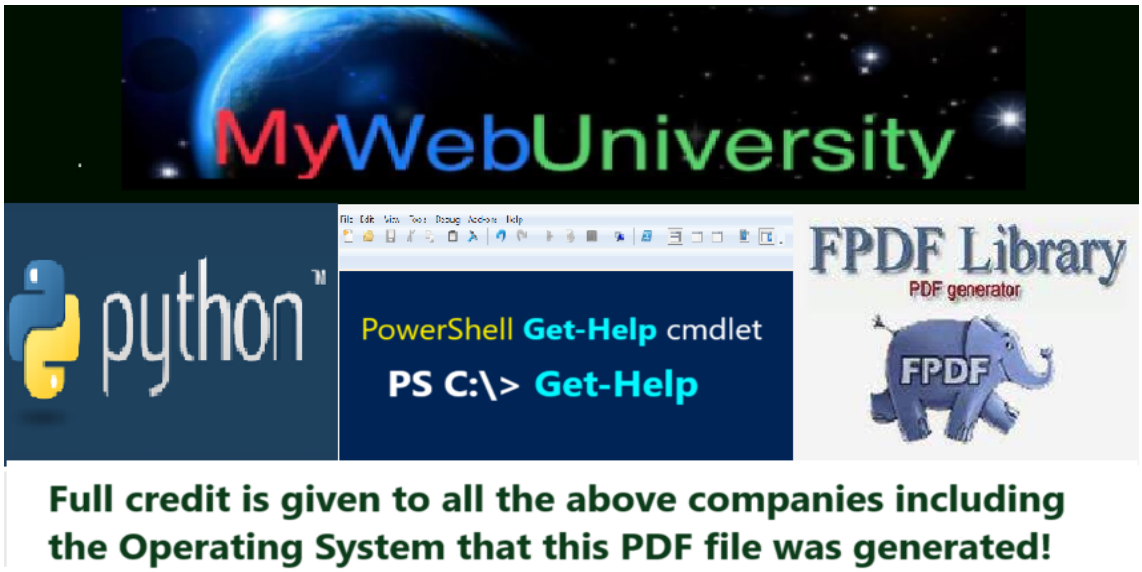

Windows PowerShell Get-Help on Cmdlet 'New-PSSession'

PS:\>Get-HELP New-PSSession -Full

NAME

New-PSSession

SYNOPSIS

Creates a persistent connection to a local or remote computer.

SYNTAX

New-PSSession [-ConnectionUri] <System.Uri[]> [-AllowRedirection] [-Authentication {Default | Basic | Negotiate | NegotiateWithImplicitCredential | Credssp | Digest |

Kerberos}] [-CertificateThumbprint <System.String>] [-ConfigurationName <System.String>] [-Credential <System.Management.Automation.PSCredential>]

[-EnableNetworkAccess] [-Name <System.String[]>] [-SessionOption <System.Management.Automation.Remoting.PSSessionOption>] [-ThrottleLimit <System.Int32>]

[<CommonParameters>]

New-PSSession [[-ComputerName] <System.String[]>] [-ApplicationName <System.String>] [-Authentication {Default | Basic | Negotiate | NegotiateWithImplicitCredential |

Credssp | Digest | Kerberos}] [-CertificateThumbprint <System.String>] [-ConfigurationName <System.String>] [-Credential <System.Management.Automation.PSCredential>]

[-EnableNetworkAccess] [-Name <System.String[]>] [-Port <System.Int32>] [-SessionOption <System.Management.Automation.Remoting.PSSessionOption>] [-ThrottleLimit
<System.Int32>] [-UseSSL] [<CommonParameters>]

New-PSSession [-VMId] <System.Guid[]> [-ConfigurationName <System.String>] [-Credential <System.Management.Automation.PSCredential>] [-Name <System.String[]>]
[-ThrottleLimit <System.Int32>] [<CommonParameters>]

New-PSSession [-ConfigurationName <System.String>] [-Credential <System.Management.Automation.PSCredential>] [-Name <System.String[]>] [-ThrottleLimit <System.Int32>]
-VMName <System.String[]> [<CommonParameters>]

New-PSSession [-ConfigurationName <System.String>] -ContainerId <System.String[]> [-Name <System.String[]>] [-RunAsAdministrator] [-ThrottleLimit <System.Int32>]
[<CommonParameters>]

New-PSSession [[-Session] <System.Management.Automation.Runspaces.PSSession[]>] [-EnableNetworkAccess] [-Name <System.String[]>] [-ThrottleLimit <System.Int32>]
[<CommonParameters>]

DESCRIPTION

The `New-PSSession` cmdlet creates a PowerShell session (PSSession) on a local or remote computer. When you create a PSSession , PowerShell establishes a persistent
connection to the remote computer.

Use a PSSession to run multiple commands that share data, such as a function or the value of a variable. To run commands in a PSSession , use the `Invoke-Command`
cmdlet. To use the PSSession to interact directly with a remote computer, use the `Enter-PSSession` cmdlet. For more information, see about_PSSessions
(about/about_PSSessions.md).

You can run commands on a remote computer without creating a PSSession by using the ComputerName parameters of

`Enter-PSSession` or `Invoke-Command`. When you use the

ComputerName parameter, PowerShell creates a temporary connection that is used for the command and is then closed.

PARAMETERS

-AllowRedirection <System.Management.Automation.SwitchParameter>

Indicates that this cmdlet allows redirection of this connection to an alternate Uniform Resource Identifier (URI).

When you use the ConnectionURI parameter, the remote destination can return an instruction to redirect to a different URI. By default, PowerShell does not

redirect connections, but you can use this parameter to enable it to redirect the connection.

You can also limit the number of times the connection is redirected by changing the MaximumConnectionRedirectionCount session option value. Use the

MaximumRedirection parameter of the `New-PSSessionOption` cmdlet or set the MaximumConnectionRedirectionCount property of the $PSSessionOption preference

variable. The default value is `5`.

Required? false

Position? named

Default value False

Accept pipeline input? False

Accept wildcard characters? false

-ApplicationName <System.String>

Specifies the application name segment of the connection URI. Use this parameter to specify the application name when you are not using the ConnectionURI

parameter in the command.

The default value is the value of the `$PSSessionApplicationName` preference variable on the local computer. If this preference variable is not defined, the

default value is `WSMAN`. This value is appropriate for most uses. For more information, see about_Preference_Variables (About/about_Preference_Variables.md).

The WinRM service uses the application name to select a listener to service the connection request. The value of this parameter should match the value of the

URLPrefix property of a listener on the remote computer.

Required? false

Position? named

Default value None

Accept pipeline input? True (ByPropertyName)

Accept wildcard characters? false

-Authentication <System.Management.Automation.Runspaces.AuthenticationMechanism>

Specifies the mechanism that is used to authenticate the user's credentials. The acceptable values for this parameter are:

- `Default`

- `Basic`

- `Credssp`

- `Digest`

- `Kerberos`

- `Negotiate`

- `NegotiateWithImplicitCredential`

The default value is `Default`.

For more information about the values of this parameter, see AuthenticationMechanism Enumeration

(/dotnet/api/system.management.automation.runspaces.authenticationmechanism).

> [!CAUTION] > Credential Security Support Provider (CredSSP) authentication, in which the user credentials are > passed to a remote computer to be authenticated,

is designed for commands that require > authentication on more than one resource, such as accessing a remote network share. This mechanism > increases the

security risk of the remote operation. If the remote computer is compromised, the > credentials that are passed to it can be used to control the network session.

Required? false

Position? named

Default value None

Accept pipeline input? False

Accept wildcard characters? false

-CertificateThumbprint <System.String>

Specifies the digital public key certificate (X509) of a user account that has permission to perform this action. Enter the certificate thumbprint of the

certificate.

Certificates are used in client certificate-based authentication. They can be mapped only to local user accounts; they do not work with domain accounts.

To get a certificate, use the `Get-Item` or `Get-ChildItem` command in the PowerShell `Cert:` drive.

Required? false

Position? named

Default value None

Accept pipeline input? False

Accept wildcard characters? false

-ComputerName <System.String[]>

Specifies an array of names of computers. This cmdlet creates a persistent connection (PSSession) to the specified

computer. If you enter multiple computer

names, `New-PSSession` creates multiple PSSession objects, one for each computer. The default is the local computer.

Type the NetBIOS name, an IP address, or a fully qualified domain name of one or more remote computers. To specify the local computer, type the computer name,

`localhost`, or a dot (`.`). When the computer is in a different domain than the user, the fully qualified domain name is required. You can also pipe a computer

name, in quotation marks, to `New-PSSession`.

To use an IP address in the value of the ComputerName parameter, the command must include the Credential parameter. Also, the computer must be configured for

HTTPS transport or the IP address of the remote computer must be included in the WinRM TrustedHosts list on the local computer. For instructions for adding a

computer name to the TrustedHosts list, see "How to Add a Computer to the Trusted Host List" in about_Remote_Troubleshooting

(about/about_Remote_Troubleshooting.md).

To include the local computer in the value of the ComputerName parameter, start Windows PowerShell by using the Run as administrator option .

Required? false

Position? 0

Default value None

Accept pipeline input? True (ByPropertyName, ByValue)

Accept wildcard characters? false

-ConfigurationName <System.String>

Specifies the session configuration that is used for the new PSSession .

Enter a configuration name or the fully qualified resource URI for a session configuration. If you specify only the configuration name, the following schema URI

is prepended: `http://schemas.microsoft.com/PowerShell`.

The session configuration for a session is located on the remote computer. If the specified session configuration does not exist on the remote computer, the

command fails.

The default value is the value of the `$PSSessionConfigurationName` preference variable on the local computer. If this preference variable is not set, the default

is `Microsoft.PowerShell`. For more information, see about_Preference_Variables (About/about_Preference_Variables.md).

Required? false

Position? named

Default value None

Accept pipeline input? True (ByPropertyName)

Accept wildcard characters? false

-ConnectionUri <System.Uri[]>

Specifies a URI that defines the connection endpoint for the session. The URI must be fully qualified. The format of this string is as follows:

`<Transport>://<ComputerName>:<Port>/<ApplicationName>`

The default value is as follows:

`http://localhost:5985/WSMAN`

If you do not specify a ConnectionURI , you can use the UseSSL , ComputerName , Port , and ApplicationName parameters to specify the ConnectionURI values.

Valid values for the Transport segment of the URI are HTTP and HTTPS. If you specify a connection URI with a Transport segment, but do not specify a port, the

session is created with standards ports: `80` for HTTP and `443` for HTTPS. To use the default ports for PowerShell remoting, specify port `5985` for HTTP or

`5986` for HTTPS.

If the destination computer redirects the connection to a different URI, PowerShell prevents the redirection unless you use the AllowRedirection parameter in the

command.

Required? true

Position? 0

Default value None

Accept pipeline input? True (ByPropertyName)

Accept wildcard characters? false

-ContainerId <System.String[]>

Specifies an array of IDs of containers. This cmdlet starts an interactive session with each of the specified containers. Use the `docker ps` command to get a

list of container IDs. For more information, see the help for the docker ps (https://docs.docker.com/engine/reference/commandline/ps/)command.

Required? true

Position? named

Default value None

Accept pipeline input? True (ByPropertyName)

Accept wildcard characters? false

-Credential <System.Management.Automation.PSCredential>

Specifies a user account that has permission to do this action. The default is the current user.

Type a user name, such as `User01` or `Domain01\User01`, or enter a PSCredential object generated by the `Get-Credential` cmdlet. If you type a user name, you're

prompted to enter the password.

Credentials are stored in a PSCredential (/dotnet/api/system.management.automation.pscredential)object and the password is stored as a SecureString

(/dotnet/api/system.security.securestring).

> [!NOTE] > For more information about SecureString data protection, see > How secure is SecureString?
(/dotnet/api/system.security.securestring#how-secure-is-securestring).

Required? false
Position? named
Default value Current user
Accept pipeline input? True (ByPropertyName)
Accept wildcard characters? false

-EnableNetworkAccess <System.Management.Automation.SwitchParameter>

Indicates that this cmdlet adds an interactive security token to loopback sessions. The interactive token lets you run commands in the loopback session that get
data from other computers. For example, you can run a command in the session that copies XML files from a remote computer to the local computer.

A loopback session is a PSSession that originates and ends on the same computer. To create a loopback session, omit the ComputerName parameter or set its value to
dot (`.`), `localhost`, or the name of the local computer.

By default, this cmdlet creates loopback sessions by using a network token, which might not provide sufficient permission to authenticate to remote computers.

The EnableNetworkAccess parameter is effective only in loopback sessions. If you use EnableNetworkAccess when you create a session on a remote computer, the
command succeeds, but the parameter is ignored.

You can also enable remote access in a loopback session by using the `CredSSP` value of the Authentication parameter, which delegates the session credentials to
other computers.

To protect the computer from malicious access, disconnected loopback sessions that have interactive tokens, which

are those created by using the

EnableNetworkAccess parameter, can be reconnected only from the computer on which the session was created. Disconnected sessions that use CredSSP authentication

can be reconnected from other computers. For more information, see `Disconnect-PSSession`.

This parameter was introduced in PowerShell 3.0.

Required? false

Position? named

Default value False

Accept pipeline input? False

Accept wildcard characters? false

-Name <System.String[]>

Specifies a friendly name for the PSSession .

You can use the name to refer to the PSSession when you use other cmdlets, such as `Get-PSSession` and `Enter-PSSession`. The name is not required to be unique to

the computer or the current session.

Required? false

Position? named

Default value None

Accept pipeline input? False

Accept wildcard characters? false

-Port <System.Int32>

Specifies the network port on the remote computer that is used for this connection. To connect to a remote computer, the remote computer must be listening on the

port that the connection uses. The default ports are `5985`, which is the WinRM port for HTTP, and `5986`, which is the WinRM port for HTTPS.

Before using another port, you must configure the WinRM listener on the remote computer to listen at that port. Use the

following commands to configure the

listener:

1. `winrm delete winrm/config/listener?Address=*+Transport=HTTP` 2. `winrm create winrm/config/listener?Address=*+Transport=HTTP @{Port="<port-number>"}`

Do not use the Port parameter unless you must. The port setting in the command applies to all computers or sessions on which the command runs. An alternate port

setting might prevent the command from running on all computers.

Required? false

Position? named

Default value None

Accept pipeline input? False

Accept wildcard characters? false

-RunAsAdministrator <System.Management.Automation.SwitchParameter>

Indicates that the PSSession runs as administrator.

Required? false

Position? named

Default value False

Accept pipeline input? False

Accept wildcard characters? false

-Session <System.Management.Automation.Runspaces.PSSession[]>

Specifies an array of PSSession objects that this cmdlet uses as a model for the new PSSession . This parameter creates new PSSession objects that have the same

properties as the specified PSSession objects.

Enter a variable that contains the PSSession objects or a command that creates or gets the PSSession objects, such as a `New-PSSession` or `Get-PSSession` command.

The resulting PSSession objects have the same computer name, application name, connection URI, port, configuration name, throttle limit, and Secure Sockets Layer

(SSL) value as the originals, but they have a different display name, ID, and instance ID (GUID).

Required? false

Position? 0

Default value None

Accept pipeline input? True (ByPropertyName, ByValue)

Accept wildcard characters? false

-SessionOption <System.Management.Automation.Remoting.PSSessionOption>

Specifies advanced options for the session. Enter a SessionOption object, such as one that you create by using the `New-PSSessionOption` cmdlet, or a hash table

in which the keys are session option names and the values are session option values.

The default values for the options are determined by the value of the `$PSSessionOption` preference variable, if it is set. Otherwise, the default values are

established by options set in the session configuration.

The session option values take precedence over default values for sessions set in the `$PSSessionOption` preference variable and in the session configuration.

However, they do not take precedence over maximum values, quotas or limits set in the session configuration.

For a description of the session options that includes the default values, see `New-PSSessionOption`. For information about the `$PSSessionOption` preference

variable, see about_Preference_Variables (About/about_Preference_Variables.md). For more information about session configurations, see

about_Session_Configurations (About/about_Session_Configurations.md).

Required? false

Position? named

Default value None

Accept pipeline input? False

Accept wildcard characters? false

-ThrottleLimit <System.Int32>

Specifies the maximum number of concurrent connections that can be established to run this command. If you omit this parameter or enter a value of `0` (zero), the

default value, `32`, is used.

The throttle limit applies only to the current command, not to the session or to the computer.

Required? false

Position? named

Default value None

Accept pipeline input? False

Accept wildcard characters? false

-UseSSL <System.Management.Automation.SwitchParameter>

Indicates that this cmdlet uses the SSL protocol to establish a connection to the remote computer. By default, SSL is not used.

WS-Management encrypts all PowerShell content transmitted over the network. The UseSSL parameter offers an additional protection that sends the data across an

HTTPS connection instead of an HTTP connection.

If you use this parameter, but SSL is not available on the port that is used for the command, the command fails.

Required? false

Position? named

Default value False

Accept pipeline input? False

Accept wildcard characters? false

-VMId <System.Guid[]>

Specifies an array of virtual machine IDs. This cmdlet starts a PowerShell Direct interactive session with each of the

specified virtual machines. For more information, see Virtual Machine automation and management using PowerShell (/virtualization/hyper-v-on-windows/user-guide/powershell-direct).

Use `Get-VM` to see the virtual machines that are available on your Hyper-V host.

Required? true
Position? 0
Default value None
Accept pipeline input? True (ByPropertyName)
Accept wildcard characters? false

-VMName <System.String[]>

Specifies an array of names of virtual machines. This cmdlet starts a PowerShell Direct interactive session with each of the specified virtual machines. For more information, see Virtual Machine automation and management using PowerShell (/virtualization/hyper-v-on-windows/user-guide/powershell-direct).

Use `Get-VM` to see the virtual machines that are available on your Hyper-V host.

Required? true
Position? named
Default value None
Accept pipeline input? True (ByPropertyName)
Accept wildcard characters? false

<CommonParameters>

This cmdlet supports the common parameters: Verbose, Debug,
ErrorAction, ErrorVariable, WarningAction, WarningVariable,
OutBuffer, PipelineVariable, and OutVariable. For more information, see
about_CommonParameters (https:/go.microsoft.com/fwlink/?LinkID=113216).

INPUTS

System.String

You can pipe a string to this cmdlet.

System.URI

You can pipe a URI to this cmdlet.

System.Management.Automation.Runspaces.PSSession

You can pipe a session object to this cmdlet.

OUTPUTS

System.Management.Automation.Runspaces.PSSession

NOTES

Windows PowerShell includes the following aliases for `New-PSSession`:

- `nsn`

- This cmdlet uses the PowerShell remoting infrastructure. To use this cmdlet, the local computer and any remote computers must be configured for PowerShell remoting. For more information, see about_Remote_Requirements (About/about_Remote_Requirements.md). - To create a PSSession on the local computer, start PowerShell with the **Run as administrator** option. - When you are finished with the PSSession , use the `Remove-PSSession` cmdlet to delete the PSSession and release its resources.

------ Example 1: Create a session on the local computer ------

$s = New-PSSession

This command creates a new PSSession on the local computer and saves the PSSession in the `$s` variable.

You can now use this PSSession to run commands on the local computer.

------- Example 2: Create a session on a remote computer -------

$Server01 = New-PSSession -ComputerName Server01

This command creates a new PSSession on the Server01 computer and saves it in the `$Server01` variable.

When creating multiple PSSession objects, assign them to variables with useful names. This will help you manage the PSSession objects in subsequent commands.

------- Example 3: Create sessions on multiple computers -------

$s1, $s2, $s3 = New-PSSession -ComputerName Server01,Server02,Server03

This command creates three PSSession objects, one on each of the computers specified by the ComputerName parameter.

The command uses the assignment operator (`=`) to assign the new PSSession objects to variables: `$s1`, `$s2`, `$s3`. It assigns the Server01 PSSession to `$s1`, the

Server02 PSSession to `$s2`, and the Server03 PSSession to `$s3`.

When you assign multiple objects to a series of variables, PowerShell assigns each object to a variable in the series respectively. If there are more objects than

variables, all remaining objects are assigned to the last variable. If there are more variables than objects, the remaining variables are empty (`$null`).

------ Example 4: Create a session with a specified port ------

New-PSSession -ComputerName Server01 -Port 8081 -UseSSL -ConfigurationName E12

This command creates a new PSSession on the Server01 computer that connects to server port `8081` and uses the SSL protocol. The new PSSession uses an alternative

session configuration called `E12`.

Before setting the port, you must configure the WinRM listener on the remote computer to listen on port 8081. For more information, see the description of the Port

parameter.

--- Example 5: Create a session based on an existing session ---

New-PSSession -Session $s -Credential Domain01\User01

This command creates a PSSession with the same properties as an existing PSSession . You can use this command format when the resources of an existing PSSession are

exhausted and a new PSSession is needed to offload some of the demand.

The command uses the Session parameter of `New-PSSession` to specify the PSSession saved in the `$s` variable. It uses the credentials of the `Domain1\Admin01` user

to complete the command.

Example 6: Create a session with a global scope in a different domain

$global:s = New-PSSession -ComputerName Server1.Domain44.Corpnet.Fabrikam.com -Credential Domain01\Admin01

This example shows how to create a PSSession with a global scope on a computer in a different domain.

By default, PSSession objects created at the command line are created with local scope and PSSession objects created in a script have script scope.

To create a PSSession with global scope, create a new PSSession and then store the PSSession in a variable that is cast to a global scope. In this case, the `$s`

variable is cast to a global scope.

The command uses the ComputerName parameter to specify the remote computer. Because the computer is in a different domain than the user account, the full name of the

computer is specified together with the credentials of the user.

-------- Example 7: Create sessions for many computers --------

$rs = Get-Content C:\Test\Servers.txt | New-PSSession -ThrottleLimit 50

This command creates a PSSession on each of the 200 computers listed in the `Servers.txt` file and it stores the resulting PSSession in the `$rs` variable. The

PSSession objects have a throttle limit of `50`.

You can use this command format when the names of computers are stored in a database, spreadsheet, text file, or other text-convertible format.

---------- Example 8: Create a session by using a URI ----------

$s = New-PSSession -URI http://Server01:91/NewSession -Credential Domain01\User01

This command creates a PSSession on the Server01 computer and stores it in the `$s` variable. It uses the URI parameter to specify the transport protocol, the remote

computer, the port, and an alternate session configuration. It also uses the Credential parameter to specify a user account that has permission to create a session on

the remote computer.

----- Example 9: Run a background job in a set of sessions -----

$s = New-PSSession -ComputerName (Get-Content Servers.txt) -Credential Domain01\Admin01 -ThrottleLimit 16

Invoke-Command -Session $s -ScriptBlock {Get-Process PowerShell} -AsJob

These commands create a set of PSSession objects and then run a background job in each of the PSSession objects.

The first command creates a new PSSession on each of the computers listed in the `Servers.txt` file. It uses the `New-PSSession` cmdlet to create the PSSession . The

value of the ComputerName parameter is a command that uses the `Get-Content` cmdlet to get the list of computer names the `Servers.txt` file.

The command uses the Credential parameter to create the PSSession objects that have the permission of a domain administrator, and it uses the ThrottleLimit parameter

to limit the command to `16` concurrent connections. The command saves the PSSession objects in the `$s` variable.

The second command uses the AsJob parameter of the `Invoke-Command` cmdlet to start a background job that runs a `Get-Process PowerShell` command in each of the

PSSession objects in `$s`.

For more information about PowerShell background jobs, see about_Jobs (About/about_Jobs.md) and [about_Remote_Jobs](About/about_Remote_Jobs.md).

- Example 10: Create a session for a computer by using its URI -

New-PSSession -ConnectionURI https://management.exchangelabs.com/Management

This command creates a PSSession objects that connects to a computer that is specified by a URI instead of a computer name.

---------- Example 11: Create a session option ----------

$so = New-PSSessionOption -SkipCACheck

New-PSSession -ConnectionUri https://management.exchangelabs.com/Management -SessionOption $so -Credential Server01\Admin01

This example shows how to create a session option object and use the SessionOption parameter.

The first command uses the `New-PSSessionOption` cmdlet to create a session option. It saves the resulting SessionOption object in the `$so` variable.

The second command uses the option in a new session. The command uses the `New-PSSession` cmdlet to create a new session. The value of the SessionOption parameter is

the SessionOption object in the `$so` variable.

RELATED LINKS

Online Version: https://learn.microsoft.com/powershell/module/microsoft.powershell.core/new-pssession?view=powershell-5.1&WT.mc_id=ps-gethelp

Connect-PSSession

Disconnect-PSSession

Enter-PSSession

Exit-PSSession

Get-PSSession

Invoke-Command

Receive-PSSession

Remove-PSSession