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

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

NAME

Wait-Event

SYNOPSIS

Waits until a particular event is raised before continuing to run.

SYNTAX

Wait-Event [[-SourceIdentifier] <System.String>] [-Timeout <System.Int32>] [<CommonParameters>]

DESCRIPTION

The `Wait-Event` cmdlet suspends execution of a script or function until a particular event is raised. Execution resumes when the event is detected. To cancel the

wait, press <kbd>CTRL</kbd>+<kbd>C</kbd>.

This feature provides an alternative to polling for an event. It also allows you to determine the response to an event in two different ways:

- using the Action parameter of the event subscription - waiting for an event to return and then respond with an action

PARAMETERS

-SourceIdentifier <System.String>

Specifies the source identifier that this cmdlet waits for events. By default, `Wait-Event` waits for any event.

Required? false

Position? 0

Default value None

Accept pipeline input? True (ByPropertyName)

Accept wildcard characters? false

-Timeout <System.Int32>

Specifies the maximum time, in seconds, that `Wait-Event` waits for the event to occur. The default, -1, waits indefinitely. The timing starts when you submit the

`Wait-Event` command.

If the specified time is exceeded, the wait ends and the command prompt returns, even if the event has not been raised. No error message is displayed.

Required? false

Position? named

Default value -1

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.String

OUTPUTS

System.Management.Automation.PSEventArgs

NOTES

Events, event subscriptions, and the event queue exist only in the current session. If you close the current session, the event queue is discarded and the event subscription is canceled.

---------- Example 1: Wait for the next event ----------

```
Wait-Event
```

Example 2: Wait for an event with a specified source identifier

```
Wait-Event -SourceIdentifier "ProcessStarted"
```

---------- Example 3: Wait for a timer elapsed event ----------

```
$Timer = New-Object Timers.Timer
$objectEventArgs = @{
    InputObject = $Timer
    EventName = 'Elapsed'
    SourceIdentifier = 'Timer.Elapsed'
```

```
}
Register-ObjectEvent @objectEventArgs
$Timer.Interval = 2000
$Timer.Autoreset = $False
$Timer.Enabled = $True
Wait-Event Timer.Elapsed

ComputerName     :
RunspaceId       : bb560b14-ff43-48d4-b801-5adc31bbc6fb
EventIdentifier  : 1
Sender           : System.Timers.Timer
SourceEventArgs  : System.Timers.ElapsedEventArgs
SourceArgs       : {System.Timers.Timer, System.Timers.ElapsedEventArgs}
SourceIdentifier : Timer.Elapsed
TimeGenerated    : 4/23/2020 2:30:37 PM
MessageData      :
```

---- Example 4: Wait for an event after a specified timeout ----

```
Wait-Event -SourceIdentifier "ProcessStarted" -Timeout 90
```

RELATED LINKS

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

Get-Event

Get-EventSubscriber

New-Event

Register-EngineEvent

Register-ObjectEvent

Remove-Event

Unregister-Event

Wait-Event