Full credit is given to the above companies including the OS that this PDF file was generated!

PowerShell Get-Help on command 'Get-PSSession'

PS C:\Users\wahid> Get-Help Get-PSSession

NAME

Get-PSSession

SYNOPSIS

Gets the PowerShell sessions on local and remote computers.

SYNTAX

Get-PSSession [-ConnectionUri] <System.Uri[]> [-AllowRedirection]
[-Authentication {Default | Basic | Negotiate |
NegotiateWithImplicitCredential | Credssp | Digest | Kerberos}]
[-CertificateThumbprint <System.String>] [-ConfigurationName <System.String>]
[-Credential <System.Management.Automation.PSCredential>] [-InstanceId
<System.Guid[]>] [-SessionOption
<System.Management.Automation.Remoting.PSSessionOption>] [-State {All | Opened
| Disconnected | Closed | Broken}] [-ThrottleLimit <System.Int32>]
[<CommonParameters>]

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

[-CertificateThumbprint <System.String>] [-ConfigurationName <System.String>]
[-Credential <System.Management.Automation.PSCredential>] [-Name
<System.String[]>] [-SessionOption
<System.Management.Automation.Remoting.PSSessionOption>] [-State {All | Opened
| Disconnected | Closed | Broken}] [-ThrottleLimit <System.Int32>]
[<CommonParameters>]

Get-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>] [-InstanceId
<System.Guid[]>] [-Port <System.Int32>] [-SessionOption
<System.Management.Automation.Remoting.PSSessionOption>] [-State {All | Opened
| Disconnected | Closed | Broken}] [-ThrottleLimit <System.Int32>] [-UseSSL]
[<CommonParameters>]

Get-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>] [-Name
<System.String[]>] [-Port <System.Int32>] [-SessionOption
<System.Management.Automation.Remoting.PSSessionOption>] [-State {All | Opened
| Disconnected | Closed | Broken}] [-ThrottleLimit <System.Int32>] [-UseSSL]
[<CommonParameters>]

Get-PSSession [-ConfigurationName <System.String>] -ContainerId
<System.String[]> [-Name <System.String[]>] [-State {All | Opened |
Disconnected | Closed | Broken}] [<CommonParameters>]

Get-PSSession [-ConfigurationName <System.String>] -ContainerId
<System.String[]> [-InstanceId <System.Guid[]>] [-State {All | Opened |

Disconnected | Closed | Broken}] [<CommonParameters>]

Get-PSSession [-ConfigurationName <System.String>] [-Name <System.String[]>]
[-State {All | Opened | Disconnected | Closed | Broken}] -VMId <System.Guid[]>
[<CommonParameters>]

Get-PSSession [-ConfigurationName <System.String>] [-InstanceId
<System.Guid[]>] [-State {All | Opened | Disconnected | Closed | Broken}]
-VMId <System.Guid[]> [<CommonParameters>]

Get-PSSession [-ConfigurationName <System.String>] [-Name <System.String[]>]
[-State {All | Opened | Disconnected | Closed | Broken}] -VMName
<System.String[]> [<CommonParameters>]

Get-PSSession [-ConfigurationName <System.String>] [-InstanceId
<System.Guid[]>] [-State {All | Opened | Disconnected | Closed | Broken}]
-VMName <System.String[]> [<CommonParameters>]

Get-PSSession [-Id] <System.Int32[]> [<CommonParameters>]

Get-PSSession [-InstanceId <System.Guid[]>] [<CommonParameters>]

Get-PSSession [-Name <System.String[]>] [<CommonParameters>]

DESCRIPTION

The `Get-PSSession` cmdlet gets the user-managed PowerShell sessions (
PSSessions) on local and remote computers.

Starting in Windows PowerShell 3.0, sessions are stored on the computers at
the remote end of each connection. You can use the ComputerName or
ConnectionUri parameters of `Get-PSSession` to get the sessions that connect
to the local computer or remote computers, even if they were not created in

the current session.

Without parameters, `Get-PSSession` gets all sessions that were created in the current session.

Use the filtering parameters, including Name , ID , InstanceID , State , ApplicationName , and ConfigurationName to select from among the sessions that `Get-PSSession` returns.

Use the remaining parameters to configure the temporary connection in which the `Get-PSSession` command runs when you use the ComputerName or ConnectionUri parameters.

> [!NOTE] > In Windows PowerShell 2.0, without parameters, `Get-PSSession` gets all sessions that were created > in the current session. The ComputerName parameter gets sessions that were created in the > current session and connect to the specified computer.

For more information about PowerShell sessions, see about_PSSessions (about/about_PSSessions.md).

PARAMETERS

-AllowRedirection <System.Management.Automation.SwitchParameter>

Indicates that this cmdlet allows redirection of this connection to an alternate Uniform Resource Identifier (URI). By default, PowerShell does not redirect connections.

This parameter configures the temporary connection that is created to run a `Get-PSSession` command with the ConnectionUri parameter.

This parameter was introduced in Windows PowerShell 3.0.

-ApplicationName <System.String>

Specifies the name of an application. This cmdlet connects only to sessions that use the specified application.

Enter the application name segment of the connection URI. For example, in the following connection URI, the application name is WSMan: `http://localhost:5985/WSMAN`. The application name of a session is stored in the Runspace.ConnectionInfo.AppName property of the session.

The value of this parameter is used to select and filter sessions. It does not change the application that the session uses.

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

Specifies the mechanism that is used to authenticate credentials for the session in which the `Get-PSSession` command runs.

This parameter configures the temporary connection that is created to run a `Get-PSSession` command with the ComputerName or ConnectionUri parameter.

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's 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.
This parameter was introduced in Windows PowerShell 3.0.

-CertificateThumbprint <System.String>

Specifies the digital public key certificate (X509) of a user account that has permission to create the session in which the `Get-PSSession` command runs. Enter the certificate thumbprint of the certificate.

This parameter configures the temporary connection that is created to run a `Get-PSSession` command with the ComputerName or ConnectionUri parameter.

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 thumbprint, use a `Get-Item` or `Get-ChildItem`

command in the PowerShell `Cert:` drive.

This parameter was introduced in Windows PowerShell 3.0.

-ComputerName <System.String[]>

Specifies an array of names of computers. Gets the sessions that connect to the specified computers. Wildcard characters are not permitted. There is no default value.

Beginning in Windows PowerShell 3.0, PSSession objects are stored on the computers at the remote end of each connection. To get the sessions on the specified computers, PowerShell creates a temporary connection to each computer and runs a `Get-PSSession` command.

Type the NetBIOS name, an IP address, or a fully-qualified domain name of one or more computers. To specify the local computer, type the computer name, `localhost`, or a dot (`.`).

> [!NOTE] > This parameter gets sessions only from computers that run Windows PowerShell 3.0 or later versions > of PowerShell. Earlier versions do not store sessions.

-ConfigurationName <System.String>

Specifies the name of a configuration. This cmdlet gets only to sessions that use the specified session configuration.

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 configuration name of a session is stored in the ConfigurationName property of the session.

The value of this parameter is used to select and filter sessions. It does

not change the session configuration that the session uses.

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

-ConnectionUri <System.Uri[]>

Specifies a URI that defines the connection endpoint for the temporary session in which the `Get-PSSession` command runs. The URI must be fully qualified.

This parameter configures the temporary connection that is created to run a `Get-PSSession` command with the ConnectionUri parameter.

The format of this string is:

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

The default value is: `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.

This parameter was introduced in Windows PowerShell 3.0.

This parameter gets sessions only from computers that run Windows PowerShell 3.0 or later versions of Windows PowerShell. Earlier versions do not store sessions.

-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.

-Credential <System.Management.Automation.PSCredential>

Specifies a user credential. This cmdlet runs the command with the permissions of the specified user. Specify a user account that has permission to connect to the remote computer and run a `Get-PSSession` command. 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).

This parameter configures to the temporary connection that is created to run a `Get-PSSession` command with the ComputerName or ConnectionUri parameter.

This parameter was introduced in Windows PowerShell 3.0.

-Id <System.Int32[]>

Specifies an array of session IDs. This cmdlet gets only the sessions with the specified IDs. Type one or more IDs, separated by commas, or use the range operator (`..`) to specify a range of IDs. You cannot use the ID parameter together with the ComputerName parameter.

An ID is an integer that uniquely identifies the user-managed sessions in the current session. It is easier to remember and type than the InstanceId , but it is unique only within the current session. The ID of a session is stored in the ID property of the session.

-InstanceId <System.Guid[]>

Specifies an array of instance IDs of sessions. This cmdlet gets only the sessions with the specified instance IDs.

The instance ID is a GUID that uniquely identifies a session on a local or remote computer. The InstanceID is unique, even when you have multiple sessions running in PowerShell.

The instance ID of a session is stored in the InstanceID property of the session.

-Name <System.String[]>

Specifies an array of session names. This cmdlet gets only the sessions that have the specified friendly names. Wildcard characters are permitted.

The friendly name of a session is stored in the Name property of the session.

-Port <System.Int32>

Specifies the specified network port that is used for the temporary connection in which the `Get-PSSession` command runs. 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 an alternate port, you must configure the WinRM listener on the remote computer to listen at that port. To configure the listener, type the following two commands at the PowerShell prompt:

`Remove-Item -Path WSMan:\Localhost\listener\listener* -Recurse`

`New-Item -Path WSMan:\Localhost\listener -Transport http -Address * -Port <port-number>`

This parameter configures to the temporary connection that is created to run a `Get-PSSession` command with the ComputerName or ConnectionUri parameter.

Do not use the Port parameter unless you must. The Port set 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.

This parameter was introduced in Windows PowerShell 3.0.

-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, including 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).

-State <Microsoft.PowerShell.Commands.SessionFilterState>

Specifies a session state. This cmdlet gets only sessions in the specified state. The acceptable values for this parameter are: `All`, `Opened`, `Disconnected`, `Closed`, and `Broken`. The default value is `All`.

The session state value is relative to the current sessions. Sessions that were not created in the current sessions and are not connected to the current session have a state of `Disconnected` even when they are connected to a different session.

The state of a session is stored in the State property of the session.

This parameter was introduced in Windows PowerShell 3.0.

-ThrottleLimit <System.Int32>

Specifies the maximum number of concurrent connections that can be established to run the `Get-PSSession` 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.

This parameter was introduced in Windows PowerShell 3.0.

-UseSSL <System.Management.Automation.SwitchParameter>
Indicates that this cmdlet uses the Secure Sockets Layer (SSL) protocol to establish the connection in which the `Get-PSSession` command runs. By default, SSL is not used. If you use this parameter, but SSL is not available on the port used for the command, the command fails.

This parameter configures the temporary connection that is created to run a `Get-PSSession` command with the ComputerName parameter.

This parameter was introduced in Windows PowerShell 3.0.

-VMId <System.Guid[]>
Specifies an array of ID of virtual machines. This cmdlet starts an interactive session with each of the specified virtual machines. To see the virtual machines that are available to you, use the following command:

`Get-VM | Select-Object -Property Name, ID`

-VMName <System.String[]>
Specifies an array of names of virtual machines. This cmdlet starts an interactive session with each of the specified virtual machines. To see the virtual machines that are available to you, use the `Get-VM` cmdlet.

<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).

---- Example 1: Get sessions created in the current session ----

Get-PSSession

This command gets all of the PSSessions that were created in the current session. It does not get PSSessions that were created in other sessions or on other computers, even if they connect to this computer.

--- Example 2: Get sessions connected to the local computer ---

Get-PSSession -ComputerName "localhost"

This command gets the PSSessions that are connected to the local computer. To indicate the local computer, type the computer name, localhost, or a dot (`.`)

The command returns all of the sessions on the local computer, even if they were created in different sessions or on different computers.

------- Example 3: Get sessions connected to a computer -------

Get-PSSession -ComputerName "Server02"

```
Id Name            ComputerName    State         ConfigurationName
Availability
 -- ----            ------------    -----         -----------------
------------
  2 Session3        Server02        Disconnected  ITTasks
Busy
  1 ScheduledJobs   Server02        Opened        Microsoft.PowerShell
Available
  3 Test            Server02        Disconnected  Microsoft.PowerShell
Busy
```

This command gets the PSSessions that are connected to the Server02 computer.

The command returns all of the sessions on Server02, even if they were created in different sessions or on different computers.

The output shows that two of the sessions have a `Disconnected` state and a `Busy` availability. They were created in different sessions and are currently in use. The `ScheduledJobs` session, which is `Opened` and `Available`, was created in the current session.
---------- Example 4: Save results of this command ----------

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

This example shows how to save the results of a `Get-PSSession` command in multiple variables.

The first command uses the `New-PSSession` cmdlet to create PSSessions on three remote computers.

The second command uses a `Get-PSSession` cmdlet to get the three PSSessions . It then saves each of the PSSessions in a separate variable.

When PowerShell assigns an array of objects to an array of variables, it assigns the first object to the first variable, the second object to the second variable, and so on. If there are more objects than variables, it assigns all remaining objects to the last variable in the array. If there are more variables than objects, the extra variables are not used.
----- Example 5: Delete a session by using an instance ID -----

Get-PSSession | Format-Table -Property ComputerName, InstanceID
$s = Get-PSSession -InstanceID a786be29-a6bb-40da-80fb-782c67f7db0f
Remove-PSSession -Session $s

This example shows how to get a PSSession by using its instance ID, and then to delete the PSSession .

The first command gets all of the PSSessions that were created in the current session. It sends the PSSessions to the `Format-Table` cmdlet, which displays the ComputerName and InstanceID properties of each PSSession .

The second command uses the `Get-PSSession` cmdlet to get a particular PSSession and to save it in the `$s` variable. The command uses the InstanceID parameter to identify the PSSession .

The third command uses the Remove-PSSession cmdlet to delete the PSSession in the `$s` variable.

----- Example 6: Get a session that has a particular name -----

```
Get-PSSession -ComputerName Server02, Server12 -Name BackupJob*
-ConfigurationName ITTasks -SessionOption @{OperationTimeout=240000}

Id Name            ComputerName    State         ConfigurationName
Availability
 -- ----            ------------    -----         -----------------
------------
  3 BackupJob04     Server02        Disconnected  ITTasks
  None

$s = Get-PSSession -ComputerName Server02 -Name BackupJob04 -ConfigurationName
ITTasks | Connect-PSSession
$s

Id Name            ComputerName    State         ConfigurationName
Availability
-- ----            ------------    -----         -----------------
------------
```

5 BackupJob04 Server02 Opened ITTasks
Available

The first command gets sessions on the Server02 and Server12 remote computers that have names that begin with `BackupJob` and use the `ITTasks` session configuration.The command uses the Name parameter to specify the name pattern and the ConfigurationName parameter to specify the session configuration. The value of the SessionOption parameter is a hash table that sets the value of the OperationTimeout to `240000` milliseconds (4 minutes). This setting gives the command more time to complete.The ConfigurationName and SessionOption parameters are used to configure the temporary sessions in which the `Get-PSSession` cmdlet runs on each computer.The output shows that the command returns the `BackupJob04` session. The session is disconnected and the Availability is `None`, which indicates that it is not in use.

The second command uses the `Get-PSSession` cmdlet to get to the `BackupJob04` session and the `Connect-PSSession` cmdlet to connect to the session. The command saves the session in the `$s` variable.

The third command gets the session in the `$s` variable. The output shows that the `Connect-PSSession` command was successful. The session is in the Opened state and is available for use.

---------- Example 7: Get a session by using its ID ----------

Get-PSSession -Id 2

This command gets the PSSession with ID `2`. Because the value of the ID property is unique only in the current session, the Id parameter is valid only for local commands.

REMARKS

To see the examples, type: "get-help Get-PSSession -examples".
For more information, type: "get-help Get-PSSession -detailed".
For technical information, type: "get-help Get-PSSession -full".

For online help, type: "get-help Get-PSSession -online"