PowerShell Get-Help on command 'Register-ArgumentCompleter'

PS C:\Users\wahid> Get-Help Register-ArgumentCompleter

NAME

    Register-ArgumentCompleter

SYNOPSIS

    Registers a custom argument completer.

SYNTAX

    Register-ArgumentCompleter [-CommandName <System.String[]>] [-Native]
    -ScriptBlock <System.Management.Automation.ScriptBlock> [<CommonParameters>]

    Register-ArgumentCompleter [-CommandName <System.String[]>] -ParameterName
    <System.String> -ScriptBlock <System.Management.Automation.ScriptBlock>
    [<CommonParameters>]

DESCRIPTION

    The `Register-ArgumentCompleter` cmdlet registers a custom argument completer.
    An argument completer allows you to provide dynamic tab completion, at run
    time for any command that you specify.

PARAMETERS

-CommandName <System.String[]>

Specifies the name of the commands as an array.

-Native <System.Management.Automation.SwitchParameter>

Indicates that the argument completer is for a native command where PowerShell cannot complete parameter names.

-ParameterName <System.String>

Specifies the name of the parameter whose argument is being completed. The parameter name specified cannot be an enumerated value, such as the ForegroundColor parameter of the `Write-Host` cmdlet.

For more information on enums, see about_Enum (./About/about_Enum.md).

-ScriptBlock <System.Management.Automation.ScriptBlock>

Specifies the commands to run to perform tab completion. The script block you provide should return the values that complete the input. The script block must unroll the values using the pipeline (`ForEach-Object`, `Where-Object`, etc.), or another suitable method. Returning an array of values causes PowerShell to treat the entire array as one tab completion value.

The script block must accept the following parameters in the order specified below. The names of the parameters aren't important because PowerShell passes in the values by position.

- `$commandName` (Position 0) - This parameter is set to the name of the command for which the script block is providing tab completion. - `$parameterName` (Position 1) - This parameter is set to the parameter whose value requires tab completion. - `$wordToComplete` (Position 2) - This parameter is set to value the user has provided before they pressed

<kbd>Tab</kbd>. Your script block should use this value to determine tab completion   values. - `$commandAst` (Position 3) - This parameter is set to the Abstract Syntax   Tree (AST) for the current input line. For more information, see Ast Class (/dotnet/api/system.management.automation.language.ast). - `$fakeBoundParameters` (Position 4) - This parameter is set to a hashtable containing the   `$PSBoundParameters` for the cmdlet, before the user pressed <kbd>Tab</kbd>. For more information,   see about_Automatic_Variables (./About/about_Automatic_Variables.md).

When you specify the Native parameter, the script block must take the following parameters in the specified order. The names of the parameters aren't important because PowerShell passes in the values by position.

- `$wordToComplete` (Position 0) - This parameter is set to value the user has provided before they   pressed <kbd>Tab</kbd>. Your script block should use this value to determine tab completion   values. - `$commandAst` (Position 1) - This parameter is set to the Abstract Syntax Tree (AST) for the current input line. For more information, see Ast Class (/dotnet/api/system.management.automation.language.ast). - `$cursorPosition` (Position 2) - This parameter is set to the position of the cursor when the user   pressed <kbd>Tab</kbd>.

You can also provide an ArgumentCompleter as a parameter attribute. For more information, see about_Functions_Advanced_Parameters (./About/about_Functions_Advanced_Parameters.md).

<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: Register a custom argument completer -------

```
$scriptBlock = {
    param($commandName, $parameterName, $wordToComplete, $commandAst,
$fakeBoundParameters)

    (Get-TimeZone -ListAvailable).Id | Where-Object {
        $_ -like "$wordToComplete*"
    } | ForEach-Object {
          "'$_'"
    }
}
Register-ArgumentCompleter -CommandName Set-TimeZone -ParameterName Id
-ScriptBlock $scriptBlock
```

The first command creates a script block which takes the required parameters which are passed in when the user presses <kbd>Tab</kbd>. For more information, see the ScriptBlock parameter description.

Within the script block, the available values for Id are retrieved using the `Get-TimeZone` cmdlet. The Id property for each Time Zone is piped to the `Where-Object` cmdlet. The `Where-Object` cmdlet filters out any ids that do not start with the value provided by `$wordToComplete`, which represents the text the user typed before they pressed <kbd>Tab</kbd>. The filtered ids are piped to the `ForEach-Object` cmdlet which encloses each value in quotes, should the value contain spaces.

The second command registers the argument completer by passing the scriptblock, the ParameterName Id and the CommandName `Set-TimeZone`.

----- Example 2: Add details to your tab completion values -----

```
$s = {
    param($commandName, $parameterName, $wordToComplete, $commandAst,
$fakeBoundParameters)
    $services = Get-Service | Where-Object {$_.Status -eq "Running" -and
$_.Name -like "$wordToComplete*"}
    $services | ForEach-Object {
        New-Object -Type System.Management.Automation.CompletionResult
-ArgumentList $_.Name,
            $_.Name,
            "ParameterValue",
            $_.Name
    }
}
Register-ArgumentCompleter -CommandName Stop-Service -ParameterName Name
-ScriptBlock $s
```

The first command creates a script block which takes the required parameters which are passed in when the user presses <kbd>Tab</kbd>. For more information, see the ScriptBlock parameter description.

Within the script block, the first command retrieves all running services using the `Where-Object` cmdlet. The services are piped to the `ForEach-Object` cmdlet. The `ForEach-Object` cmdlet creates a new `[System.Management.Automation.CompletionResult]` (/dotnet/api/system.management.automation.completionresult)object and populates it with the values of the current service (represented by the pipeline variable `$_`).

The CompletionResult object allows you to provide additional details to each returned value:

- completionText (String) - The text to be used as the auto completion result. This is the value   sent to the command. - listItemText (String) - The text to be displayed in a list, such as when the user presses <kbd>Ctrl</kbd>+<kbd>Space</kbd>. This is used for display only and is not passed to the command   when selected. - resultType (CompletionResultType (/dotnet/api/system.management.automation.completionresulttype))- The type of completion result. - toolTip (String) - The text for the tooltip with details to be displayed about the object.   This is visible when the user selects an item after pressing <kbd>Ctrl</kbd>+<kbd>Space</kbd>.

The last command demonstrates that stopped services can still be passed in manually to the `Stop-Service` cmdlet. The tab completion is the only aspect affected.

---- Example 3: Register a custom Native argument completer ----

```
$scriptblock = {
    param($wordToComplete, $commandAst, $cursorPosition)
        dotnet complete --position $cursorPosition $commandAst.ToString() |
ForEach-Object {
            [System.Management.Automation.CompletionResult]::new($_, $_,
'ParameterValue', $_)
        }
}
Register-ArgumentCompleter -Native -CommandName dotnet -ScriptBlock
$scriptblock
```

The first command creates a script block which takes the required parameters which are passed in when the user presses <kbd>Tab</kbd>. For more information, see the ScriptBlock parameter description.

Within the script block, the `dotnet complete` command is used to perform the tab completion. The results are piped to the `ForEach-Object` cmdlet which use the new static method of the System.Management.Automation.CompletionResult (/dotnet/api/system.management.automation.completionresult)class to create a new CompletionResult object for each value.

REMARKS

To see the examples, type: "get-help Register-ArgumentCompleter -examples".

For more information, type: "get-help Register-ArgumentCompleter -detailed".

For technical information, type: "get-help Register-ArgumentCompleter -full".

For online help, type: "get-help Register-ArgumentCompleter -online"