]] [[-Timeout] [<Int32>]] [[-Credential] [<PSCredential>]]

[<CommonParameters>]

DESCRIPTION

Gets the values set for the targeted Splunk instance. These are the

settings found in the Splunk web interface Manager > System settings >

General settings

PARAMETERS

-ComputerName <String>

Name of the Splunk instance to get the settings for (Default is

$SplunkDefaultConnectionObject.ComputerName.)

-Port <Int32>

Port of the REST Instance (i.e. 8089) (Default is

$SplunkDefaultConnectionObject.Port.)

-Protocol <String>

Protocol to use to access the REST API must be 'http' or 'https'

(Default is $SplunkDefaultConnectionObject.Protocol.)

-Timeout <Int32>

How long to wait for the REST API to respond (Default is

$SplunkDefaultConnectionObject.Timeout.)

-Credential <PSCredential>

Credential object with the user name and password used to access the

REST API (Default is $SplunkDefaultConnectionObject.Credential.)

<CommonParameters>

This cmdlet supports the common parameters: Verbose, Debug,

ErrorAction, ErrorVariable, WarningAction, WarningVariable,

OutBuffer and OutVariable. For more information, type,

"get-help about\_commonparameters".

-------------------------- EXAMPLE 1 --------------------------

Get-Splunkd

Description

-----------

Gets the values set for the targeted Splunk instance using the

$SplunkDefaultConnectionObject settings.

...

REMARKS

To see the examples, type: "get-help Get-Splunkd -examples".

For more information, type: "get-help Get-Splunkd -detailed".

For technical information, type: "get-help Get-Splunkd -full".

You can use the –Examples parameter to see the examples for the specified cmdlet:

PS > Get-Help –Name Get-Splunkd -Examples

NAME

Get-Splunkd

SYNOPSIS

Gets the values set for the targeted Splunk instance.

-------------------------- EXAMPLE 1 --------------------------

Get-Splunkd

Description

-----------

Gets the values set for the targeted Splunk instance using the

$SplunkDefaultConnectionObject settings.

-------------------------- EXAMPLE 2 --------------------------

Get-Splunkd -ComputerName MySplunkInstance -Port 8089 -Protocol https `

–Timeout 5000 -Credential $MyCreds

Description

-----------

Gets the values set for MySplunkInstance connecting on port 8089 with a 5

second timeout.

-------------------------- EXAMPLE 3 --------------------------

$SplunkServers | Get-Splunkd

Description

-----------

Gets the values set for each Splunk server in the pipeline using the

$SplunkDefaultConnectionObject settings.

-------------------------- EXAMPLE 4 --------------------------

$SplunkServers | Get-Splunkd -Port 8089 -Protocol https -Timeout 5000 `

–Credential $MyCreds

Description

-----------

Gets the values set for each Splunk server in the pipeline connecting on

port 8089 with a 5 second timeout and using credentials provided.

If you require more thorough information about one or more cmdlet parameters, specify the –Parameter parameter, including a wildcard pattern of parameter names:

PS > Get-Help –Name Get-Splunkd –Parameter Proto\*

-Protocol <String>

Protocol to use to access the REST API must be 'http' or 'https' (Default

is $SplunkDefaultConnectionObject.Protocol.)

Required? false

Position? 3

Default value

Accept pipeline input? false

Accept wildcard characters? false

Finally, if you would like to view the complete documentation for the cmdlet, including the synopsis, details, examples, parameter information, use the –Full parameter of the Get-Help cmdlet:

Get-Help –Name *CommandName* -Full

### See Also

* **Get-Help Get-Help -Full**

## Disable SSL Certificate Validation

### Problem

You want to disable SSL certificate validation in your PowerShell session to prevent connection errors while working against Splunk servers using default untrusted SSL certificates.

### Solution

Use the Splunk module Disable-CertificateValidation cmdlet to suspend certificate validation:

Disable-CertificateValidation

### Discussion

By default Splunk servers use an SSL certificate from an untrusted issuer (SplunkServerDefaultCert). In order to connect over HTTPS to a server using this default certificate, normal SSL validation procedures must be bypassed.

The Disable-CertificateValidation cmdlet changes your PowerShell session to bypass normal SSL certificate validation procedures. This affects only your PowerShell process – no web browsers or other programs are affected by using this cmdlet. The validation process remains bypassed until one of the following occurs:

* The Splunk module Enable-CertificateValidation cmdlet is invoked;
* The current PowerShell session ends.

### See Also

* **Get-Help Disable-CertificateValidation**
* **Get-Help Enable-CertificateValidation**

## Create a Default Splunk Connection

### Problem

You want to use a common set of Splunk instance connection parameters for Splunk module cmdlets.

### Solution

Use the Connect-Splunk cmdlet to define a default Splunk connection:

Connect-Splunk –ComputerName SplunkServerName `

–Credentials SplunkCredentials `

[–Port ManagementPortNumber] [–Protocol HttpsOrHttp ] `

[–Timeout ConnectionTimeoutInMilliseconds]

This example connects to the Splunk server named “SplunkServer001” using the default timeout, port, and protocol values. The credentials used to connect to the server are collected using the built-in Get-Credential PowerShell cmdlet. Finally, the Test-Splunkd Splunk module cmdlet is invoked using the cached connection parameters:

PS > $credential = Get-Credential

cmdlet Get-Credential at command pipeline position 1

Supply values for the following parameters:

Credential

PS > Connect-Splunk -Credential $credential –ComputerName SplunkServer001

PS > Get-Splunkd

ComputerName OSName Version MgmtPort SessionTimeout EnableWeb …

--------------- ------ ------- -------- -------------- --------- …

SplunkServer001 Windows 4.2.2 8089 1h True …

### Discussion

Most cmdlets in the Splunk module accept the same set of connection parameters; these include:

* -ComputerName: the name or IP address of the Splunk instance to which to connect;
* -Protocol: the protocol to use to connect, either **http** or **https**;
* -Port: the management port to which to connect;
* -Credential: the connection credentials used to authenticate with the Splunk server.

Because specifying these parameters for every cmdlet can be cumbersome, the Splunk module can cache the connection parameters and use them as default values.

### See Also

* **Get-Help Connect-Splunk**
* **Get-Help Get-Credential**

# Check and Manage Splunk Services

## Test Active Directory Objects for Splunk Services

### Problem

You want to obtain a list of computers from Active Directory and determine which have Splunk services running on them.

### Solution

Pipe the output of the Active Directory module Get-ADComputer cmdlet to the Test-Splunkd cmdlet.

PS > Get-ADComputer –Filter \* | Test-Splunkd

WinXP001

Lnx001

…

### Discussion

By specifying the –Filter parameter as a match-all wildcard (\*), the Get-ADComputer cmdlet will return every computer object from Active Directory. These computer objects are then piped to the Test-Splunkd cmdlet from the Splunk module.

Test-Splunkd will query each computer specified in the pipeline for the presence of active Splunk services. The cmdlet outputs the names of computers on which one or more Splunk services are active.

The Get-ADComputer cmdlet is available as part of the Active Directory module for Windows PowerShell in Windows Server 2008 R2. For more information on the Active Directory module please see the following URL:

* http://go.microsoft.com/fwlink/?LinkID=139658

### See Also

* **Get-Help Test-Splunkd**
* http://go.microsoft.com/fwlink/?LinkID=139658
* Create a Default Splunk Connection on page 11

## Query the Status of Splunk Services on a Set of Hosts

### Problem

You want to know the status of Splunk services across a set of hosts.

### Solution

Use the built-in Get-Service PowerShell cmdlet to retrieve the Splunk services from the list of hosts:

Get-Service –Name Splunk\* -ComputerName ListOfHostNames

To view the list of services for each host, pipe the result of the Get-Service cmdlet to the Format-Table PowerShell cmdlet, specifying the MachineName, DisplayName, and Status property names for the –Property parameter:

Format-Table –Property MachineName, DisplayName, Status

In this example, the status of Splunk services is queried for the ‘Server001’ and ‘Server002’ hosts:

PS > Get-Service –Name splunk\* -ComputerName Server001,Server002 | `

Format-Table –Property MachineName, Status, DisplayName

MachineName Status DisplayName

----------- ------- -----------

Server001 Running Splunkd

Server001 Running Splunkweb

Server002 Stopped Splunkd

### Discussion

This example filters the list of services returned by Get-Service to those with names that start with “Splunk” by using a wildcard expression Splunk\* for the –Name parameter. As a result the Get-Service output will include the Splunkd management service, the Splunkweb web interface service, and the Splunkforwarder service.

The Get-Service cmdlet returns System.ServiceProcess.ServiceController .NET objects. PowerShell has a default format for these objects that will report the service name and current status, but this default format does not include the machine name on which the service is running. For example:

PS > Get-Service

Status Name DisplayName

------ ---- -----------

Stopped AeLookupSvc Application Experience

Stopped ALG Application Layer Gateway Service

Stopped AppIDSvc Application Identity

Running Appinfo Application Information

…

In order to see the service host information, the Get-Service output must be manipulated. In this case the output is piped to the Format-Table standard PowerShell cmdlet with a list of wanted System.ServiceProcess.ServiceController properties specified in the –Property parameter. The result is a well-formatted table of host names, service status, and service names.

You can view a list of available System.ServiceProcess.ServiceController properties using the Get-Member standard cmdlet:

PS > Get-Service | Get-Member –MemberType Properties

TypeName: System.ServiceProcess.ServiceController

Name MemberType Definition

---- ---------- ----------

Name AliasProperty Name = ServiceName

RequiredServices AliasProperty RequiredServices = ServicesDependedOn

CanPauseAndContinue Property System.Boolean CanPauseAndContinue {get;}

CanShutdown Property System.Boolean CanShutdown {get;}

CanStop Property System.Boolean CanStop {get;}

Container Property …

DependentServices Property …

DisplayName Property System.String DisplayName {get;set;}

MachineName Property System.String MachineName {get;set;}

ServiceHandle Property …

ServiceName Property System.String ServiceName {get;set;}

ServicesDependedOn Property …

ServiceType Property …

Site Property System.ComponentModel.ISite Site {get;set;}

Status Property …

You can specify any of these property names to Format-Table to customize the output. For instance:

PS > Get-Service –Name splunk\* | `

Format-Table –Property Name, CanStop, CanShutdown

Name CanStop CanShutdown

---- ------- -----------

Splunkd True True

Splunkweb True False

#### Sorting the Output by Service Name

Because PowerShell cmdlets write objects and not simple text, you have great control over the output and formatting. For instance, you can easily augment this example to sort the output by service name using the Sort-Object built-in PowerShell cmdlet:

PS > Get-Service –Name splunk\* -ComputerName Server001,Server002 | `

Sort-Object –Property DisplayName | `

Format-Table –Property MachineName, Status, DisplayName

MachineName Status DisplayName

----------- ------- -----------

Server001 Running Splunkd

Server002 Stopped Splunkd

Server001 Running Splunkweb

The output from Get-Service is piped to the Sort-Object cmdlet before being piped to the Format-Table cmdlet for formatting. The Sort-Object cmdlet reorders the list of System.ServiceProcess.ServiceController objects based on the values of their DisplayName property.

### See Also

* **Get-Help Get-Service**
* **Get-Help Format-Table**
* **Get-Help Format-List**
* **Get-Help Format-Wide**
* **Get-Service | Get-Member**
* **Get-Help Sort-Object**
* Create a Default Splunk Connection on page 11

## Manage Splunk Services on a Set of Hosts

### Problem

You want to start or stop Splunk services on a set of hosts.

### Solution

Use the standard PowerShell Get-Service cmdlet to retrieve the Splunk services from the relevant host machines; please see Query the Status of Splunk Services on a Set of Hosts on page 13 for details.

Pipe the output of Get-Service to the built-in Foreach-Object cmdlet to iterate over each System.ServiceProcess.ServiceController object, stopping and starting the service as you need.

In this example, the Splunk services are started on servers Server001 and Server002:

PS > Get-Service –Name splunk\* -ComputerName Server001,Server002 | `

Foreach-Object –Process { $\_.Start() }

### Discussion

Please refer to Query the Status of Splunk Services on a Set of Hosts on page 13 for information on using the Get-Service PowerShell cmdlet to retrieve Splunk services on remote hosts.

The Foreach-Object cmdlet allows you to specify a script block in the –Process parameter that will be executed against each object in the pipeline. This script block has access to a special variable ($\_) that is the “current pipeline item.” In our solution, the $\_ variable represents each Splunk service object on Server001 and Server002. Within the script block we can call any methods and properties supported by this object.

You can discover service object properties and methods using the Get-Member standard cmdlet:

PS > Get-Service | Get-Member

TypeName: System.ServiceProcess.ServiceController

Name MemberType Definition

---- ---------- ----------

Name AliasProperty Name = ServiceName

RequiredServices AliasProperty RequiredServices = ServicesDependedOn

Disposed Event …

Close Method System.Void Close()

Continue Method System.Void Continue()

CreateObjRef Method …

Dispose Method System.Void Dispose()

Equals Method bool Equals(System.Object obj)

ExecuteCommand Method System.Void ExecuteCommand(int command)

GetHashCode Method int GetHashCode()

GetLifetimeService Method System.Object GetLifetimeService()

GetType Method type GetType()

InitializeLifetimeService Method System.Object InitializeLifetimeService()

Pause Method System.Void Pause()

Refresh Method System.Void Refresh()

Start Method …

Stop Method System.Void Stop()

ToString Method string ToString()

WaitForStatus Method …

CanPauseAndContinue Property System.Boolean CanPauseAndContinue {get;}

CanShutdown Property System.Boolean CanShutdown {get;}

CanStop Property System.Boolean CanStop {get;}

Container Property …

DependentServices Property …

DisplayName Property System.String DisplayName {get;set;}

MachineName Property System.String MachineName {get;set;}

ServiceHandle Property …

ServiceName Property System.String ServiceName {get;set;}

ServicesDependedOn Property …

ServiceType Property …

Site Property System.ComponentModel.ISite Site {get;set;}

Status Property …

### See Also

* **Get-Help Get-Service**
* **Get-Help Foreach-Object**
* Create a Default Splunk Connection on page 11

## Restart Splunk

### Problem

You want to force one or more Splunk servers to restart their Splunk services.

### Solution

Pipe the list of servers to the Splunk module Restart-SplunkService cmdlet:

ListOfServers | Restart-SplunkService

This example restarts the Splunk services on hosts Server001 and Server002:

 ‘Server001’,’Server002’ | Restart-SplunkService

### Discussion

The Restart-SplunkService cmdlet instructs the Splunk instance to restart its services.

The solution assumes that each server uses the same Splunk management port, protocol, and credentials, and that these connection parameters have been cached using the Connect-Splunk module cmdlet.

### See Also

* **Get-Help Restart-SplunkService**
* Create a Default Splunk Connection on page 11

# Search Splunk

## View Raw Event Data

### Problem

You want to execute a Splunk search and view the raw event text.

### Solution

Use the Splunk module Search-Splunk cmdlet to execute a search against a Splunk instance:

Search-Splunk –Search SearchString

To isolate the raw event text, pipe the output to the Select-Object built-in PowerShell cmdlet. Use the –ExpandProperty parameter to isolate and display the Raw search result data:

Select-Object –ExpandProperty Raw

This example executes a search for the term ‘User’ on the current default Splunk instance. The raw event text data is output to the console:

PS > Search-Splunk -Search 'User' | Select-Object -ExpandProperty Raw

07/31/2011 02:57:51 PM

LogName=Application

SourceName=Google Update

EventCode=20

EventType=2

Type=Warning

ComputerName=VBOX-XP

User=beefarino

Sid=S-1-5-21-1708537768-706699826-1957994488-1003

SidType=1

Category=0

CategoryString=none

RecordNumber=3156

Message=Splunk could not get the description for this event. Either the component that raises this event is not installed on your local computer or the installation is corrupt.

Got the following information from this event:

Network Request Error.

Error: 0x80072ee7. Http status code: 0.

…

### Discussion

The “raw” event data from Splunk has not been processed – it is the original event text data before any fieldfiled extractions, tags, transforms, or field modifications have been applied.

The Search-Splunk cmdlet returns a collection of Splunk.SDK.Search.OneshotResult objects that match the search string provided in the –Search parameter.

The default formatting provided by the Splunk module outputs these results in table. The raw event string data for each search result is stored in the Raw property of the Splunk.SDK.Search.OneshotResult object. In order to view the event data in its raw form, the Select-Object cmdlet is used to isolate and expand the Raw property.

For information about using server-side filtering and limiting with the Search-Splunk cmdlet, please see View Event Data in a Table on page 19.

### See Also

* **Get-Help Search-Splunk**
* **Get-Help Select-Object**
* Create a Default Splunk Connection on page 11
* View Event Data in a Table on page 19.

## View Event Data in a Table

### Problem

You want to execute a Splunk search and receive events in tabular form.

### Solution

Use the Search-Splunk cmdlet to execute a search.

This example searches the Splunk event data for the term “Error”:

PS > Search-Splunk –Search ‘Error’

Date Host Source SourceType …

---- ---- ------ ---------- …

7/31/2011 2:57:51 PM vbox-xp WinEventLog:Application WinEventLog:Application …

7/31/2011 2:50:22 PM vbox-xp WinEventLog:Application WinEventLog:Application …

7/31/2011 2:48:28 PM vbox-xp WinEventLog:Application WinEventLog:Application …

…

### Discussion

The Search-Splunk cmdlet returns a collection of Splunk.SDK.Search.OneshotResult objects that match the search string provided in the –Search parameter. By default, these results are displayed in a table.

You can customize the formatting of the search results using standard PowerShell formatting cmdlets. For example, to display the search results as a list of host name, event instant, and raw event data, you can use the Format-List cmdlet:

PS > Search-Splunk –Search 'error' | Format-List -Property host, source, raw

Host : vbox-xp

Source : WinEventLog:Application

raw : {07/31/2011 02:57:51 PM

LogName=Application

SourceName=Google Update

EventCode=20

EventType=2

Type=Warning

ComputerName=VBOX-XP

User=beefarino

Sid=S-1-5-21-1708537768-706699826-1957994488-1003

SidType=1

Category=0

CategoryString=none

RecordNumber=3156

Message=Splunk could not get the description for this event. Either the

component that raises this event is not installed on your local computer

or the installation is corrupt.

Got the following information from this event:

Network Request , .

, : 0x80072ee7. Http status code: 0.

}

Host : vbox-xp

Source : WinEventLog:Application

raw : {07/31/2011 02:50:22 PM

LogName=Application

SourceName=Google Update

EventCode=20

EventType=2

Type=Warning

ComputerName=VBOX-XP

User=beefarino

Sid=S-1-5-21-1708537768-706699826-1957994488-1003

SidType=1

Category=0

CategoryString=none

RecordNumber=3155

Message=Splunk could not get the description for this event. Either the component that raises this event is not installed on your local computer or th

e installation is corrupt.

Got the following information from this event:

Network Request , .

, : 0x80072ee7. Http status code: 0.

}

#### Server-Side Filtering and Limiting

If you expect a search to return many results, you can use the –MaxReturnCount parameter of Search-Splunk to return a limited number of records to your PowerShell session:

PS > Search-Splunk –Search 'user' –MaxReturnCount 2

Date Host Source …

---- ---- ------ …

7/31/2011 2:57:51 PM vbox-xp WinEventLog:Application …

7/31/2011 2:50:22 PM vbox-xp WinEventLog:Application …

You can also tailor your search to a specific time period using the –StartTime and –EndTime parameters.

Using these parameters is far more efficient than filtering results through a pipeline, because the filtering is performed on the Splunk server. For example, these two commands produce identical output; however, the first command must retrieve and process every search result before filtering them, while the second must only retrieve and process a single search result returned by the Splunk server:

PS > Search-Splunk –Search 'user' | Select-Object –First 1

Date Host Source …

---- ---- ------ …

7/31/2011 2:57:51 PM vbox-xp WinEventLog:Application …

PS > Search-Splunk –Search 'user' –MaxReturnCount 1

Date Host Source …

---- ---- ------ …

7/31/2011 2:57:51 PM vbox-xp WinEventLog:Application …

### See Also

* **Get-Help Search-Splunk**
* **Get-Help Select-Object**
* **Get-Help Format-List**
* **Get-Help Format-Table**
* **Get-Help Format-Wide**
* Create a Default Splunk Connection on page 11
* View Raw Event Data on page 17

## Specify Alternate Credentials for a Splunk Search

### Problem

You need to specify a set of Splunk credentials inline with a search request.

### Solution

Use the –Credential parameter of the Search-Splunk cmdlet to specify your Splunk username:

Search-Splunk –Search SearchString –Credential SplunkUserName

In this example, a search is executed against the current Splunk server using the username “searcher”:

PS > Search-Splunk -Search 'Error 1234' -Credential Searcher

Password: \*\*\*\*\*\*\*\*

Date Host Source …

---- ---- ------ …

7/31/2011 2:57:51 PM vbox-xp WinEventLog:Application …

…

### Discussion

When you specify a username in the –Credential parameter of any Splunk module cmdlet, PowerShell will prompt you for the correct Splunk password before attempting to connect to the Splunk server.

If you need to reuse a common set of Splunk credentials across a series of Splunk module cmdlets, consider setting a default credential set as described in Create a Default Splunk Connection on page 11. Alternatively, you can create a credential using the New-SplunkCredential cmdlet of the Splunk module. By saving this credential to a variable, you can reuse it easily:

PS > $cred = New-SplunkCredential –UserName Searcher

Password: \*\*\*\*\*\*\*\*\*\*\*

PS > Search-Splunk –Search ‘Error 1234’ –Credential $cred

### See Also

* **Get-Help Search-Splunk**
* **Get-Help New-SplunkCredential**
* **Get-Help Get-Credential**
* Create a Default Splunk Connection on page 11

# Manage Splunk Server Classes

## Retrieve a List of Server Classes

### Problem

You want a list of server classes, their associated apps and matching patterns.

### Solution

Use the Splunk module Get-SplunkServerClass cmdlet to retrieve a list of server classes from a Splunk instance:

Get-SplunkServerClass [-Filter ServerClassFilterString]

This example retrieves all available server classes from the current default Splunk instance:

PS > Get-SplunkServerClass

ComputerName Name Disabled FilterType RestartSplunkd Whitelist

------------ ---- -------- ---------- -------------- ---------

indexer ServerCls1 False whitelist SERVER001

indexer ITAdmin False whitelist IT1,IT2,IT3,…

indexer Marketing False whitelist MktgNFS,MktgSharepoint

indexer DMZClass False Blacklist

### Discussion

The Get-SplunkServerClass cmdlet returns server class objects. You can use PowerShell’s sorting and formatting cmdlets to customize the output to your needs.

For instance, you can sort the server class list by name by piping the output of Get-SplunkServerClass to the Sort-Object cmdlet:

PS > Get-SplunkServerClass | Sort-Object –Property Name

ComputerName Name Disabled FilterType RestartSplunkd Whitelist

------------ ---- -------- ---------- -------------- ---------

indexer DMZClass False Blacklist

indexer ITAdmin False whitelist IT1,IT2,IT3,…

indexer Marketing False whitelist MktgNFS,MktgSharepoint

indexer ServerCls1 False whitelist SERVER001

You can also filter the server class list using the built-in PowerShell Where-Object cmdlet. This example outputs only the server classes that filter by whitelists:

PS > Get-SplunkServerClass | Where-Object { $\_.FilterType –eq ‘whitelist’ }

ComputerName Name Disabled FilterType RestartSplunkd Whitelist

------------ ---- -------- ---------- -------------- ---------

indexer ServerCls1 False whitelist SERVER001

indexer ITAdmin False whitelist IT1,IT2,IT3,…

indexer Marketing False whitelist MktgNFS,MktgSharepoint

### See Also

* **Get-Help Get-SplunkServerClass**
* **Get-Help Sort-Object**
* **Get-Help Where-Object**
* Create a Default Splunk Connection on page 11

## Retrieve a List of Deployment Clients

### Problem

You want a list of server classes and connected deployment clients, filtered by last connect time.

### Solution

PS > Get-SplunkDeploymentClient

### Discussion

### See Also

* **Get-Help Get-SplunkDeploymentClient**
* Create a Default Splunk Connection on page 11

## Create a New Server Class

### Problem

You want to create a new server class.

### Solution

Use the Splunk module cmdlet New-SplunkServerClass to define the new server class.

In this example, the New-SplunkServerClass cmdlet is used to create a new whitelist server class that includes the hosts named Server01 and Server02 in the class whitelist:

PS > New-SlunkServerClass -Name MyServerClass -FilterType whitelist `

-WhiteList 'server01','server02'

ComputerName Name Disabled FilterType RestartSplunkd Whitelist …

------------ ---- -------- ---------- -------------- --------- …

indexer MyServerClass False whitelist SERVER01,SERVER02

### Discussion

The New-SplunkServerClass cmdlet is used to create a new server class on a Splunk host.

By default new server classes are created as whitelist filters. To create a new blacklist server class, specify the –FilterType parameter as “blacklist”. For instance:

PS > New-SplunkServerClass –Name NoWeb –FilterType Blacklist `

–Blacklist WebServer1, WebServer2

ComputerName Name Disabled FilterType RestartSplunkd … Blacklist

------------ ---- -------- ---------- -------------- … ---------

indexer MyServerClass False Blacklist … WebServer1,WebServer2

### See Also

* **Get-Help New-SplunkServerClass**
* Create a Default Splunk Connection on page 11

## Remove a Server Class

### Problem

You want to remove an existing server class from a Splunk instance.

### Solution

Pass the name of the server class to the Remove-SplunkServerClass cmdlet:

Remove-SplunkServerClass –Name ServerClassName

In this example, the server class with name ServerClass1 is

PS > Remove-SplunkServerClass -Name ServerClass1

### Discussion

## Add Hosts from Active Directory to Server Class White List

### Problem

You want to query Active Directory for all hosts in a site, and add them to a server class white list.

### Solution

Use the Active Directory module Get-ADComputer cmdlet to retrieve a list of hosts in the current site. Pass this list to the –Whitelist parameter of the New-SplunkServerClass Splunk module cmdlet.

For instance:

PS > $hosts = Get-ADComputer -Filter \* | Select-Object -ExpandProperty DnsHostName

PS > New-SplunkServerClass -Name AllHostsWhiteList `

-FilterType WhiteList -WhiteList $hosts

…

### Discussion

By specifying the –Filter parameter as a match-all wildcard (\*), the Get-ADComputer cmdlet will return a Microsoft.ActiveDirectory.Management.ADComputer object for every computer in Active Directory. Each object contains the computer host name in the DnsHostName property. The output of Get-ADComputer is piped to the built-in Select-Object PowerShell cmdlet; using the –ExpandProperty parameter isolates the DnsHostName property on each of the objects.

The result of this pipeline is the list of host names stored in the $hosts variable. This variable is used as the –Whitelist parameter value for the New-SplunkServerClass cmdlet.

The Get-ADComputer cmdlet is available as part of the Active Directory module for Windows PowerShell in Windows Server 2008 R2. For more information on the Active Directory module please see the following URL:

* http://go.microsoft.com/fwlink/?LinkID=139658

### See Also

* **Get-Help New-SplunkServerClass**
* **Get-Help Get-ADComputer**
* http://go.microsoft.com/fwlink/?LinkID=139658
* Create a Default Splunk Connection on page 11

## Add Hosts from an Active Directory Organizational Unit to Server Class White List

### Problem

You want to create a server class white list containing all of the hosts in an Active Directory Organizational Unit.

### Solution

Use the –Filter parameter of the Get-ADComputer cmdlet to match hosts belonging to the specific OU. Pass this list to the –Whitelist parameter of the New-SplunkServerClass Splunk module cmdlet.

This example creates a whitelist server class containing every computer from the LocalOrgUnit OU:

PS > $hosts = Get-ADComputer -Filter { DistinguishedName –like “\*OU=LocalOrgUnit\*” } `

| Select-Object -ExpandProperty DnsHostName

PS > New-SplunkServerClass -Name OUWhiteList -FilterType WhiteList -WhiteList $hosts

…

### Discussion

The –Filter parameter of Get-ADComputer can be a filter expression. By creating a filter expression that matches the OU name in the computer’s DistinguishedName value, the Get-ADComputer cmdlet will return a Microsoft.ActiveDirectory.Management.ADComputer object for every computer in the OU. Each object contains the computer host name in the DnsHostName property. The output of Get-ADComputer is piped to the built-in Select-Object PowerShell cmdlet; using the –ExpandProperty parameter isolates the DnsHostName property on each of the objects.

The result of this pipeline is the list of host names stored in the $hosts variable. This variable is used as the –Whitelist parameter value for the New-SplunkServerClass cmdlet.

The Get-ADComputer cmdlet is available as part of the Active Directory module for Windows PowerShell in Windows Server 2008 R2. For more information on the Active Directory module please see the following URL:

* http://go.microsoft.com/fwlink/?LinkID=139658

### See Also

* **Get-Help New-SplunkServerClass**
* **Get-Help Get-ADComputer**
* http://go.microsoft.com/fwlink/?LinkID=139658
* Create a Default Splunk Connection on page 11

## Add Hosts from an Active Directory Group to Server Class White List

### Problem

You want to add the hosts from an Active Directory group to a server class white list.

### Solution

Use the –LDAPFilter parameter of Get-ADComputer to isolate computers from the relevant group. Pass this list to the –Whitelist parameter of the New-SplunkServerClass Splunk module cmdlet.

PS > $hosts = Get-ADComputer -LDAPFilter `

‘(&(objectCategory=computer)(memberOf=cn=TestGroup, ou=Sales, `

dc=MyDomain,dc=com))’ | `

Select-Object *-ExpandProperty* DnsHostName

PS > New-SplunkServerClass -Name GpWhiteList -FilterType WhiteList -WhiteList $hosts

…

### Discussion

The –LDAPFilter parameter of Get-ADComputer accepts an LDAP search filter as defined in RFC2254. Using a filter expression that matches computers in the Active Directory group, the Get-ADComputer cmdlet returns a Microsoft.ActiveDirectory.Management.ADComputer object for every computer in the group. Each object contains the computer host name in the DnsHostName property. The output of Get-ADComputer is piped to the built-in Select-Object PowerShell cmdlet; using the –ExpandProperty parameter isolates the DnsHostName property on each of the objects.

The result of this pipeline is the list of host names stored in the $hosts variable. This variable is used as the –Whitelist parameter value for the New-SplunkServerClass cmdlet.

The Get-ADComputer cmdlet is available as part of the Active Directory module for Windows PowerShell in Windows Server 2008 R2. For more information on the Active Directory module please see the following URL:

* http://go.microsoft.com/fwlink/?LinkID=139658

For more information about LDAP search filters, please see:

* http://msdn.microsoft.com/en-us/library/aa746475.aspx
* http://www.ietf.org/rfc/rfc2254.txt

### See Also

* **Get-Help New-SplunkServerClass**
* **Get-Help Get-ADComputer**
* http://go.microsoft.com/fwlink/?LinkID=139658
* http://msdn.microsoft.com/en-us/library/aa746475.aspx
* http://www.ietf.org/rfc/rfc2254.txt
* Create a Default Splunk Connection on page 11

## Add List of Hosts from Splunk Search to Server Class White List

### Problem

You want to create a whitelist server class based on the results of a Splunk search.

### Solution

Use the Splunk-Search cmdlet to obtain the relevant search results (see View Event Data in a Table on page 19 for details on using Splunk-Search). Pipe these results to the standard PowerShell Select-Object cmdlet, using the –ExpandProperty and –Unique parameters to isolate unique values of the Host property:

Select-Object –ExpandProperty Host -Unique

Pass the resulting list to the –Whitelist parameter of the New-SplunkServerClass Splunk module cmdlet.

This example searches the current Splunk instance for the string “IIS”. The results of the search are used as the basis for a new whitelist server class named “IISWhiteList”:

PS > $hosts = Search-Splunk -Search "IIS" | `

Select-Object -ExpandProperty Host -Unique

PS > New-SplunkServerClass -Name IISWhiteList -FilterType WhiteList -WhiteList $hosts

…

### Discussion

The Search-Splunk cmdlet returns a Splunk.SDK.Search.OneshotResult object for each search result. Each of these objects exposes a Host property that identifies the computer that sourced the event matching the search.

Using the –Unique parameter of Select-Object ensures that a particular object value appears only once in the pipeline. As a result, the $hosts variable contains a list of unique host names from the Splunk search results. The $host variable is used as the –Whitelist parameter value for the New-SplunkServerClass cmdlet.

### See Also

* **Get-Help New-SplunkServerClass**
* **Get-Help Search-Splunk**
* **Get-Help Select-Object**
* Create a Default Splunk Connection on page 11
* View Event Data in a Table on page 19

## Add a List of VM Host Names from HyperV to a Server Class White List

### Problem

You want to query Hyper-V for a list of VM host names and add them to a server class white list.

### Solution

Use the Get-VMHost cmdlet from the PowerShell Management Library for Hyper-V to obtain a list of VM host names. Isolate the DnsHostName value from each host, passing the resulting list to the –Whitelist parameter of the New-SplunkServerClass Splunk module cmdlet.

This example creates a whitelist server class named “VMHWhiteList”. The whitelist for the server class contains the complete list of VM hosts from the current Hyper-V site.

PS > $hosts = Get-VMHost | Select-Object -ExpandProperty DnsHostName

PS > New-SplunkServerClass -Name VMHWhiteList -FilterType WhiteList -WhiteList $hosts

### Discussion

The Get-VMHost cmdlet is available as part of the PowerShell Management Library for Hyper-V. It returns a list of Microsoft.ActiveDirectory.Management.ADComputer objects representing the current list of VM host computers in the site. Each of these objects contains the computer host name in the DnsHostName property. The output of Get-VMHost is piped to the built-in Select-Object PowerShell cmdlet; using the –ExpandProperty parameter isolates the DnsHostName property on each of the objects.

The result of this pipeline is the list of host names stored in the $hosts variable. This variable is used as the –Whitelist parameter value for the New-SplunkServerClass cmdlet.

For more information about the PowerShell Management Library for Hyper-V, please see the following URL:

* http://pshyperv.codeplex.com/

### See Also

* **Get-Help New-SplunkServerClass**
* **Get-Help Select-Object**
* **Get-Help Get-VMHost**
* http://pshyperv.codeplex.com/
* Create a Default Splunk Connection on page 11

# Phase 2

## Apply an Input Configuration to Multiple Forwarders across all Hosts in an AD Site.

### Problem

You want to apply a specific input configuration to every host in an Active Directory site.

### Solution

Pipe the output of the Active Directory module Get-ADComputer cmdlet to the relevant Splunk input cmdlet. For instance, to create a new Windows performance monitor input, use the New-SplunkInputWinPerfmon cmdlet.

PS > Get-ADComputer –Filter \* | New-SplunkInputWinPerfMon -name ‘processes’ `

-interval 30 -object 'process' -counters 'elapsed time' -instances \*

…

### Discussion

By specifying the –Filter parameter as a match-all wildcard (\*), the Get-ADComputer cmdlet will return every computer object from Active Directory. These computer objects are then piped to the New-SplunkInputWinPerfmon cmdlet from the Splunk module.

New-SplunkInputWinPerfmon will create a new Splunk input on each computer specified in the pipeline. There are many related cmdlets for creating and managing various types of Splunk inputs; to see a complete list of these Input management cmdlets, use the following command:

PS > Get-Command –module Splunk \*input\*

The Get-ADComputer cmdlet is available as part of the Active Directory module for Windows PowerShell in Windows Server 2008 R2. For more information on the Active Directory module please see the following URL:

* http://go.microsoft.com/fwlink/?LinkID=139658

### See Also

* **Get-Help Test-Splunkd**
* **Get-Command –module Splunk \*input\***
* http://go.microsoft.com/fwlink/?LinkID=139658
* Query the Splunk Module Features on page 7
* Get Help for a Splunk Module Cmdlet on page 8
* Create a Default Splunk Connection on page 11

## Apply an Output Configuration to Multiple Forwarders across all Hosts in an AD Site.

### Problem

You want to apply a specific output configuration to every host in an Active Directory site.

### Solution

Pipe the output of the Active Directory module Get-ADComputer cmdlet to the relevant Splunk output cmdlet. For instance, to create a new data forwarder output, use the New-SplunkOutputServer cmdlet.

PS > Get-ADComputer –Filter \* | New-SplunkOutputServer –name Indexer:9000 `

-initialBackoff 30 –maxQueueSize 50

…

### Discussion

By specifying the –Filter parameter as a match-all wildcard (\*), the Get-ADComputer cmdlet will return every computer object from Active Directory. These computer objects are then piped to the New-SplunkOutputServer cmdlet from the Splunk module.

New-SplunkOutputServer will create a new Splunk forwarder configuration on each computer specified in the pipeline. There are many related cmdlets for creating and managing various types of Splunk outputs; to see a complete list of these Input management cmdlets, use the following command:

PS > Get-Command –module Splunk \*output\*

The Get-ADComputer cmdlet is available as part of the Active Directory module for Windows PowerShell in Windows Server 2008 R2. For more information on the Active Directory module please see the following URL:

* http://go.microsoft.com/fwlink/?LinkID=139658

### See Also

* **Get-Command –module Splunk \*output\***
* http://go.microsoft.com/fwlink/?LinkID=139658
* Query the Splunk Module Features on page 7
* Get Help for a Splunk Module Cmdlet on page 8
* Create a Default Splunk Connection on page 11

## Apply an Application Configuration to Multiple Forwarders across all Hosts in an AD Site.

### Problem

You want to apply a specific application configuration to every host in an Active Directory site.

### Solution

Pipe the output of the Active Directory module Get-ADComputer cmdlet to the relevant Splunk application management cmdlet. For instance, to create a new data forwarder output, use the New-SplunkApplication cmdlet.

PS > Get-ADComputer –Filter \* | New-SplunkApplication –name myApp `

-label “My Application” –manageable –visible –template “sample-App”

…

### Discussion

By specifying the –Filter parameter as a match-all wildcard (\*), the Get-ADComputer cmdlet will return every computer object from Active Directory. These computer objects are then piped to the New-SplunkApplication cmdlet from the Splunk module.

New-SplunkApplication will define a new Splunk application on each computer specified in the pipeline. There are many related cmdlets for creating and managing Splunk applications; to see a complete list of these Input management cmdlets, use the following command:

PS > Get-Command –module Splunk \*application\*

The Get-ADComputer cmdlet is available as part of the Active Directory module for Windows PowerShell in Windows Server 2008 R2. For more information on the Active Directory module please see the following URL:

* http://go.microsoft.com/fwlink/?LinkID=139658

### See Also

* **Get-Help Test-Splunkd**
* **Get-Command –module Splunk \*application\***
* http://go.microsoft.com/fwlink/?LinkID=139658
* Query the Splunk Module Features on page 7
* Get Help for a Splunk Module Cmdlet on page 8
* Create a Default Splunk Connection on page 11

## Install a Splunk Application to Multiple Forwarders across all Hosts in an AD site.

### Problem

You want to install a Splunk application to every host in an Active Directory site.

### Solution

Pipe the output of the Active Directory module Get-ADComputer cmdlet to the Install-SplunkApplication cmdlet.

PS > Get-ADComputer –Filter \* | Install-SplunkApplication `

–name ‘/Users/Zeus/Downloads/maps.tar.gz’

…

### Discussion

By specifying the –Filter parameter as a match-all wildcard (\*), the Get-ADComputer cmdlet will return every computer object from Active Directory. These computer objects are then piped to the Install-SplunkApplication cmdlet from the Splunk module.

Install-SplunkApplication will install the application from the file path or URL specified. It is important to note that a file path is *local to the Splunk instance on which you are installing the application*. In the current example, this implies that the path ‘/Users/Zeus/Downloads/maps.tar.gz’ exists on host for the current default Splunk connection.

The Get-ADComputer cmdlet is available as part of the Active Directory module for Windows PowerShell in Windows Server 2008 R2. For more information on the Active Directory module please see the following URL:

* http://go.microsoft.com/fwlink/?LinkID=139658

### See Also

* **Get-Help Install-SplunkApplication**
* http://go.microsoft.com/fwlink/?LinkID=139658
* Query the Splunk Module Features on page 7
* Get Help for a Splunk Module Cmdlet on page 8
* Create a Default Splunk Connection on page 11

## Apply an Input Configuration to Multiple Forwarders across all Hosts in an AD OU.

### Problem

You want to apply a specific input configuration to every host in an Active Directory Organizational Unit.

### Solution

Pipe the output of the Active Directory module Get-ADComputer cmdlet to the relevant Splunk input cmdlet. For instance, to create a new Windows performance monitor input, use the New-SplunkInputWinPerfmon cmdlet.

PS > $hosts = Get-ADComputer -Filter { DistinguishedName –like “\*OU=LocalOrgUnit\*” } `

| Select-Object –Expand DnsHostName

PS > $hosts | New-SplunkInputWinPerfMon -name ‘processes’ `

-interval 30 -object 'process' -counters 'elapsed time' -instances \*

…

### Discussion

The –Filter parameter of Get-ADComputer can be a filter expression. By creating a filter expression that matches the OU name in the computer’s DistinguishedName value, the Get-ADComputer cmdlet will return a Microsoft.ActiveDirectory.Management.ADComputer object for every computer in the OU. The output of Get-ADComputer is piped to the built-in Select-Object PowerShell cmdlet; using the –ExpandProperty parameter isolates the DnsHostName property on each of the objects.

The result of this pipeline is the list of host names stored in the $hosts variable. This variable is then piped to the New-SplunkInputWinPerfmon cmdlet from the Splunk module.

New-SplunkInputWinPerfmon will create a new Splunk input on each computer specified in the pipeline. There are many related cmdlets for creating and managing various types of Splunk inputs; to see a complete list of these Input management cmdlets, use the following command:

PS > Get-Command –module Splunk \*input\*

The Get-ADComputer cmdlet is available as part of the Active Directory module for Windows PowerShell in Windows Server 2008 R2. For more information on the Active Directory module please see the following URL:

* http://go.microsoft.com/fwlink/?LinkID=139658

### See Also

* **Get-Command –module Splunk \*input\***
* http://go.microsoft.com/fwlink/?LinkID=139658
* Query the Splunk Module Features on page 7
* Get Help for a Splunk Module Cmdlet on page 8
* Create a Default Splunk Connection on page 11

## Apply an Output Configuration to Multiple Forwarders across all Hosts in an AD OU.

### Problem

You want to apply a specific output configuration to every host in an Active Directory Organizational Unit.

### Solution

Pipe the output of the Active Directory module Get-ADComputer cmdlet to the relevant Splunk output cmdlet. For instance, to create a new data forwarder output, use the New-SplunkOutputServer cmdlet.

PS > $hosts = Get-ADComputer -Filter { DistinguishedName –like “\*OU=LocalOrgUnit\*” } `

| Select-Object –Expand DnsHostName

PS > $hosts | New-SplunkOutputServer –name Indexer:9000 `

-initialBackoff 30 –maxQueueSize 50

…

### Discussion

The –Filter parameter of Get-ADComputer can be a filter expression. By creating a filter expression that matches the OU name in the computer’s DistinguishedName value, the Get-ADComputer cmdlet will return a Microsoft.ActiveDirectory.Management.ADComputer object for every computer in the OU. The output of Get-ADComputer is piped to the built-in Select-Object PowerShell cmdlet; using the –ExpandProperty parameter isolates the DnsHostName property on each of the objects.

The result of this pipeline is the list of host names stored in the $hosts variable. This variable is then piped to the New-SplunkOutputServer cmdlet from the Splunk module.

New-SplunkOutputServer will create a new Splunk forwarder configuration on each computer specified in the pipeline. There are many related cmdlets for creating and managing various types of Splunk outputs; to see a complete list of these Input management cmdlets, use the following command:

PS > Get-Command –module Splunk \*output\*

The Get-ADComputer cmdlet is available as part of the Active Directory module for Windows PowerShell in Windows Server 2008 R2. For more information on the Active Directory module please see the following URL:

* http://go.microsoft.com/fwlink/?LinkID=139658

### See Also

* **Get-Command –module Splunk \*output\***
* http://go.microsoft.com/fwlink/?LinkID=139658
* Query the Splunk Module Features on page 7
* Get Help for a Splunk Module Cmdlet on page 8
* Create a Default Splunk Connection on page 11

## Apply an Application Configuration to Multiple Forwarders across all Hosts in an AD OU.

### Problem

You want to apply a specific application configuration to every host in an Active Directory Organizational Unit.

### Solution

Pipe the output of the Active Directory module Get-ADComputer cmdlet to the relevant Splunk application management cmdlet. For instance, to create a new data forwarder output, use the New-SplunkApplication cmdlet.

PS > $hosts = Get-ADComputer -Filter { DistinguishedName –like “\*OU=LocalOrgUnit\*” } `

| Select-Object –Expand DnsHostName

PS > $hosts | New-SplunkApplication –name myApp `

-label “My Application” –manageable –visible –template “sample-App”

…

### Discussion

The –Filter parameter of Get-ADComputer can be a filter expression. By creating a filter expression that matches the OU name in the computer’s DistinguishedName value, the Get-ADComputer cmdlet will return a Microsoft.ActiveDirectory.Management.ADComputer object for every computer in the OU. The output of Get-ADComputer is piped to the built-in Select-Object PowerShell cmdlet; using the –ExpandProperty parameter isolates the DnsHostName property on each of the objects.

The result of this pipeline is the list of host names stored in the $hosts variable. This variable is then piped to the New-SplunkApplication cmdlet from the Splunk module.

New-SplunkApplication will define a new Splunk application on each computer specified in the pipeline. There are many related cmdlets for creating and managing Splunk applications; to see a complete list of these Input management cmdlets, use the following command:

PS > Get-Command –module Splunk \*application\*

The Get-ADComputer cmdlet is available as part of the Active Directory module for Windows PowerShell in Windows Server 2008 R2. For more information on the Active Directory module please see the following URL:

* http://go.microsoft.com/fwlink/?LinkID=139658

### See Also

* **Get-Command –module Splunk \*application\***
* http://go.microsoft.com/fwlink/?LinkID=139658
* Query the Splunk Module Features on page 7
* Get Help for a Splunk Module Cmdlet on page 8
* Create a Default Splunk Connection on page 11

## Install a Splunk Application to Multiple Forwarders across all Hosts in an AD OU.

### Problem

You want to install a Splunk application to every host in an Active Directory Organization Unit.

### Solution

Pipe the output of the Active Directory module Get-ADComputer cmdlet to the Install-SplunkApplication cmdlet.

PS > $hosts = Get-ADComputer -Filter { DistinguishedName –like “\*OU=LocalOrgUnit\*” } `

| Select-Object –Expand DnsHostName

PS > $hosts | Install-SplunkApplication –name ‘/Users/Zeus/Downloads/maps.tar.gz’

…

### Discussion

The –Filter parameter of Get-ADComputer can be a filter expression. By creating a filter expression that matches the OU name in the computer’s DistinguishedName value, the Get-ADComputer cmdlet will return a Microsoft.ActiveDirectory.Management.ADComputer object for every computer in the OU. The output of Get-ADComputer is piped to the built-in Select-Object PowerShell cmdlet; using the –ExpandProperty parameter isolates the DnsHostName property on each of the objects.

The result of this pipeline is the list of host names stored in the $hosts variable. This variable is then piped to the Install-SplunkApplication cmdlet from the Splunk module.

Install-SplunkApplication will install the application from the file path or URL specified. It is important to note that a file path is *local to the Splunk instance on which you are installing the application*. In the current example, this implies that the path ‘/Users/Zeus/Downloads/maps.tar.gz’ exists on host for the current default Splunk connection.

The Get-ADComputer cmdlet is available as part of the Active Directory module for Windows PowerShell in Windows Server 2008 R2. For more information on the Active Directory module please see the following URL:

* http://go.microsoft.com/fwlink/?LinkID=139658

### See Also

* **Get-Help Install-SplunkApplication**
* http://go.microsoft.com/fwlink/?LinkID=139658
* Query the Splunk Module Features on page 7
* Get Help for a Splunk Module Cmdlet on page 8
* Create a Default Splunk Connection on page 11

## Apply an Input Configuration to Multiple Forwarders across all Hosts Active in the Domain.

### Problem

You want to apply a specific input configuration to every host in a specific domain.

### Solution

Pipe the output of the Active Directory module Get-ADComputer cmdlet to the relevant Splunk input cmdlet. For instance, to create a new Windows performance monitor input, use the New-SplunkInputWinPerfmon cmdlet.

PS > $hosts = Get-ADComputer -Filter \* -SearchBase “DC=MyDomain,DC=com” `

| Select-Object –Expand DnsHostName

PS > $hosts | New-SplunkInputWinPerfMon -name ‘processes’ `

-interval 30 -object 'process' -counters 'elapsed time' -instances \*

…

### Discussion

Using the –SearchBase parameter of Get-ADComputer defines the starting point for the ActiveDirectory search; this example searches the domain named “MyDomain.” By specifying the –Filter parameter as a match-all wildcard (\*), the Get-ADComputer cmdlet will return every computer object found under the specific -SearchBase. The output of Get-ADComputer is piped to the built-in Select-Object PowerShell cmdlet; using the –ExpandProperty parameter isolates the DnsHostName property on each of the objects.

The result of this pipeline is the list of host names stored in the $hosts variable. This variable is then piped to the New-SplunkInputWinPerfmon cmdlet from the Splunk module.

New-SplunkInputWinPerfmon will create a new Splunk input on each computer specified in the pipeline. There are many related cmdlets for creating and managing various types of Splunk inputs; to see a complete list of these Input management cmdlets, use the following command:

PS > Get-Command –module Splunk \*input\*

The Get-ADComputer cmdlet is available as part of the Active Directory module for Windows PowerShell in Windows Server 2008 R2. For more information on the Active Directory module please see the following URL:

* http://go.microsoft.com/fwlink/?LinkID=139658

### See Also

* **Get-Command –module Splunk \*input\***
* http://go.microsoft.com/fwlink/?LinkID=139658
* Query the Splunk Module Features on page 7
* Get Help for a Splunk Module Cmdlet on page 8
* Create a Default Splunk Connection on page 11

## Apply an Output Configuration to Multiple Forwarders across all Hosts Active in the Domain.

### Problem

You want to apply a specific application configuration to every host in a specific domain.

### Solution

Pipe the output of the Active Directory module Get-ADComputer cmdlet to the relevant Splunk application management cmdlet. For instance, to create a new data forwarder output, use the New-SplunkApplication cmdlet.

PS > $hosts = Get-ADComputer -Filter \* -SearchBase “DC=MyDomain,DC=com” `

| Select-Object –Expand DnsHostName

PS > $hosts | New-SplunkOutputServer –name Indexer:9000 `

-initialBackoff 30 –maxQueueSize 50

…

### Discussion

Using the –SearchBase parameter of Get-ADComputer defines the starting point for the ActiveDirectory search; this example searches the domain named “MyDomain.” By specifying the –Filter parameter as a match-all wildcard (\*), the Get-ADComputer cmdlet will return every computer object found under the specific -SearchBase. The output of Get-ADComputer is piped to the built-in Select-Object PowerShell cmdlet; using the –ExpandProperty parameter isolates the DnsHostName property on each of the objects.

The result of this pipeline is the list of host names stored in the $hosts variable. This variable is then piped to the New-SplunkOutputServer cmdlet from the Splunk module.

New-SplunkOutputServer will create a new Splunk forwarder configuration on each computer specified in the pipeline. There are many related cmdlets for creating and managing various types of Splunk outputs; to see a complete list of these Input management cmdlets, use the following command:

PS > Get-Command –module Splunk \*output\*

The Get-ADComputer cmdlet is available as part of the Active Directory module for Windows PowerShell in Windows Server 2008 R2. For more information on the Active Directory module please see the following URL:

* http://go.microsoft.com/fwlink/?LinkID=139658

### See Also

* **Get-Command –module Splunk \*application\***
* http://go.microsoft.com/fwlink/?LinkID=139658
* Query the Splunk Module Features on page 7
* Get Help for a Splunk Module Cmdlet on page 8
* Create a Default Splunk Connection on page 11

## Apply an Application Configuration to Multiple Forwarders across all Hosts Active in the Domain.

### Problem

You want to apply a specific application configuration to every host in a specific domain.

### Solution

Pipe the output of the Active Directory module Get-ADComputer cmdlet to the relevant Splunk application management cmdlet. For instance, to create a new data forwarder output, use the New-SplunkApplication cmdlet.

PS > $hosts = Get-ADComputer -Filter \* -SearchBase “DC=MyDomain,DC=com” `

| Select-Object –Expand DnsHostName

PS > $hosts | New-SplunkApplication –name myApp `

-label “My Application” –manageable –visible –template “sample-App”

…

### Discussion

Using the –SearchBase parameter of Get-ADComputer defines the starting point for the ActiveDirectory search; this example searches the domain named “MyDomain.” By specifying the –Filter parameter as a match-all wildcard (\*), the Get-ADComputer cmdlet will return every computer object found under the specific -SearchBase. The output of Get-ADComputer is piped to the built-in Select-Object PowerShell cmdlet; using the –ExpandProperty parameter isolates the DnsHostName property on each of the objects.

The result of this pipeline is the list of host names stored in the $hosts variable. This variable is then piped to the New-SplunkApplication cmdlet from the Splunk module.

New-SplunkApplication will define a new Splunk application on each computer specified in the pipeline. There are many related cmdlets for creating and managing Splunk applications; to see a complete list of these Input management cmdlets, use the following command:

PS > Get-Command –module Splunk \*application\*

The Get-ADComputer cmdlet is available as part of the Active Directory module for Windows PowerShell in Windows Server 2008 R2. For more information on the Active Directory module please see the following URL:

* http://go.microsoft.com/fwlink/?LinkID=139658

### See Also

* **Get-Command –module Splunk \*application\***
* http://go.microsoft.com/fwlink/?LinkID=139658
* Query the Splunk Module Features on page 7
* Get Help for a Splunk Module Cmdlet on page 8
* Create a Default Splunk Connection on page 11

## Install a Splunk Application to Multiple Forwarders across all Hosts Active in the Domain.

### Problem

You want to install a Splunk application to every active host in a specific domain.

### Solution

Pipe the output of the Active Directory module Get-ADComputer cmdlet to the Install-SplunkApplication cmdlet.

PS > $hosts = Get-ADComputer -Filter \* -SearchBase “DC=MyDomain,DC=com” `

| Select-Object –Expand DnsHostName

PS > $hosts | Install-SplunkApplication –name ‘/Users/Zeus/Downloads/maps.tar.gz’

…

### Discussion

Using the –SearchBase parameter of Get-ADComputer defines the starting point for the ActiveDirectory search; this example searches the domain named “MyDomain.” By specifying the –Filter parameter as a match-all wildcard (\*), the Get-ADComputer cmdlet will return every computer object found under the specific -SearchBase. The output of Get-ADComputer is piped to the built-in Select-Object PowerShell cmdlet; using the –ExpandProperty parameter isolates the DnsHostName property on each of the objects.

The result of this pipeline is the list of host names stored in the $hosts variable. This variable is then piped to the Install-SplunkApplication cmdlet from the Splunk module.

Install-SplunkApplication will install the application from the file path or URL specified. It is important to note that a file path is *local to the Splunk instance on which you are installing the application*. In the current example, this implies that the path ‘/Users/Zeus/Downloads/maps.tar.gz’ exists on host for the current default Splunk connection.

The Get-ADComputer cmdlet is available as part of the Active Directory module for Windows PowerShell in Windows Server 2008 R2. For more information on the Active Directory module please see the following URL:

* http://go.microsoft.com/fwlink/?LinkID=139658

### See Also

* **Get-Help Install-SplunkApplication**
* http://go.microsoft.com/fwlink/?LinkID=139658
* Query the Splunk Module Features on page 7
* Get Help for a Splunk Module Cmdlet on page 8
* Create a Default Splunk Connection on page 11

## Run Tasks that Impact Multiple Hosts in Parallel.

### Problem

You want to apply Splunk configuration updates to multiple machines in parallel.

### Solution

Execute the relevant Splunk Resource Kit cmdlet as a PowerShell job using the Start-Job standard PowerShell cmdlet.

PS > Get-ADComputer –Filter \* | Start-Job {

Import-Module Splunk;

$args | Connect-Splunk;

New-SplunkOutputServer –name “Indexer:9000” -initialBackoff 30 `

–maxQueueSize 50;

} –ArgumentList ( Get-SplunkConnectionObject )

…

### Discussion

This example is an adaptation of the solution to Apply an Output Configuration to Multiple Forwarders across all Hosts in an AD Site on page 33. In the original example, the list of hosts returned from the Get-ADComputer cmdlet is processed iteratively, such that the first host must be processed before the next, as so on. In the current solution, the built-in Start-Job PowerShell cmdlet is used to execute the output configuration asynchronously for each machine.

Start-Job accepts a script block parameter that defines the complete asynchronous operation:

PS > Get-ADComputer –Filter \* | Start-Job { `

Import-Module Splunk; `

$args | Connect-Splunk; `

New-SplunkOutputServer –name “Indexer:9000” -initialBackoff 30 `

–maxQueueSize 50; `

} –ArgumentList ( Get-SplunkConnectionObject )

…

Note that this script block is executed in a new PowerShell session; the current session modules and variables are not available in the job’s new session. In order to make the current example execute successfully, it is necessary to import the Splunk module inside of the job’s script block.

In addition, no default connection exists in the example job’s PowerShell session. In this example, the default connection object obtained using the Get-SplunkConnectionObject cmdlet from the Splunk PowerShell Resource Kit; it is shared as a job argument with the new session and used to initialize the session’s default connection using the Connect-Splunk cmdlet:

PS > Get-ADComputer –Filter \* | Start-Job { `

Import-Module Splunk; `

$args | Connect-Splunk; `

New-SplunkOutputServer –name “Indexer:9000” -initialBackoff 30 `

–maxQueueSize 50; `

} –ArgumentList ( Get-SplunkConnectionObject )

…

One job will be created for each host returned from Get-ADComputer. The Start-Job cmdlet will return these jobs to the console and the scripts attached to each job will execute in the background. To check on the status of the jobs, use the standard PowerShell Get-Job cmdlet:

PS > Get-Job

Id Name State HasMoreData … Command

-- ---- ----- ----------- … -------

2 Job2 Running False … …

1 Job1 Completed False … …

…

The state of each background job will update as they complete execution. To retrieve the results of completed jobs, use the standard Receive-Job PowerShell cmdlet:

PS > Get-Job | Receive-Job

…

Finally, to remove completed jobs from the running job list, use the Remove-Job built-in PowerShell cmdlet:

PS > Remove-Job –state Completed

…

### See Also

* **Get-Help about\_jobs**
* **Get-Help about\_script\_blocks**
* **Get-Help Start-Job**
* **Get-Help Get-Job**
* **Get-Help Receive-Job**
* **Get-Help Remove-Job**
* **Get-Help Get-SplunkConnectionObject**
* Import the Splunk Module on page 7
* Create a Default Splunk Connection on page 11

## Forward Splunk Alerts to SCOM

### Problem

You want to integrate alerts from Splunk with your local SCOM instance.

### Solution

Follow the instructions for creating an SCOM monitor based on a Windows PowerShell script at the following URL:

* http://technet.microsoft.com/en-us/library/ff381420.aspx

Modify the PowerShell script in the example to pull data from your Splunk indexer. In this example, a hard search is made against the Splunk indexer to find any instances of the string “error” found in the last five minutes; the SCOM monitor reports a bad result if any instance of the string “error” is found in the Splunk indexes.

param($computerName)

import-module Splunk;

$credential = c:\tools\load-credentials.ps1;

connect-splunk -computername ‘Splunkery’ `

-protocol 'https' `

-port 8089 `

-credentials $credential;

$fiveMinutesAgo = ( get-date ).addMinutes( -5 ).toString( ‘s’ );

$results = search-splunk –search ‘error’ –starttime $fiveMinutesAgo

$testSuccessful = -not $results;

$api = new-object -comObject 'MOM.ScriptAPI'

$api.LogScriptEvent('MyScript.ps1',20,4,$computerName)

$bag = $api.CreatePropertyBag()

$bag.AddValue('ComputerName',$computerName)

if ($testSuccessful -eq $true)

{$bag.AddValue('Result','Good')}

else

{$bag.AddValue('Result','Bad')}

$bag

### Discussion

This example assumes you are using an instance of SCOM that supports PowerShell v2.0 or later.

Because SCOM modules are run in an unattended session, the credentials required by the Connect-Splunk cmdlet must be provided from a stored cache. In this example, the credential object is created by a script file named load-credentials.ps1; this file would manually create the credential necessary to connect to the Splunk indexer:

# load-credentials.ps1

New-Object System.Management.Automation.PSCredential(

‘searcher’,

( ConvertTo-SecureString *-String* ‘password’*-AsPlainText* *-Force* )

);

Using the standard New-Object PowerShell cmdlet, a new System.Management.Automation.PSCredential instance is created, with a username of “searcher” and a password of “password”. PSCredential objects store passwords encrypted in memory; it is necessary to use the standard ConvertTo-SecureString cmdlet to transform the plaintext password into an encrypted blob the PSCredential object will accept.

The Splunk-Search cmdlet can limit searches by time. In this solution only the last five minutes of indexed data is searched for the string “error”. The built-in get-date PowerShell cmdlet returns a System.DateTime object representing the current system time. Using the AddMinutes method of the datetime object allows us to set the time the object represents to five minutes in the past. The ToString method formats the datetime instant as an ISO-8601 time string suitable for application in the Search-Splunk cmdlet.

If any results are returned from the search, the module will indicate that the module task has failed for the specified target computer. This is accomplished using the MOM Scripting API library; for more information on this API, refer to this URL:

* http://msdn.microsoft.com/en-us/library/bb437523.aspx

### See Also

* http://msdn.microsoft.com/en-us/library/bb437523.aspx
* http://technet.microsoft.com/en-us/library/ff381420.aspx
* **Get-Help Search-Splunk**
* Import the Splunk Module on page 7
* Create a Default Splunk Connection on page 11

## Forward SCOM alerts to Splunk.