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

PowerShell Get-Help on command 'Enter-PSHostProcess'

PS C:\Users\wahid> Get-Help Enter-PSHostProcess

NAME

Enter-PSHostProcess

SYNOPSIS

Connects to and enters into an interactive session with a local process.

SYNTAX

Enter-PSHostProcess [-HostProcessInfo]

<Microsoft.PowerShell.Commands.PSHostProcessInfo> [[-AppDomainName]

<System.String>] [<CommonParameters>]

Enter-PSHostProcess [-Id] <System.Int32> [[-AppDomainName] <System.String>]

[<CommonParameters>]

Enter-PSHostProcess [-Name] <System.String> [[-AppDomainName] <System.String>]

[<CommonParameters>]

Enter-PSHostProcess [-Process] <System.Diagnostics.Process> [[-AppDomainName]

<System.String>] [<CommonParameters>]

DESCRIPTION

The `Enter-PSHostProcess` cmdlet connects to and enters into an interactive session with a local process.

Instead of creating a new process to host PowerShell and run a remote session, the remote, interactive session is run in an existing process that is already running PowerShell. When you are interacting with a remote session on a specified process, you can enumerate running runspaces, and then select a runspace to debug by running either `Debug-Runspace` or `Enable-RunspaceDebug`.

The process that you want to enter must be hosting PowerShell (System.Management.Automation.dll). You must be either a member of the Administrators group on the computer on which the process is found, or you must be the user who is running the script that started the process.

After you have selected a runspace to debug, a remote debug session is opened for the runspace if it is either currently running a command or is stopped in the debugger. You can then debug the runspace script in the same way you would debug other remote session scripts.

Detach from a debugging session, and then the interactive session with the process, by running exit twice, or stop script execution by running the existing debugger quit command.

If you specify a process by using the Name parameter, and there is only one process found with the specified name, the process is entered. If more than one process with the specified name is found, PowerShell returns an error, and lists all processes found with the specified name.

To support attaching to processes on remote computers, the `Enter-PSHostProcess` cmdlet is enabled in a specified remote computer, so that you can attach to a local process within a remote PowerShell session.

PARAMETERS

-AppDomainName <System.String>

Specifies an application domain name to connect to if omitted, uses DefaultAppDomain . Use `Get-PSHostProcessInfo` to display the application domain names.

-HostProcessInfo <Microsoft.PowerShell.Commands.PSHostProcessInfo>

Specifies a PSHostProcessInfo object that can be connected to with PowerShell. Use `Get-PSHostProcessInfo` to get the object.

-Id <System.Int32>

Specifies a process by the process ID. To get a process ID, run the `Get-Process` cmdlet.

-Name <System.String>

Specifies a process by the process name. To get a process name, run the `Get-Process` cmdlet. You can also get process names from the Properties dialog box of a process in Task Manager.

-Process <System.Diagnostics.Process>

Specifies a process by the process object. The simplest way to use this parameter is to save the results of a `Get-Process` command that returns process that you want to enter in a variable, and then specify the variable as the value of this parameter.

<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 Part 1: Start debugging a runspace within the PowerShell ISE process

```
PS C:\> Enter-PSHostProcess -Name powershell_ise
[Process:1520]: PS C:\>  Get-Runspace
Id    Name          InstanceId                           State
Availability
--    -------       -----------                          ------
-------------
1     Runspace1     2d91211d-9cce-42f0-ab0e-71ac258b32b5     Opened
Available
2     Runspace2     a3855043-cb16-424a-a616-685360c3763b     Opened
RemoteDebug
3     MyLocalRS     2236dbd8-2105-4dec-a15a-a27d0bfaacb5     Opened
LocalDebug
4     MyRunspace    771356e9-8c44-4b70-9de5-dd17cb41e48e     Opened
Busy
5     Runspace8     3e517382-a97a-49ba-9c3c-fd21f6664288     Broken
None
```

---------- Example part 2: Debug a specific runspace ----------

```
[Process:1520]: PS C:\>  (Get-Runspace -Id 4).ScriptStackTrace
Command                 Arguments                       Location
-------                 ---------                       --------
MyModuleWorkflowF1        {}
TestNoFile3.psm1: line 6
WFTest1                 {}
TestNoFile2.ps1: line 14
TestNoFile2.ps1           {}
TestNoFile2.ps1: line 22
<ScriptBlock>             {}                            <No file>
```

[Process: 1520]: PS C:\> Debug-Runspace -Id 4

Hit Line breakpoint on 'C:\TestWFVar1.ps1:83'

At C:\TestWFVar1.ps1:83 char:1

+ $scriptVar = "Script Variable"

+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

[Process: 1520]: [RSDBG: 4]: PS C:\>

Start an interactive debugging session with this runspace by running the

`Debug-Runspace` cmdlet.

---- Example part 3: Finish the debugging session and exit ----

[Process:346]: [RSDBG: 3]: PS C:\> exit

[Process:1520]: PS C:\>

[Process:1520]: PS C:\> Exit-PSHostProcess

PS C:\>

REMARKS

To see the examples, type: "get-help Enter-PSHostProcess -examples".

For more information, type: "get-help Enter-PSHostProcess -detailed".

For technical information, type: "get-help Enter-PSHostProcess -full".

For online help, type: "get-help Enter-PSHostProcess -online"