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

PowerShell Get-Help on command 'Connect-WSMan'

PS C:\Users\wahid> Get-Help Connect-WSMan

NAME

    Connect-WSMan

SYNOPSIS

    Connects to the WinRM service on a remote computer.

SYNTAX

```
    Connect-WSMan [[-ComputerName] <System.String>] [-ApplicationName
    <System.String>] [-Authentication {None | Default | Digest | Negotiate | Basic
    | Kerberos | ClientCertificate | Credssp}] [-CertificateThumbprint
    <System.String>] [-Credential <System.Management.Automation.PSCredential>]
    [-OptionSet <System.Collections.Hashtable>] [-Port <System.Int32>]
    [-SessionOption <Microsoft.WSMan.Management.SessionOption>] [-UseSSL]
    [<CommonParameters>]

    Connect-WSMan [-Authentication {None | Default | Digest | Negotiate | Basic |
    Kerberos | ClientCertificate | Credssp}] [-CertificateThumbprint
    <System.String>] [-ConnectionURI <System.Uri>] [-Credential
    <System.Management.Automation.PSCredential>] [-OptionSet
    <System.Collections.Hashtable>] [-Port <System.Int32>] [-SessionOption
```

<Microsoft.WSMan.Management.SessionOption>] [<CommonParameters>]

DESCRIPTION

The `Connect-WSMan` cmdlet connects to the WinRM service on a remote computer, and it establishes a persistent connection to the remote computer. You can use this cmdlet in the context of the WSMan provider to connect to the WinRM service on a remote computer. However, you can also use this cmdlet to connect to the WinRM service on a remote computer before you change to the WSMan provider. The remote computer appears in the root directory of the WSMan provider.

Explicit credentials are required when the client and server computers are in different domains or workgroups.

For information about how to disconnect from the WinRM service on a remote computer, see the `Disconnect-WSMan` cmdlet.

PARAMETERS

-ApplicationName <System.String>

Specifies the application name in the connection. The default value of the ApplicationName parameter is WSMAN. The complete identifier for the remote endpoint is in the following format:

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

For example: `http://server01:8080/WSMAN`

Internet Information Services (IIS), which hosts the session, forwards requests with this endpoint to the specified application. This default setting of WSMAN is appropriate for most uses. This parameter is designed to be used if many computers establish remote connections to one computer

that is running Windows PowerShell. In this case, IIS hosts Web Services
for Management (WS-Management) for efficiency.

-Authentication <Microsoft.WSMan.Management.AuthenticationMechanism>
Specifies the authentication mechanism to be used at the server. The
acceptable values for this parameter are:

- `Basic`- Basic is a scheme in which the user name and password are sent
in clear text to the   server or proxy. - `Default` - Use the
authentication method implemented by the WS-Management protocol. This is
the   default. - `Digest` - Digest is a challenge-response scheme that
uses a server-specified data string for the   challenge. - `Kerberos` -
The client computer and the server mutually authenticate by using Kerberos
  certificates. - `Negotiate` - Negotiate is a challenge-response scheme
that negotiates with the server or proxy to   determine the scheme to use
for authentication. For example, this parameter value allows for
negotiation to determine whether the Kerberos protocol or NTLM is used. -
`CredSSP` - Use Credential Security Support Provider (CredSSP)
authentication, which lets the user   delegate credentials. This option is
designed for commands that run on one remote computer but   collect data
from or run additional commands on other remote computers.

> [!CAUTION] > CredSSP delegates the user credentials from the local
computer to a remote computer. This practice > increases the security risk
of the remote operation. If the remote computer is compromised, when >
credentials are passed to it, the credentials 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 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 thumbprint, use the `Get-Item` or `Get-ChildItem` command in the Windows PowerShell Cert: drive.

-ComputerName <System.String>

Specifies the computer against which to run the management operation. The value can be a fully qualified domain name, a NetBIOS name, or an IP address. Use the local computer name, use localhost, or use a dot (`.`) to specify the local computer. The local computer is the default. When the remote computer is in a different domain from the user, you must use a fully qualified domain name must be used. You can pipe a value for this parameter to the cmdlet.

-ConnectionURI <System.Uri>

Specifies the connection endpoint. The format of this string is as follows:

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

The following string is a correctly formatted value for this parameter:

`http://Server01:8080/WSMAN`

The URI must be fully qualified.

-Credential <System.Management.Automation.PSCredential>

Specifies a user account that has permission to perform this action. The default is the current user. Type a user name, such as `User01`, `Domain01\User01`, or `User@Domain.com`. Or, enter a PSCredential object, such as one returned by the `Get-Credential` cmdlet. When you type a user name, this cmdlet prompts you for a password.

-OptionSet <System.Collections.Hashtable>

Specifies a set of switches to a service to modify or refine the nature of the request. These resemble switches used in command-line shells because they are service specific. Any number of options can be specified.

The following example demonstrates the syntax that passes the values 1, 2, and 3 for the a, b, and c parameters:

`-OptionSet @{a=1;b=2;c=3}`

-Port <System.Int32>

Specifies the port to use when the client connects to the WinRM service. When the transport is HTTP, the default port is 80. When the transport is HTTPS, the default port is 443.

When you use HTTPS as the transport, the value of the ComputerName parameter must match the server's certificate common name (CN). However, if the SkipCNCheck parameter is specified as part of the SessionOption parameter, the certificate common name of the server does not have to match the host name of the server. The SkipCNCheck parameter should be used only for trusted computers.

-SessionOption <Microsoft.WSMan.Management.SessionOption>

Specifies extended options for the WS-Management session. Enter a SessionOption object that you create by using the `New-WSManSessionOption` cmdlet. For more information about the options that are available, type `Get-Help New-WSManSessionOption`.

-UseSSL <System.Management.Automation.SwitchParameter>

Specifies that the Secure Sockets Layer (SSL) protocol is used to establish a connection to the remote computer. By default, SSL is not used.

    WS-Management encrypts all the Windows PowerShell content that is
    transmitted over the network. The UseSSL parameter lets you specify the
    additional protection of HTTPS instead of HTTP. If SSL is not available on
    the port that is used for the connection, and you specify this parameter,
    the command fails.

<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 remote computer -----------

```
PS C:\> Connect-WSMan -ComputerName "server01"
PS C:\> cd wsman:
PS WSMan:\>
PS WSMan:\> dir

WSManConfig: Microsoft.WSMan.Management\WSMan::WSMan

ComputerName                           Type
------------                           ----
localhost                              Container
server01                               Container
```

This command creates a connection to the remote server01 computer.

The `Connect-WSMan` cmdlet is generally used in the context of the WSMan provider to connect to a remote computer, in this case the server01 computer. However, you can use the cmdlet to establish connections to remote computers before you change to the WSMan provider. Those connections appear in the ComputerName list.

Example 2: Connect to a remote computer by using Administrator credentials

```
PS C:\> $cred = Get-Credential Administrator
PS C:\> Connect-WSMan -ComputerName "server01" -Credential $cred
PS C:\> cd wsman:
PS WSMan:\>
PS WSMan:\> dir

WSManConfig: Microsoft.WSMan.Management\WSMan::WSMan

ComputerName                         Type
------------                        ----
localhost                           Container
server01                            Container
```

This command creates a connection to the remote system server01 using the Administrator account credentials.

The first command uses the Get-Credential cmdlet to get the Administrator credentials and then stores them in the `$cred` variable. `Get-Credential` prompts you for a password of username and password through a dialog box or at the command line, depending on system registry settings.

The second command uses the Credential parameter to pass the credentials stored in $cred to `Connect-WSMan`. `Connect-WSMan` then connects to the remote system server01 by using the Administrator credentials.

Example 3: Connect to a remote computer over a specified port

```
PS C:\> Connect-WSMan -ComputerName "server01" -Port 80
PS C:\> cd wsman:
PS WSMan:\>
PS WSMan:\> dir
```

```
WSManConfig: Microsoft.WSMan.Management\WSMan::WSMan
ComputerName                    Type
------------                    ----
localhost                       Container
server01                        Container
```

This command creates a connection to the remote server01 computer over port 80.

Example 4: Connect to a remote computer by using connection options

```
PS C:\> $a = New-WSManSessionOption -OperationTimeout 30000
PS C:\> Connect-WSMan -ComputerName "server01" -SessionOption $a
PS C:\> cd wsman:
PS WSMan:\> dir

WSManConfig: Microsoft.WSMan.Management\WSMan::WSMan
ComputerName                    Type
------------                    ----
localhost                       Container
server01                        Container
```

This example creates a connection to the remote server01 computer by using the connection options that are defined in the `New-WSManSessionOption` command.

The first command uses `New-WSManSessionOption` to store a set of connection setting options in the `$a` variable. In this case, the session options set a connection time out of 30 seconds (30,000 milliseconds).

The second command uses the SessionOption parameter to pass the credentials that are stored in the `$a` variable to `Connect-WSMan`. Then, `Connect-WSMan` connects to the remote server01 computer by using the specified session options.

REMARKS

To see the examples, type: "get-help Connect-WSMan -examples".

For more information, type: "get-help Connect-WSMan -detailed".

For technical information, type: "get-help Connect-WSMan -full".

For online help, type: "get-help Connect-WSMan -online"