PowerShell Get-Help on command 'Receive-PSSession'

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

```
NAME
    Receive-PSSession


SYNOPSIS
    Gets results of commands in disconnected sessions



SYNTAX
    Receive-PSSession [-ConnectionUri] <System.Uri> [-AllowRedirection]
    [-Authentication {Default | Basic | Negotiate |
    NegotiateWithImplicitCredential | Credssp | Digest | Kerberos}]
    [-CertificateThumbprint <System.String>] [-ConfigurationName <System.String>]
    [-Credential <System.Management.Automation.PSCredential>] [-JobName
    <System.String>] -Name <System.String> [-OutTarget {Default | Host | Job}]
    [-SessionOption <System.Management.Automation.Remoting.PSSessionOption>]
    [-Confirm] [-WhatIf] [<CommonParameters>]


    Receive-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> [-JobName <System.String>] [-OutTarget {Default | Host | Job}]

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

[-Confirm] [-WhatIf] [<CommonParameters>]

Receive-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> [-JobName <System.String>] [-OutTarget {Default | Host | Job}]

[-Port <System.Int32>] [-SessionOption

<System.Management.Automation.Remoting.PSSessionOption>] [-UseSSL] [-Confirm]

[-WhatIf] [<CommonParameters>]

Receive-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>] [-JobName

<System.String>] -Name <System.String> [-OutTarget {Default | Host | Job}]

[-Port <System.Int32>] [-SessionOption

<System.Management.Automation.Remoting.PSSessionOption>] [-UseSSL] [-Confirm]

[-WhatIf] [<CommonParameters>]

Receive-PSSession [-Id] <System.Int32> [-JobName <System.String>] [-OutTarget

{Default | Host | Job}] [-Confirm] [-WhatIf] [<CommonParameters>]

Receive-PSSession -InstanceId <System.Guid> [-JobName <System.String>]

[-OutTarget {Default | Host | Job}] [-Confirm] [-WhatIf] [<CommonParameters>]

Receive-PSSession [-JobName <System.String>] -Name <System.String> [-OutTarget

{Default | Host | Job}] [-Confirm] [-WhatIf] [<CommonParameters>]

Receive-PSSession [-Session]

<System.Management.Automation.Runspaces.PSSession> [-JobName <System.String>]

[-OutTarget {Default | Host | Job}] [-Confirm] [-WhatIf] [<CommonParameters>]

DESCRIPTION

The `Receive-PSSession` cmdlet gets the results of commands running in PowerShell sessions (PSSession) that were disconnected. If the session is currently connected, `Receive-PSSession` gets the results of commands that were running when the session was disconnected. If the session is still disconnected, `Receive-PSSession` connects to the session, resumes any commands that were suspended, and gets the results of commands running in the session.

This cmdlet was introduced in PowerShell 3.0.

You can use a `Receive-PSSession` in addition to or instead of a `Connect-PSSession` command. `Receive-PSSession` can connect to any disconnected or reconnected session that was started in other sessions or on other computers.

`Receive-PSSession` works on PSSessions that were disconnected intentionally using the `Disconnect-PSSession` cmdlet or the `Invoke-Command` InDisconnectedSession parameter. Or disconnected unintentionally by a network interruption.

If you use the `Receive-PSSession` cmdlet to connect to a session in which no commands are running or suspended, `Receive-PSSession` connects to the session, but returns no output or errors.

For more information about the Disconnected Sessions feature, see about_Remote_Disconnected_Sessions

(./About/about_Remote_Disconnected_Sessions.md).

Some examples use splatting to reduce the line length and improve readability.
For more information, see about_Splatting (./About/about_Splatting.md).

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 doesn't 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.

-ApplicationName <System.String>

Specifies 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, WSMan is the application name: `http://localhost:5985/WSMAN`.

The application name of a session is stored in the Runspace.ConnectionInfo.AppName property of the session.

The parameter's value is used to select and filter sessions. It doesn't change the application that the session uses.

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

Specifies the mechanism that's used to authenticate the user credentials in the command to reconnect to a disconnected session. 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.

-CertificateThumbprint <System.String>

Specifies the digital public key certificate (X509) of a user account that has permission to connect to the disconnected session. Enter the certificate thumbprint of the certificate.

Certificates are used in client certificate-based authentication. Certificates can be mapped only to local user accounts, and don't work with domain accounts.

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

-ComputerName <System.String>

Specifies the computer on which the disconnected session is stored. Sessions are stored on the computer that's at the server-side, or receiving end of a connection. The default is the local computer.

Type the NetBIOS name, an IP address, or a fully qualified domain name (FQDN) of one computer. Wildcard characters aren't permitted. To specify the local computer, type the computer name, a dot (`.`), `$env:COMPUTERNAME`, or localhost.

-ConfigurationName <System.String>

Specifies the name of a session configuration. This cmdlet connects 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 parameter's value is used to select and filter sessions. It doesn't 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 that is used to reconnect to the disconnected session.

The URI must be fully qualified. The string's format is as follows:

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

The default value is as follows:

`http://localhost:5985/WSMAN`

If you don't specify a connection URI, you can use the UseSSL , ComputerName , Port , and ApplicationName parameters to specify the connection URI 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 don't specify a port, the session is created with standard 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.

-Credential <System.Management.Automation.PSCredential>

Specifies a user account that has permission to connect to the disconnected session. 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).

-Id <System.Int32>

Specifies the ID of a disconnected session. The Id parameter works only when the disconnected session was previously connected to the current session.

This parameter is valid, but not effective, when the session is stored on the local computer, but wasn't connected to the current session.

-InstanceId <System.Guid>

Specifies the instance ID of the disconnected session. The instance ID is

a GUID that uniquely identifies a PSSession on a local or remote computer.
The instance ID is stored in the InstanceID property of the PSSession .

-JobName <System.String>
Specifies a friendly name for the job that `Receive-PSSession` returns.

`Receive-PSSession` returns a job when the value of the OutTarget parameter is Job or the job that's running in the disconnected session was started in the current session.

If the job that's running in the disconnected session was started in the current session, PowerShell reuses the original job object in the session and ignores the value of the JobName parameter.

If the job that's running in the disconnected session was started in a different session, PowerShell creates a new job object. It uses a default name, but you can use this parameter to change the name.

If the default value or explicit value of the OutTarget parameter isn't Job, the command succeeds, but the JobName parameter has no effect.

-Name <System.String>
Specifies the friendly name of the disconnected session.

-OutTarget <Microsoft.PowerShell.Commands.OutTarget>
Determines how the session results are returned. The acceptable values for this parameter are:

- Job . Returns the results asynchronously in a job object. You can use the JobName parameter   to specify a name or new name for the job. - Host . Returns the results to the command line (synchronously). If the command is being resumed   or the results consist of a large number of objects, the response might be delayed.

The default value of the OutTarget parameter is Host. If the command that's being received in a disconnected session was started in the current session, the default value of the OutTarget parameter is the form in which the command was started. If the command was started as a job, by default, it's returned as a job. Otherwise, it's returned to the host program by default.

Typically, the host program displays returned objects at the command line without delay, but this behavior can vary.

-Port <System.Int32>

Specifies the remote computer's network port that's used to reconnect to the session. To connect to a remote computer, it 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 on 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>`

Don't use the Port parameter unless it's necessary. The port that's 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.

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

Specifies the disconnected session. Enter a variable that contains the

PSSession or a command that creates or gets the PSSession , such as a `Get-PSSession` command.

-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's 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 don't 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).

-UseSSL <System.Management.Automation.SwitchParameter>

Indicates that this cmdlet uses the Secure Sockets Layer (SSL) protocol to connect to the disconnected session. By default, SSL isn't used.

WS-Management encrypts all PowerShell content transmitted over the network. UseSSL is an additional protection that sends the data across an HTTPS connection instead of an HTTP connection.

    If you use this parameter and SSL isn't available on the port that's used
    for the command, the command fails.

-Confirm <System.Management.Automation.SwitchParameter>
    Prompts you for confirmation before running the cmdlet.

-WhatIf <System.Management.Automation.SwitchParameter>
    Shows what would happen if the cmdlet runs. The cmdlet isn't run.

<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: Connect to a PSSession --------------

```
Receive-PSSession -ComputerName Server01 -Name ITTask
```

The `Receive-PSSession` specifies the remote computer with the ComputerName parameter. The Name parameter identifies the ITTask session on the Server01 computer. The example gets the results of commands that were running in the ITTask session.

Because the command doesn't use the OutTarget parameter, the results appear on the command line.
Example 2: Get results of all commands on disconnected sessions

```
Get-PSSession -ComputerName Server01, Server02 | Receive-PSSession
```

`Get-PSSession` uses the ComputerName parameter to specify the remote computers. The objects are sent down the pipeline to `Receive-PSSession`.
- Example 3: Get the results of a script running in a session -

```
$parms = @{
  ComputerName = "Server01"
  Name = "ITTask"
  OutTarget = "Job"
  JobName = "ITTaskJob01"
  Credential = "Domain01\Admin01"
}
Receive-PSSession @parms

Id     Name           State        HasMoreData     Location
--     ----           -----        -----------     --------
16     ITTaskJob01    Running      True            Server01
```

The command uses the ComputerName and Name parameters to identify the disconnected session. It uses the OutTarget parameter with a value of Job to direct `Receive-PSSession` to return the results as a job. The JobName parameter specifies a name for the job in the reconnected session. The Credential parameter runs the `Receive-PSSession` command using the permissions of a domain administrator.

The output shows that `Receive-PSSession` returned the results as a job in the current session. To get the job results, use a `Receive-Job` command

-------- Example 4: Get results after a network outage --------

```
PS> $s = New-PSSession -ComputerName Server01 -Name AD -ConfigurationName
ADEndpoint
PS> $s

Id  Name   ComputerName    State       ConfigurationName     Availability
--  ----   ------------    -----       -----------------     ------------
8   AD     Server01        Opened      ADEndpoint            Available
```

```
PS> Invoke-Command -Session $s -FilePath
\\Server12\Scripts\SharedScripts\New-ADResolve.ps1

Running "New-ADResolve.ps1"

# Network outage
# Restart local computer
# Network access is not re-established within 4 minutes


PS> Get-PSSession -ComputerName Server01

Id  Name   ComputerName    State        ConfigurationName      Availability
--  ----   ------------    -----        -----------------      ------------
1  Backup  Server01       Disconnected   Microsoft.PowerShell          None
8  AD      Server01       Disconnected   ADEndpoint                    None


PS> Receive-PSSession -ComputerName Server01 -Name AD -OutTarget Job -JobName
AD

Job Id   Name      State        HasMoreData     Location
--       ----      -----        -----------     --------
16       ADJob     Running      True            Server01


PS> Get-PSSession -ComputerName Server01

Id  Name    ComputerName    State        ConfigurationName     Availability
--  ----    ------------    -----        -----------------     ------------
1  Backup   Server01       Disconnected  Microsoft.PowerShell         Busy
8  AD       Server01       Opened       ADEndpoint              Available
```

The `New-PSSession` cmdlet creates a session on the Server01 computer and saves the session in the `$s` variable. The `$s` variable displays that the State is Opened and the Availability is Available. These values indicate that you're connected to the session and can run commands in the session.

The `Invoke-Command` cmdlet runs a script in the session in the `$s` variable. The script begins to run and return data, but a network outage occurs that interrupts the session. The user has to exit the session and restart the local computer.

When the computer restarts, the user starts PowerShell and runs a `Get-PSSession` command to get sessions on the Server01 computer. The output shows that the AD session still exists on the Server01 computer. The State indicates that the AD session is disconnected. The Availability value of None, indicates that the session isn't connected to any client sessions.

The `Receive-PSSession` cmdlet reconnects to the AD session and gets the results of the script that ran in the session. The command uses the OutTarget parameter to request the results in a job named ADJob . The command returns a job object and the output indicates that the script is still running.

The `Get-PSSession` cmdlet is used to check the job state. The output confirms that the `Receive-PSSession` cmdlet reconnected to the AD session, which is now open and available for commands. And, the script resumed execution and is getting the script results.

-------- Example 5: Reconnect to disconnected sessions --------

```
PS> $parms = @{
    InDisconnectedSession = $True
    ComputerName = "Server01", "Server02", "Server30"
    FilePath = "\\Server12\Scripts\SharedScripts\Get-BugStatus.ps1"
    Name = "BugStatus"
```

```
      SessionOption = @{IdleTimeout = 86400000}
      ConfigurationName = "ITTasks"
    }
PS> Invoke-Command @parms
PS> Exit


PS> $s = Get-PSSession -ComputerName Server01, Server02, Server30 -Name
BugStatus
PS> $s

Id  Name   ComputerName    State        ConfigurationName     Availability
--  ----   ------------    -----        -----------------     ------------
1  ITTask  Server01       Disconnected  ITTasks               None
8  ITTask  Server02       Disconnected  ITTasks               None
2  ITTask  Server30       Disconnected  ITTasks               None


PS> $Results = Receive-PSSession -Session $s
PS> $s

Id  Name   ComputerName    State        ConfigurationName     Availability
--  ----   ------------    -----        -----------------     ------------
1  ITTask  Server01       Opened       ITTasks             Available
8  ITTask  Server02       Opened       ITTasks             Available
2  ITTask  Server30       Opened       ITTasks             Available


PS> $Results

Bug Report - Domain 01
----------------------
ComputerName        BugCount        LastUpdated
```

```
--------------        ---------         ------------
Server01              121               Friday, December 30, 2011 5:03:34 PM
```

The `Invoke-Command` cmdlet runs a script on three remote computers. Because the script gathers and summarizes data from multiple databases, it often takes the script an extended time to finish. The command uses the InDisconnectedSession parameter that starts the scripts and then immediately disconnects the sessions. The SessionOption parameter extends the IdleTimeout value of the disconnected session. Disconnected sessions are considered idle from the moment they're disconnected. It's important to set the idle time-out for long enough so that the commands can complete and you can reconnect to the session. You can set the IdleTimeout only when you create the PSSession and change it only when you disconnect from it. You can't change the IdleTimeout value when you connect to a PSSession or receiving its results. After running the command, the user exits PowerShell and closes the computer.

The next day, the user resumes Windows, starts PowerShell, and uses `Get-PSSession` to get the sessions in which the scripts were running. The command identifies the sessions by the computer name, session name, and the name of the session configuration and saves the sessions in the `$s` variable. The value of the `$s` variable is displayed and shows that the sessions are disconnected, but aren't busy.

The `Receive-PSSession` cmdlet connects to the sessions in the `$s` variable and gets their results. The command saves the results in the `$Results` variable. The `$s` variable is displayed and shows that the sessions are connected and available for commands.

The script results in the `$Results` variable are displayed in the PowerShell console. If any of the results are unexpected, the user can run commands in the sessions to investigate the root cause.

------ Example 6: Running a job in a disconnected session ------

```
PS> $s = New-PSSession -ComputerName Server01 -Name Test
PS> $j = Invoke-Command -Session $s { 1..1500 | Foreach-Object {"Return $_";
sleep 30}} -AsJob
PS> $j

Id     Name          State        HasMoreData     Location
--     ----          -----        -----------     --------
16     Job1          Running      True            Server01



PS> $s | Disconnect-PSSession

Id Name   ComputerName    State        ConfigurationName     Availability
-- ----   ------------    -----        -----------------     ------------
1  Test   Server01        Disconnected  Microsoft.PowerShell         None



PS> $j

Id     Name          State        HasMoreData     Location
--     ----          -----        -----------     --------
16     Job1          Disconnected  True            Server01



PS> Receive-Job $j -Keep

Return 1
Return 2



PS> $s2 = Connect-PSSession -ComputerName Server01 -Name Test
PS> $j2 = Receive-PSSession -ComputerName Server01 -Name Test
PS> Receive-Job $j
```

Return 3

Return 4

The `New-PSSession` cmdlet creates the Test session on the Server01 computer.
The command saves the session in the `$s` variable.

The `Invoke-Command` cmdlet runs a command in the session in the `$s`
variable. The command uses the AsJob parameter to run the command as a job and
creates the job object in the current session. The command returns a job
object that's saved in the `$j` variable. The `$j` variable displays the job
object.

The session object in the `$s` variable is sent down the pipeline to
`Disconnect-PSSession` and the session is disconnected.

The `$j` variable is displayed and shows the effect of disconnecting the job
object in the `$j` variable. The job state is now Disconnected.

The `Receive-Job` is run on the job in the `$j` variable. The output shows
that the job began to return output before the session and the job were
disconnected.

The `Connect-PSSession` cmdlet is run in the same client session. The command
reconnects to the Test session on the Server01 computer and saves the session
in the `$s2` variable.

The `Receive-PSSession` cmdlet gets the results of the job that was running in
the session. Because the command is run in the same session,
`Receive-PSSession` returns the results as a job by default and reuses the
same job object. The command saves the job in the `$j2` variable. The
`Receive-Job` cmdlet gets the results of the job in the `$j` variable.

To see the examples, type: "get-help Receive-PSSession -examples".

For more information, type: "get-help Receive-PSSession -detailed".

For technical information, type: "get-help Receive-PSSession -full".

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