MyWebUniversity
python
PowerShell Get-Help cmdlet
PS C:\> Get-Help
FPDF Library
PDF generator
FPDF
Full credit is given to all the above companies including the Operating System that this PDF file was generated!

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

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

NAME

    Disconnect-PSSession

SYNOPSIS

    Disconnects from a session.

SYNTAX

```
        Disconnect-PSSession [-Id] <System.Int32[]> [-IdleTimeoutSec <System.Int32>] [-OutputBufferingMode
<System.Management.Automation.Runspaces.OutputBufferingMode>]
    [-ThrottleLimit <System.Int32>] [-Confirm] [-WhatIf] [<CommonParameters>]

        Disconnect-PSSession [-IdleTimeoutSec <System.Int32>] -InstanceId <System.Guid[]> [-OutputBufferingMode
<System.Management.Automation.Runspaces.OutputBufferingMode>]
    [-ThrottleLimit <System.Int32>] [-Confirm] [-WhatIf] [<CommonParameters>]

        Disconnect-PSSession [-IdleTimeoutSec <System.Int32>] -Name <System.String[]> [-OutputBufferingMode
<System.Management.Automation.Runspaces.OutputBufferingMode>]
    [-ThrottleLimit <System.Int32>] [-Confirm] [-WhatIf] [<CommonParameters>]
```

Disconnect-PSSession [-Session] <System.Management.Automation.Runspaces.PSSession[]> [-IdleTimeoutSec <System.Int32>] [-OutputBufferingMode
<System.Management.Automation.Runspaces.OutputBufferingMode>] [-ThrottleLimit <System.Int32>] [-Confirm] [-WhatIf] [<CommonParameters>]

DESCRIPTION

The `Disconnect-PSSession` cmdlet disconnects a PowerShell session (PSSession), such as one started by using the `New-PSSession` cmdlet, from the current session.

As a result, the PSSession is in a disconnected state. You can connect to the disconnected PSSession from the current session or from another session on the local
computer or a different computer.

The `Disconnect-PSSession` cmdlet disconnects only open PSSessions that are connected to the current session. `Disconnect-PSSession` cannot disconnect broken or
closed PSSessions , or interactive PSSessions started by using the `Enter-PSSession` cmdlet, and it cannot disconnect PSSessions that are connected to other sessions.

To reconnect to a disconnected PSSession , use the `Connect-PSSession` or `Receive-PSSession` cmdlets.

When a PSSession is disconnected, the commands in the PSSession continue to run until they complete, unless the PSSession times out or the commands in the PSSession
are blocked by a full output buffer. To change the idle timeout, use the IdleTimeoutSec parameter. To change the output buffering mode, use the OutputBufferingMode
parameter You can also use the InDisconnectedSession parameter of the `Invoke-Command` cmdlet to run a command in a disconnected session.

For more information about the Disconnected Sessions feature, see about_Remote_Disconnected_Sessions (./About/about_Remote_Disconnected_Sessions.md).

This cmdlet is introduced in Windows PowerShell 3.0.

PARAMETERS

-Id <System.Int32[]>

Disconnects from sessions with the specified session ID. Type one or more IDs (separated by commas), or use the range operator (`..`) to specify a range of IDs.

To get the ID of a session, use the `Get-PSSession` cmdlet. The instance ID is stored in the ID property of the session.

Required?	true
Position?	1
Default value	None
Accept pipeline input?	True (ByPropertyName)
Accept wildcard characters?	false

-IdleTimeoutSec <System.Int32>

Changes the idle timeout value of the disconnected PSSession . Enter a value in seconds. The minimum value is `60` (1 minute).

The idle timeout determines how long the disconnected PSSession is maintained on the remote computer. When the timeout expires, the PSSession is deleted.

Disconnected PSSessions are considered to be idle from the moment that they are disconnected, even if commands are running in the disconnected session.

The default value for the idle timeout of a session is set by the value of the IdleTimeoutMs property of the session configuration. The default value is `7200000`

milliseconds (2 hours).

The value of this parameter takes precedence over the value of the IdleTimeout property of the `$PSSessionOption` preference variable and the default idle timeout

value in the session configuration. However, this value cannot exceed the value of the MaxIdleTimeoutMs property of the session configuration. The default value

of MaxIdleTimeoutMs is 12 hours (`43200000` milliseconds).

Required?   false

Position?   named

Default value   60

Accept pipeline input?   False

Accept wildcard characters? false

-InstanceId <System.Guid[]>

Disconnects from sessions with the specified instance IDs.

The instance ID is a GUID that uniquely identifies a session on a local or remote computer. The instance ID is unique, even across multiple sessions on multiple

computers.

To get the instance ID of a session, use the `Get-PSSession` cmdlet. The instance ID is stored in the InstanceID property of the session.

Required?   true

Position?   named

Default value   None

Accept pipeline input?   True (ByPropertyName)

Accept wildcard characters? false

-Name <System.String[]>

Disconnects from sessions with the specified friendly names. Wildcards are permitted.

To get the friendly name of a session, use the `Get-PSSession` cmdlet. The friendly name is stored in the Name property of the session.

Required?   true

Position?   named

Default value   None

Accept pipeline input?   True (ByPropertyName)

Accept wildcard characters? true

-OutputBufferingMode <System.Management.Automation.Runspaces.OutputBufferingMode>

Determines how command output is managed in the disconnected session when the output buffer is full. The default value is `Block`.

If the command in the disconnected session is returning output and the output buffer fills, the value of this parameter effectively determines whether the command

continues to run while the session is disconnected. A value of `Block` suspends the command until the session is reconnected. A value of `Drop` allows the command

to complete, although data might be lost. When using the `Drop` value, redirect the command output to a file on disk.

Valid values are:

- `Block`: When the output buffer is full, execution is suspended until the buffer is clear.

- `Drop`: When the output buffer is full, execution continues. As new output is saved, the oldest

output is discarded. - `None`: No output buffering mode is specified. The value of the OutputBufferingMode property of the session configuration is used for the

disconnected session.

```
Required?                    false
Position?                    named
Default value                Block
Accept pipeline input?       False
Accept wildcard characters?  false
```

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

Disconnects from the specified PSSessions . Enter PSSession objects, such as those that the `New-PSSession` cmdlet returns. You can also pipe a PSSession object

to `Disconnect-PSSession`.

The `Get-PSSession` cmdlet can get all PSSessions that terminate at a remote computer, including PSSessions that are disconnected and PSSessions that are

connected to other sessions on other computers. `Disconnect-PSSession` disconnects only PSSession that are connected to the current session. If you pipe other

PSSessions to `Disconnect-PSSession`, the `Disconnect-PSSession` command fails.

```
        Required?                    true
        Position?                    1
        Default value                None
        Accept pipeline input?       True (ByPropertyName, ByValue)
        Accept wildcard characters?  false
```

-ThrottleLimit <System.Int32>

Sets the throttle limit for the `Disconnect-PSSession` command.

The throttle limit is 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`, 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                32
        Accept pipeline input?       False
        Accept wildcard characters?  false
```

-Confirm <System.Management.Automation.SwitchParameter>

Prompts you for confirmation before running the cmdlet.

```
        Required?                    false
        Position?                    named
        Default value                False
        Accept pipeline input?       False
        Accept wildcard characters?  false


    -WhatIf <System.Management.Automation.SwitchParameter>
        Shows what would happen if the cmdlet runs. The cmdlet is not run.


        Required?                    false
        Position?                    named
        Default value                False
        Accept pipeline input?       False
        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.Management.Automation.Runspaces.PSSession
        You can pipe a session to this cmdlet.



OUTPUTS
    System.Management.Automation.Runspaces.PSSession
        This cmdlet returns an object representing the session that it disconnected.



NOTES



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


        - `dnsn`
```

- The `Disconnect-PSSession` cmdlet works only when the local and remote computers are running PowerShell 3.0 or later. - If you use the `Disconnect-PSSession` cmdlet on a disconnected session, the command has no effect on the session and it does not generate errors. - Disconnected loopback sessions with interactive security tokens (those created with the EnableNetworkAccess parameter) can be reconnected only from the computer on which the session was created. This restriction protects the computer from malicious access. - When you disconnect a PSSession , the session state is `Disconnected` and the availability is None .

The value of the State property is relative to the current session. Therefore, a value of `Disconnected` means that the PSSession is not connected to the current session. However, it does not mean that the PSSession is disconnected from all sessions. It might be connected to a different session. To determine whether you can connect or reconnect to the session, use the Availability property.

An Availability value of `None` indicates that you can connect to the session. A value of `Busy` indicates that you cannot connect to the PSSession because it is connected to another session.

For more information about the values of the State property of sessions, see RunspaceState Enumeration (/dotnet/api/system.management.automation.runspaces.runspacestate).

For more information about the values of the Availability property of sessions, see RunspaceAvailability Enumeration (/dotnet/api/system.management.automation.runspaces.runspaceavailability).

----------- Example 1 - Disconnect a session by name -----------

```
PS> Disconnect-PSSession -Name UpdateSession
Id Name          ComputerName   State        ConfigurationName     Availability
-- ----          ------------   -----        -----------------     ------------
1  UpdateSession  Server01      Disconnected  Microsoft.PowerShell        None
```

The output shows that the attempt to disconnect was successful. The session state is `Disconnected` and the Availability is `None`, which indicates that the session
is not busy and can be reconnected.

-- Example 2 - Disconnect a session from a specific computer --

```
PS> Get-PSSession -ComputerName Server12 -Name ITTask |
  Disconnect-PSSession -OutputBufferingMode Drop -IdleTimeoutSec 86400
Id Name          ComputerName    State        ConfigurationName     Availability
-- ----          ------------    -----        -----------------     ------------
1  ITTask        Server12        Disconnected  ITTasks              None
```

The `Disconnect-PSSession` command uses the OutputBufferingMode parameter to set the output mode to `Drop`. This setting ensures that the script that is running in
the session can continue to run even if the session output buffer is full. Because the script writes its output to a report on a file share, other output can be lost
without consequence.

The command also uses the IdleTimeoutSec parameter to extend the idle timeout of the session to 24 hours. This setting allows time for this administrator or other
administrators to reconnect to the session to verify that the script ran and troubleshoot if needed.

- Example 3 - Using multiple PSSessions on multiple computers -

```
PS> $s = New-PSSession -ComputerName Srv1, Srv2, Srv30 -Name ITTask
PS> Invoke-Command $s -FilePath \\Server01\Scripts\Get-PatchStatus.ps1
PS> Get-PSSession -Name ITTask -ComputerName Srv1 | Disconnect-PSSession
Id Name          ComputerName    State        ConfigurationName     Availability
-- ----          ------------    -----        -----------------     ------------
1 ITTask         Srv1            Disconnected  Microsoft.PowerShell        None

PS> Get-PSSession -ComputerName Srv1, Srv2, Srv30 -Name ITTask
Id Name          ComputerName    State        ConfigurationName     Availability
-- ----          ------------    -----        -----------------     ------------
 1 ITTask        Srv1            Disconnected  Microsoft.PowerShell        None
```

```
     2 ITTask         Srv2          Opened       Microsoft.PowerShell     Available
     3 ITTask         Srv30          Opened       Microsoft.PowerShell     Available

    PS> Get-PSSession -ComputerName Srv1 -Name ITTask -Credential Domain01\User01
    Id Name           ComputerName    State        ConfigurationName     Availability
    -- ----           ------------    -----        -----------------     ------------
     1 ITTask         Srv1          Disconnected  Microsoft.PowerShell         None

    PS> $s = Connect-PSSession -ComputerName Srv1 -Name ITTask -Credential Domain01\User01
    PS> Invoke-Command -Session $s {dir $HOME\Scripts\PatchStatusOutput.ps1}
    PS> Invoke-Command -Session $s {mkdir $HOME\Scripts\PatchStatusOutput}
    PS> Invoke-Command -Session $s -FilePath \\Server01\Scripts\Get-PatchStatus.ps1
    PS> Disconnect-PSSession -Session $s
```

The technician begins by creating sessions on several remote computers and running a script in each session. The first command uses the `New-PSSession` cmdlet to
create the `ITTask` session on three remote computers. The command saves the sessions in the `$s` variable. The second command uses the FilePath parameter of the
`Invoke-Command` cmdlet to run a script in the sessions in the `$s` variable.

The script running on the Srv1 computer generates unexpected errors. The technician contacts his manager and asks for assistance. The manager directs the technician
to disconnect from the session so he can investigate.The second command uses the `Get-PSSession` cmdlet to get the `ITTask` session on the Srv1 computer and the
`Disconnect-PSSession` cmdlet to disconnect it. This command does not affect the `ITTask` sessions on the other computers.

The third command uses the `Get-PSSession` cmdlet to get the `ITTask` sessions. The output shows that the `ITTask` sessions on the Srv2 and Srv30 computers were not
affected by the command to disconnect.

The manager logs on to his home computer, connects to his corporate network, starts PowerShell, and uses the `Get-PSSession` cmdlet to get the `ITTask` session on the

Srv1 computer. He uses the credentials of the technician to access the session.

Next, the manager uses the `Connect-PSSession` cmdlet to connect to the `ITTask` session on the Srv1 computer. The command saves the session in the `$s` variable.

The manager uses the `Invoke-Command` cmdlet to run some diagnostic commands in the session in the `$s` variable. He recognizes that the script failed because it did

not find a required directory. The manager uses the `MkDir` function to create the directory, and then he restarts the `Get-PatchStatus.ps1` script and disconnects

from the session.The manager reports his findings to the technician, suggests that he reconnect to the session to complete the tasks, and asks him to add a command to

the `Get-PatchStatus.ps1` script that creates the required directory if it does not exist.

----- Example 4 - Change the timeout value for a PSSession -----

```
PS> $Timeout = New-PSSessionOption -IdleTimeout 172800000
PS> $s = New-PSSession -Computer Server01 -Name ITTask -SessionOption $Timeout
PS> Disconnect-PSSession -Session $s
Disconnect-PSSession : The session ITTask cannot be disconnected because the specified
idle timeout value 172800(seconds) is either greater than the server maximum allowed
43200 (seconds) or less that the minimum allowed60(seconds).  Choose an idle time out
value that is within the allowed range and try again.

PS> Invoke-Command -ComputerName Server01 {Get-PSSessionConfiguration Microsoft.PowerShell} |
 Format-List -Property *

Architecture                  : 64
Filename                      : %windir%\system32\pwrshplugin.dll
ResourceUri                   : http://schemas.microsoft.com/powershell/microsoft.powershell
MaxConcurrentCommandsPerShell : 1000
UseSharedProcess              : false
ProcessIdleTimeoutSec         : 0
xmlns                         : http://schemas.microsoft.com/wbem/wsman/1/config/PluginConfiguration
MaxConcurrentUsers            : 5
```

```
lang                          : en-US
SupportsOptions               : true
ExactMatch                    : true
RunAsUser                     :
IdleTimeoutms                 : 7200000
PSVersion                     : 3.0
OutputBufferingMode           : Block
AutoRestart                   : false
SecurityDescriptorSddl        : O:NSG:BAD:P(A;;GA;;;BA)S:P(AU;FA;GA;;;WD)(AU;SA;GXGW;;;WD)
MaxMemoryPerShellMB           : 1024
MaxIdleTimeoutms              : 2147483647
Uri                           : http://schemas.microsoft.com/powershell/microsoft.powershell
SDKVersion                    : 2
Name                          : microsoft.powershell
XmlRenderingType              : text
Capability                    : {Shell}
RunAsPassword                 :
MaxProcessesPerShell          : 15
ParentResourceUri             : http://schemas.microsoft.com/powershell/microsoft.powershell
Enabled                       : true
MaxShells                     : 25
MaxShellsPerUser              : 25
Permission                    : BUILTIN\Administrators AccessAllowed
PSComputerName                : localhost
RunspaceId                    : aea84310-6dbf-4c21-90ac-13980039925a
PSShowComputerName            : True


PS> $s.Runspace.ConnectionInfo
ConnectionUri                 : http://Server01/wsman
ComputerName                  : Server01
Scheme                        : http
Port                          : 80
```

```
AppName                          : /wsman
Credential                       :
ShellUri                         : http://schemas.microsoft.com/powershell/Microsoft.PowerShell
AuthenticationMechanism          : Default
CertificateThumbprint            :
MaximumConnectionRedirectionCount : 5
MaximumReceivedDataSizePerCommand :
MaximumReceivedObjectSize        : 209715200
UseCompression                   : True
NoMachineProfile                 : False
ProxyAccessType                  : None
ProxyAuthentication              : Negotiate
ProxyCredential                  :
SkipCACheck                      : False
SkipCNCheck                      : False
SkipRevocationCheck              : False
NoEncryption                     : False
UseUTF16                         : False
OutputBufferingMode              : Drop
IncludePortInSPN                 : False
Culture                          : en-US
UICulture                        : en-US
OpenTimeout                      : 180000
CancelTimeout                    : 60000
OperationTimeout                 : 180000
IdleTimeout                      : 172800000

PS> Disconnect-PSSession $s -IdleTimeoutSec 43200
Id Name            ComputerName    State         ConfigurationName     Availability
-- ----            ------------    -----         -----------------     ------------
 4 ITTask          Server01        Disconnected  Microsoft.PowerShell          None

PS> $s.Runspace.ConnectionInfo.IdleTimeout
```

```
    43200000
```

The first command uses the `New-PSSessionOption` cmdlet to create a session option object. It uses the IdleTimeout parameter to set an idle timeout of 48 hours

(`172800000` milliseconds). The command saves the session option object in the `$Timeout` variable.

The second command uses the `New-PSSession` cmdlet to create the `ITTask` session on the Server01 computer. The command save the session in the `$s` variable. The

value of the SessionOption parameter is the 48-hour idle timeout in the `$Timeout` variable.

The third command disconnects the `ITTask` session in the `$s` variable. The command fails because the idle timeout value of the session exceeds the MaxIdleTimeoutMs

quota in the session configuration. Because the idle timeout is not used until the session is disconnected, this violation can go undetected while the session is in

use.

The fourth command uses the `Invoke-Command` cmdlet to run a `Get-PSSessionConfiguration` command for the `Microsoft.PowerShell` session configuration on the Server01

computer. The command uses the `Format-List` cmdlet to display all properties of the session configuration in a list.The output shows that the MaxIdleTimeoutMS

property, which establishes the maximum permitted IdleTimeout value for sessions that use the session configuration, is `43200000` milliseconds (12 hours).

The fifth command gets the session option values of the session in the `$s` variable. The values of many session options are properties of the ConnectionInfo property

of the Runspace property of the session.The output shows that the value of the IdleTimeout property of the session is `172800000` milliseconds (48 hours), which

violates the MaxIdleTimeoutMs quota of 12 hours in the session configuration.To resolve this conflict, you can use the ConfigurationName parameter to select a

different session configuration or use the IdleTimeout parameter to reduce the idle timeout of the session.

The sixth command disconnects the session. It uses the IdleTimeoutSec parameter to set the idle timeout to the 12-hour maximum.

The seventh command gets the value of the IdleTimeout property of the disconnected session, which is measured in milliseconds. The output confirms that the command

was successful.

RELATED LINKS

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

Connect-PSSession

Enter-PSSession

Exit-PSSession

Get-PSSession

Get-PSSessionConfiguration

New-PSSession

New-PSSessionOption

New-PSTransportOption

Receive-PSSession

Register-PSSessionConfiguration

Remove-PSSession

about_PSSessions

about_Remote

about_Remote_Disconnected_Sessions