Windows PowerShell Get-Help on Cmdlet 'Wait-Process'

PS:\>Get-HELP Wait-Process -Full

NAME

Wait-Process

SYNOPSIS

Waits for the processes to be stopped before accepting more input.

SYNTAX

Wait-Process [-Id] <System.Int32[]> [[-Timeout] <System.Int32>] [<CommonParameters>]

Wait-Process [[-Timeout] <System.Int32>] -InputObject <System.Diagnostics.Process[]> [<CommonParameters>]

Wait-Process [-Name] <System.String[]> [[-Timeout] <System.Int32>] [<CommonParameters>]

DESCRIPTION

The `Wait-Process` cmdlet waits for one or more running processes to be stopped before accepting input. In the PowerShell console, this cmdlet suppresses the command

prompt until the processes are stopped. You can specify a process by process name or process ID (PID), or pipe a process object to `Wait-Process`.

`Wait-Process` works only on processes running on the local computer.

PARAMETERS

-Id <System.Int32[]>

Specifies the process IDs of the processes. To specify multiple IDs, use commas to separate the IDs. To find the PID of a process, type `Get-Process`.

Required? true

Position? 0

Default value None

Accept pipeline input? True (ByPropertyName)

Accept wildcard characters? false

-InputObject <System.Diagnostics.Process[]>

Specifies the processes by submitting process objects. Enter a variable that contains the process objects, or type a command or expression that gets the process

objects, such as the `Get-Process` cmdlet.

Required? true

Position? named

Default value None

Accept pipeline input? True (ByValue)

Accept wildcard characters? false

-Name <System.String[]>

Specifies the process names of the processes. To specify multiple names, use commas to separate the names. Wildcard characters are not supported.

Required? true

Position? 0

Default value None

Accept pipeline input? True (ByPropertyName)

Accept wildcard characters? false

-Timeout <System.Int32>

Specifies the maximum time, in seconds, that this cmdlet waits for the specified processes to stop. When this interval expires, the command displays a

non-terminating error that lists the processes that are still running, and ends the wait. By default, there is no time-out.

Required? false

Position? 1

Default value None

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.Diagnostics.Process

You can pipe a process object to this cmdlet.

OUTPUTS

None

This cmdlet returns no output.

NOTES

- This cmdlet uses the WaitForExit method of the System.Diagnostics.Process class.

- Unlike `Start-Process -Wait`, `Wait-Process` only waits for the processes identified. `Start-Process -Wait` waits for the process tree (the process and all its descendants) to exit before returning control.

---------- Example 1: Stop a process and wait ----------

```
PS C:\> $nid = (Get-Process notepad).id
PS C:\> Stop-Process -Id $nid
PS C:\> Wait-Process -Id $nid
```

This example stops the Notepad process and then waits for the process to be stopped before it continues with the next command.

The first command uses the `Get-Process` cmdlet to get the ID of the Notepad process. It stores the ID in the `$nid` variable.

The second command uses the `Stop-Process` cmdlet to stop the process with the ID stored in `$nid`.

The third command uses `Wait-Process` to wait until the Notepad process is stopped. It uses the Id parameter of `Wait-Process` to identify the process.

---------- Example 2: Specifying a process ----------

```
PS C:\> $p = Get-Process notepad
PS C:\> Wait-Process -Id $p.id
PS C:\> Wait-Process -Name "notepad"
PS C:\> Wait-Process -InputObject $p
```

These commands show three different methods of specifying a process to `Wait-Process`. The first command gets the Notepad process and stores it in the `$p` variable.

The second command uses the Id parameter, the third command uses the Name parameter, and the fourth command

uses the InputObject parameter.

These commands have the same results and can be used interchangeably.

------ Example 3: Wait for processes for a specified time ------

PS C:\> Wait-Process -Name outlook, winword -Timeout 30

This command waits 30 seconds for the Outlook and Winword processes to stop. If both processes are not stopped, the cmdlet displays a non-terminating error and the

command prompt.

RELATED LINKS

Online Version: https://learn.microsoft.com/powershell/module/microsoft.powershell.management/wait-process?view=powershell-5.1&WT.mc_id=ps-gethelp

Debug-Process

Get-Process

Start-Process

Stop-Process

Wait-Process