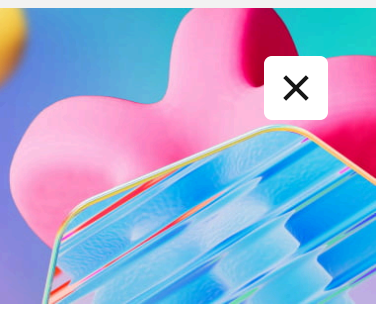

New-PSSession

Reference

Feedback

Module: Microsoft.PowerShell.Core

In this article

Creates a persistent connection to a local or remote computer.

Syntax

PowerShell Copy

```
New-PSSession
   [[-ComputerName] <String[]>]
   [-Credential <PSCredential>]
   [-Name <String[]>]
   [-EnableNetworkAccess]
   [-ConfigurationName <String>]
   [-Port <Int32>]
   [-UseSSL]
   [-ApplicationName <String>]
   [-ThrottleLimit <Int32>]
   [-SessionOption <PSSessionOption>]
   [-Authentication <AuthenticationMechanism>]
   [-CertificateThumbprint <String>]
   [<CommonParameters>]
```

PowerShell Copy

```
New-PSSession
   [-Credential <PSCredential>]
   [-Name <String[]>]
   [-EnableNetworkAccess]
   [-ConfigurationName <String>]
   [-ThrottleLimit <Int32>]
   [-ConnectionUri] <Uri[]>
   [-AllowRedirection]
   [-SessionOption <PSSessionOption>]
   [-Authentication <AuthenticationMechanism>]
```

Download PDF

```
   [-CertificateThumbprint <String>]
   [<CommonParameters>]
```

PowerShell Copy

```
New-PSSession
   -Credential <PSCredential>
   [-Name <String[]>]
   [-ConfigurationName <String>]
   [-VMId] <Guid[]>
   [-ThrottleLimit <Int32>]
   [<CommonParameters>]
```

PowerShell Copy

```
New-PSSession
   -Credential <PSCredential>
   [-Name <String[]>]
   [-ConfigurationName <String>]
   -VMName <String[]>
   [-ThrottleLimit <Int32>]
   [<CommonParameters>]
```

PowerShell Copy

```
New-PSSession
   [[-Session] <PSSession[]>]
   [-Name <String[]>]
   [-EnableNetworkAccess]
   [-ThrottleLimit <Int32>]
   [<CommonParameters>]
```

PowerShell Copy

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

PowerShell Copy

```
New-PSSession
   [-Name <String[]>]
   [-UseWindowsPowerShell]
   [<CommonParameters>]
```

PowerShell Copy

```
New-PSSession
   [-Name <String[]>]
   [-Port <Int32>]
   [-HostName] <String[]>
   [-UserName <String>]
   [-KeyFilePath <String>]
   [-Subsystem <String>]
   [-ConnectingTimeout <Int32>]
   [-SSHTransport]
   [-Options <Hashtable>]
   [<CommonParameters>]
```

PowerShell Copy

```
New-PSSession
   [-Name <String[]>]
```

```
    -SSHConnection <Hashtable[]>
    [<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.

You can run commands on a remote computer without creating a **PSSession** with 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.

Starting with PowerShell 6.0 you can use Secure Shell (SSH) to establish a connection to and create a session on a remote computer, if SSH is available on the local computer and the remote computer is configured with a PowerShell SSH endpoint. The benefit of an SSH based PowerShell remote session is that it can work across multiple platforms (Windows, Linux, macOS). For SSH based sessions you use the **HostName** or **SSHConnection** parameter set to specify the remote computer and relevant connection information. For more information about how to set up PowerShell SSH remoting, see PowerShell Remoting Over SSH.

> **Note**
>
> When using WSMan remoting from a Linux or macOS client with a HTTPS endpoint where the server certificate is not trusted (e.g., a self-signed certificate). You must provide a **PSSessionOption** that includes the **SkipCACheck** and **SkipCNCheck** values set to `$true` to successfully establish the connection. Only do this if you are in an environment where you can be certain of the server certificate and the network connection to the target system.

Examples

Example 1: Create a session on the local computer

PowerShell Copy

```
$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

PowerShell Copy

```
$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

PowerShell Copy

```
$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

PowerShell Copy

```
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

PowerShell Copy

```
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

PowerShell Copy

```
$global:s = New-PSSession -ComputerName Server1.Domain44.Corpnet.Fabrikam.com -C
```

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

PowerShell Copy

```
$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

PowerShell Copy

```
$s = New-PSSession -URI http://Server01:91/NewSession -Credential Domain01\User0
```

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

PowerShell Copy

```
$s = New-PSSession -ComputerName (Get-Content Servers.txt) -Credential Domain01\
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 and about_Remote_Jobs.

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

PowerShell Copy

```
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

PowerShell Copy

```
$so = New-PSSessionOption -SkipCACheck
New-PSSession -ConnectionUri https://management.exchangelabs.com/Management -Ses
```

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.

Example 12: Create a session using SSH

PowerShell Copy

```
New-PSSession -HostName UserA@LinuxServer01
```

This example shows how to create a new **PSSession** using Secure Shell (SSH). If SSH is configured on the remote computer to prompt for passwords then you will get a password prompt. Otherwise you will have to use SSH key based user authentication.

Example 13: Create a session using SSH and specify the port and user authentication key

PowerShell Copy

```
New-PSSession -HostName UserA@LinuxServer01:22 -KeyFilePath c:\<path>\userAKey_r
```

This example shows how to create a **PSSession** using Secure Shell (SSH). It uses the **Port** parameter to specify the port to use and the **KeyFilePath** parameter to specify an RSA key used to identify and authenticate the user on the remote computer.

Example 14: Create multiple sessions using SSH

PowerShell Copy

```
$sshConnections = @(
    @{
          HostName     = 'WinServer1'
          UserName     = 'domain\userA'
          KeyFilePath = 'c:\users\UserA\id_rsa'
    }
    @{
        HostName     = 'UserB@LinuxServer5'
        KeyFilePath = 'c:\UserB\<path>\id_rsa'
    }
)
New-PSSession -SSHConnection $sshConnections
```

This example shows how to create multiple sessions using Secure Shell (SSH) and the **SSHConnection** parameter set. The **SSHConnection** parameter takes an array of hash tables that contain connection information for each session. Note that this example requires that the target remote computers have SSH configured to support key-based user authentication.

Example 15: Create a new session using SSH options

PowerShell Copy

```
$options = @{
    Port=22
    User = 'UserB'
    Host = 'LinuxServer5'
}
New-PSSession -KeyFilePath '/Users/UserB/id_rsa' -Options $options
```

This example shows how to create a new SSH-based session a remote Linux-based machine using SSH options. The **Options** parameter takes a hashtable of values that are passed as options to the underlying `ssh` command the established the connection to the remote system.

Parameters

-AllowRedirection

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`.

Expand table

Type:	SwitchParameter
Position:	Named
Default value:	False

Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

-ApplicationName

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.

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.

Expand table

Type:	String
Position:	Named
Default value:	None
Required:	False
Accept pipeline input:	True
Accept wildcard characters:	False

-Authentication

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.

> **⊗ 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.

Expand table

Type:	AuthenticationMechanism
Accepted values:	Default, Basic, Negotiate, NegotiateWithImplicitCredential, Credssp, Digest, Kerberos
Position:	Named
Default value:	None
Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

-CertificateThumbprint

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.

Expand table

Type:	String
Position:	Named
Default value:	None
Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

-ComputerName

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.

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

Expand table

Type:	String[]
Aliases:	Cn
Position:	0
Default value:	None
Required:	False
Accept pipeline input:	True
Accept wildcard characters:	False

-ConfigurationName

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.

Expand table

Type:	String
Position:	Named
Default value:	None
Required:	False
Accept pipeline input:	True
Accept wildcard characters:	False

-ConnectingTimeout

Specifies the amount of time in milliseconds allowed for the initial SSH connection to complete. If the connection doesn't complete within the specified time, an error is returned.

This parameter was introduced in PowerShell 7.2

Expand table

Type:	Int32
Position:	Named
Default value:	unlimited
Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

-ConnectionUri

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.

Expand table

Type:	Uri[]
Aliases:	URI, CU
Position:	0
Default value:	None
Required:	True
Accept pipeline input:	True
Accept wildcard characters:	False

-ContainerId

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 command.

Expand table

Type:	String[]
Position:	Named
Default value:	None
Required:	True
Accept pipeline input:	True
Accept wildcard characters:	False

-Credential

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 object and the password is stored as a SecureString.

> **Note**

For more information about **SecureString** data protection, see How secure is SecureString?.

Expand table

Type:	PSCredential
Position:	Named
Default value:	Current user
Required:	False
Accept pipeline input:	True
Accept wildcard characters:	False

-EnableNetworkAccess

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.

Expand table

Type:	SwitchParameter
Position:	Named
Default value:	False
Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

-HostName

Specifies an array of computer names for a Secure Shell (SSH) based connection. This is similar to the **ComputerName** parameter except that the connection to the remote computer is made using SSH rather than Windows WinRM.

This parameter was introduced in PowerShell 6.0.

Expand table

Type:	String[]
Position:	0
Default value:	None
Required:	True
Accept pipeline input:	False
Accept wildcard characters:	False

-KeyFilePath

Specifies a key file path used by Secure Shell (SSH) to authenticate a user on a remote computer.

SSH allows user authentication to be performed via private/public keys as an alternative to basic password authentication. If the remote computer is configured for key authentication then this parameter can be used to provide the key that identifies the user.

This parameter was introduced in PowerShell 6.0.

Expand table

Type:	String
Aliases:	IdentityFilePath
Position:	Named
Default value:	None
Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

-Name

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.

Expand table

Type:	String[]
Position:	Named
Default value:	None
Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

-Options

Specifies a hashtable of SSH options used when connecting to a remote SSH-based session. The possible options are any values supported by the Unix-based version of the ssh command.

Any values explicitly passed by parameters take precedence over values passed in the **Options** hashtable. For example, using the **Port** parameter overrides any `Port` key-value pair passed in the **Options** hashtable.

Expand table

Type:	Hashtable
Position:	Named
Default value:	None
Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

-Port

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.

Expand table

Type:	Int32
Position:	Named
Default value:	None
Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

-RunAsAdministrator

Indicates that the **PSSession** runs as administrator.

Expand table

Type:	SwitchParameter
Position:	Named
Default value:	False

Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

-Session

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).

Expand table

Type:	PSSession[]
Position:	0
Default value:	None
Required:	False
Accept pipeline input:	True
Accept wildcard characters:	False

-SessionOption

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. For more information about session configurations, see about_Session_Configurations.

Expand table

Type:	PSSessionOption
Position:	Named
Default value:	None
Required:	False
Accept pipeline input:	False

Accept wildcard characters:	False

-SSHConnection

This parameter takes an array of hashtables where each hashtable contains one or more connection parameters needed to establish a Secure Shell (SSH) connection (**HostName**, **Port**, **UserName**, **KeyFilePath**).

The hashtable connection parameters are the same as defined for the **SSHHost** parameter set.

The **SSHConnection** parameter is useful for creating multiple sessions where each session requires different connection information.

This parameter was introduced in PowerShell 6.0.

Expand table

Type:	Hashtable[]
Position:	Named
Default value:	None
Required:	True
Accept pipeline input:	False
Accept wildcard characters:	False

-SSHTransport

Indicates that the remote connection is established using Secure Shell (SSH).

By default PowerShell uses Windows WinRM to connect to a remote computer. This switch forces PowerShell to use the HostName parameter set for establishing an SSH based remote connection.

This parameter was introduced in PowerShell 6.0.

Expand table

Type:	SwitchParameter
Accepted values:	true
Position:	Named
Default value:	False
Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

-Subsystem

Specifies the SSH subsystem used for the new **PSSession**.

This specifies the subsystem to use on the target as defined in `sshd_config`. The subsystem starts a specific version of PowerShell with predefined parameters. If the specified subsystem does not exist on the remote computer, the command fails.

If this parameter is not used, the default is the `powershell` subsystem.

Expand table

Type:	String
Position:	Named
Default value:	powershell
Required:	False
Accept pipeline input:	True
Accept wildcard characters:	False

-ThrottleLimit

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.

Expand table

Type:	Int32
Position:	Named
Default value:	None
Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

-UserName

Specifies the username for the account used to create a session on the remote computer. The user authentication method depends on how Secure Shell (SSH) is configured on the remote computer.

If SSH is configured for basic password authentication then you'll be prompted for the user password.

If SSH is configured for key-based user authentication then a key file path can be provided via the **KeyFilePath** parameter and you won't be prompted for a password. Note that if the client user key file is located in an SSH known location then the **KeyFilePath** parameter is not needed for key-based authentication, and user authentication occurs automatically based on the username. See SSH documentation about key-based user authentication for more information.

This is not a required parameter. If no **UserName** parameter is specified then the current log on username is used for the connection.

This parameter was introduced in PowerShell 6.0.

Expand table

Type:	String
Position:	Named
Default value:	None
Required:	False

Accept pipeline input:	False
Accept wildcard characters:	False

-UseSSL

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.

Expand table

Type:	SwitchParameter
Position:	Named
Default value:	False
Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

-UseWindowsPowerShell

Creates a remote connection to a new Windows PowerShell runspace on the local system.

Expand table

Type:	SwitchParameter
Position:	Named
Default value:	False
Required:	True
Accept pipeline input:	False
Accept wildcard characters:	False

-VMId

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.

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

Expand table

Type:	Guid[]
Aliases:	VMGuid
Position:	0
Default value:	None
Required:	True

Accept pipeline input:	True
Accept wildcard characters:	False

-VMName

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.

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

Expand table

Type:	String[]
Position:	Named
Default value:	None
Required:	True
Accept pipeline input:	True
Accept wildcard characters:	False

Inputs

String

You can pipe a string to this cmdlet.

System.URI

You can pipe a URI to this cmdlet.

PSSession

You can pipe a session object to this cmdlet.

Outputs

PSSession

Notes

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

- All platforms:
  - `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.
- 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.
- The **HostName** and **SSHConnection** parameter sets were included starting with PowerShell 6.0. They were added to provide PowerShell remoting based on Secure Shell

(SSH). Both SSH and PowerShell are supported on multiple platforms (Windows, Linux, macOS) and PowerShell remoting will work over these platforms where PowerShell and SSH are installed and configured. This is separate from the previous Windows-only remoting that is based on WinRM and many of the WinRM specific features and limitations do not apply. For example, WinRM-based quotas, session options, custom endpoint configuration, and disconnect/reconnect features are not supported. For more information about how to set up PowerShell SSH remoting, see PowerShell Remoting Over SSH.

The `ssh` executable obtains configuration data from the following sources in the following order:

1. command-line options
2. user's configuration file (~/.ssh/config)
3. system-wide configuration file (/etc/ssh/ssh_config)

The following cmdlet parameters get mapped into `ssh` parameters and options:

Expand table

Cmdlet parameter	ssh parameter	equivalent ssh -o option
`-KeyFilePath`	`-i <KeyFilePath>`	`-o IdentityFile=<KeyFilePath>`
`-UserName`	`-l <UserName>`	`-o User=<UserName>`
`-Port`	`-p <Port>`	`-o Port=<Port>`
`-ComputerName -Subsystem`	`-s <ComputerName> <Subsystem>`	`-o Host=<ComputerName>`

Any values explicitly passed by parameters take precedence over values passed in the **Options** hashtable. For more information about `ssh_config` files, see ssh_config(5) .

Related Links

- Connect-PSSession
- Disconnect-PSSession
- Enter-PSSession
- Exit-PSSession
- Get-PSSession
- Invoke-Command
- Receive-PSSession
- Remove-PSSession

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see our contributor guide.

PowerShell feedback

PowerShell is an open source project. Select a link to provide feedback:

Open a documentation issue

Provide product feedback