Using JEA

07/10/20196 minutes to read

In this article

This article describes the various ways you can connect to and use a JEA endpoint.

Using JEA interactively

If you're testing your JEA configuration or have simple tasks for users, you can use JEA the same way you would a regular PowerShell remoting session. For complex remoting tasks, it's recommended to use implicit remoting. Implicit remoting allows users to operate with the data objects locally.

To use JEA interactively, you need:

- The name of the computer you're connecting to (can be the local machine)
- The name of the JEA endpoint registered on that computer
- Credentials that have access to the JEA endpoint on that computer

Given that information, you can start a JEA session using the New-PSSession or Enter-PSSession cmdlets.

PowerShell

Copy

```
$nonAdminCred = Get-Credential
Enter-PSSession -ComputerName localhost -ConfigurationName JEAMaintenance -Credential $nonAdminCred
```

If the current user account has access to the JEA endpoint, you can omit the **Credential** parameter.

When the PowerShell prompt changes to `[localhost]: PS>` you know that you're now interacting with the remote JEA session. You can run `Get-Command` to check which commands are available. Consult with your administrator to learn if there are any restrictions on the available parameters or allowed parameter values.

Remember, JEA sessions operate in NoLanguage mode. Some of the ways you typically use PowerShell may not be available. For instance, you can't use variables to store data or inspect the properties on objects returned from cmdlets. The following example shows two approaches to get the same commands to work in NoLanguage mode.

PowerShell

Copy

```
# Using variables is prohibited in NoLanguage mode. The following will not work:
# $vm = Get-VM -Name 'SQL01'
# Start-VM -VM $vm

# You can use pipes to pass data through to commands that accept input from the pipeline
Get-VM -Name 'SQL01' | Start-VM

# You can also wrap subcommands in parentheses and enter them inline as arguments
Start-VM -VM (Get-VM -Name 'SQL01')
```

```powershell
# You can also use parameter sets that don't require extra data to be passed in
Start-VM -VMName 'SQL01'
```

For more complex command invocations that make this approach difficult, consider using implicit remoting or creating custom functions that wrap the functionality you require.

Using JEA with implicit remoting

PowerShell has an implicit remoting model that lets you import proxy cmdlets from a remote machine and interact with them as if they were local commands. Implicit remoting is explained in this **Hey, Scripting Guy!** blog post. Implicit remoting is useful when working with JEA because it allows you to work with JEA cmdlets in a full language mode. You can use tab completion, variables, manipulate objects, and even use local scripts to automate tasks against a JEA endpoint. Anytime you invoke a proxy command, the data is sent to the JEA endpoint on the remote machine and executed there.

Implicit remoting works by importing cmdlets from an existing PowerShell session. You can optionally choose to prefix the nouns of each proxy cmdlet with a string of your choosing. The prefix allows you to distinguish the commands that are for the remote system. A temporary script module containing all the proxy commands is created and imported for the duration of your local PowerShell session.

PowerShell

Copy

```powershell
# Create a new PSSession to your JEA endpoint
$jeasession = New-PSSession -ComputerName 'SERVER01' -ConfigurationName 'JEAMaintenance'
```

```powershell
# Import the entire PSSession and prefix each imported cmdlet with "JEA"
Import-PSSession -Session $jeasession -Prefix 'JEA'

# Invoke "Get-Command" on the remote JEA endpoint using the proxy cmdlet
Get-JEACommand
```

Important

Some systems may not be able to import an entire JEA session due to constraints in the default JEA cmdlets. To get around this, only import the commands you need from the JEA session by explicitly providing their names to the `-CommandName` parameter. A future update will address the issue with importing entire JEA sessions on affected systems.

If you're unable to import a JEA session because of JEA constraints on the default parameters, follow the steps below to filter out the default commands from the imported set. You can continue use commands like `Select-Object`, but you'll just use the local version installed on your computer instead of the one imported from the remote JEA session.

PowerShell

Copy

```powershell
# Create a new PSSession to your JEA endpoint
$jeasession = New-PSSession -ComputerName 'SERVER01' -ConfigurationName 'JEAMaintenance'

# Get a list of all the commands on the JEA
```

```
endpoint
$commands = Invoke-Command -Session $jeasession -ScriptBlock { Get-Command }

# Filter out the default cmdlets
$jeaDefaultCmdlets = 'Clear-Host', 'Exit-PSSession', 'Get-Command', 'Get-FormatData', 'Get-Help', 'Measure-Object', 'Out-Default', 'Select-Object'
$filteredCommands = $commands.Name | Where-Object { $jeaDefaultCmdlets -notcontains $_ }

# Import only commands explicitly added in role capabilities and prefix each imported cmdlet with "JEA"
Import-PSSession -Session $jeasession -Prefix 'JEA' -CommandName $filteredCommands
```

You can also persist the proxied cmdlets from implicit remoting using Export-PSSession. For more information about implicit remoting, see the documentation for Import-PSSession and Import-Module.

Using JEA programmatically

JEA can also be used in automation systems and in user applications, such as in-house helpdesk apps and websites. The approach is the same as that for building apps that talk to unconstrained PowerShell endpoints. Ensure the program is designed to work with limitation imposed by JEA.

For simple, one-off tasks, you can use Invoke-Command to run commands in a JEA session.

PowerShell

Copy

```
Invoke-Command -ComputerName 'SERVER01' -ConfigurationName 'JEAMaintenance' -ScriptBlock { Get-Process; Get-Service }
```

To check which commands are available for use when you connect to a JEA session, run `Get-Command` and iterate through the results to check for the allowed parameters.

PowerShell

Copy

```
$allowedCommands = Invoke-Command -ComputerName 'SERVER01' -ConfigurationName 'JEAMaintenance' -ScriptBlock { Get-Command }
$allowedCommands | Where-Object { $_.CommandType -in 'Function', 'Cmdlet' } | Format-Table Name, Parameters
```

If you're building a C# app, you can create a PowerShell runspace that connects to a JEA session by specifying the configuration name in a WSManConnectionInfo object.

C#

Copy

```
// using System.Management.Automation;
var computerName = "SERVER01";
var configName   = "JEAMaintenance";
// See https://docs.microsoft.com/dotnet/api/system.management.automation.pscredential
var creds        = // create a PSCredential object here

WSManConnectionInfo connectionInfo = new WSManConnectionInfo(
    false,                 // Use SSL
    computerName,          // Computer name
    5985,                  // WSMan Port
    "/wsman",              // WSMan Path
                           // Connection URI with config name
    string.Format(CultureInfo.InvariantCulture, "https://schemas.microsoft.com/powershell/{0}", configName),
    creds);                // Credentials

// Now, use the connection info to create a runspace where you can run the commands
using (Runspace runspace = RunspaceFactory.CreateRunspace(connectionInfo))
{
    // Open the runspace
    runspace.Open();

    using (PowerShell ps = PowerShell.Create())
    {
        // Set the PowerShell object to use the JEA runspace
        ps.Runspace = runspace;

        // Now you can add and invoke commands
        ps.AddCommand("Get-Command");
        foreach (var result in ps.Invoke())
        {
            Console.WriteLine(result);
        }
    }

    // Close the runspace
    runspace.Close();
}
```

Using JEA with PowerShell Direct

Hyper-V in Windows 10 and Windows Server 2016 offers PowerShell Direct, a feature that allows Hyper-V administrators to manage virtual machines with PowerShell regardless of the network configuration or remote management settings on the virtual machine.

You can use PowerShell Direct with JEA to give a Hyper-V administrator limited access to your VM. This can be useful if you lose network connectivity to your VM and need a datacenter admin to fix the network settings.

No additional configuration is required to use JEA over PowerShell Direct. However, the guest operating system running inside the virtual machine must be Windows 10, Windows Server 2016, or higher. The Hyper-V admin can connect to the JEA endpoint by using the `-VMName` or `-VMId` parameters on PSRemoting cmdlets:

PowerShell

Copy

```powershell
# Entering a JEA session using PowerShell Direct when the VM name is unique
Enter-PSSession -VMName 'SQL01' -ConfigurationName 'NICMaintenance' -Credential 'localhost\JEAformyHoster'

# Entering a JEA session using PowerShell Direct using VM ids
$vm = Get-VM -VMName 'MyVM' | Select-Object -First 1
Enter-PSSession -VMId $vm.VMId -ConfigurationName 'NICMaintenance' -Credential 'localhost\JEAformyHoster'
```

It's recommended you create a dedicated user account with the minimum rights needed to manage the system for use by a Hyper-V administrator. Remember, even an unprivileged user can sign into a Windows machine by default, including using unconstrained PowerShell. That allows them to browse the file system and learn more about your OS environment. To lock down a Hyper-V administrator and limit them to only access a VM using PowerShell Direct with JEA, you must deny local logon rights to the Hyper-V admin's JEA account.