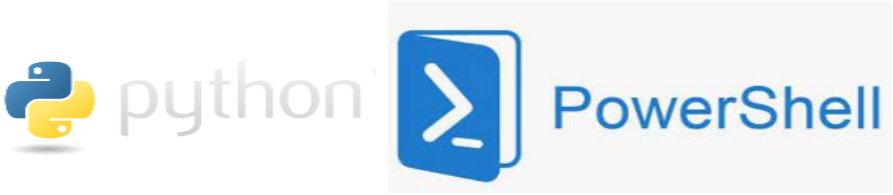

Full credit is given to the above companies including the OS that this PDF file was generated!

PowerShell Get-Help on command 'New-ItemProperty'

PS C:\Users\wahid> Get-Help New-ItemProperty

NAME

New-ItemProperty

SYNOPSIS

Creates a new property for an item and sets its value.

SYNTAX

New-ItemProperty [-Name] <System.String> [-Credential
<System.Management.Automation.PSCredential>] [-Exclude <System.String[]>]
[-Filter <System.String>] [-Force] [-Include <System.String[]>] -LiteralPath
<System.String[]> [-PropertyType <System.String>] [-UseTransaction] [-Value
<System.Object>] [-Confirm] [-WhatIf] [<CommonParameters>]

New-ItemProperty [-Path] <System.String[]> [-Name] <System.String>
[-Credential <System.Management.Automation.PSCredential>] [-Exclude
<System.String[]>] [-Filter <System.String>] [-Force] [-Include
<System.String[]>] [-PropertyType <System.String>] [-UseTransaction] [-Value
<System.Object>] [-Confirm] [-WhatIf] [<CommonParameters>]

DESCRIPTION

The `New-ItemProperty` cmdlet creates a new property for a specified item and sets its value. Typically, this cmdlet is used to create new registry values, because registry values are properties of a registry key item.

This cmdlet does not add properties to an object.

- To add a property to an instance of an object, use the `Add-Member` cmdlet.

- To add a property to all objects of a particular type, modify the Types.ps1xml file.

PARAMETERS

-Credential <System.Management.Automation.PSCredential>

Specifies a user account that has permission to perform this action. The default is the current user.

Type a user name, such as `User01` or `Domain01\User01`, or enter a PSCredential object, such as one generated by the `Get-Credential` cmdlet. If you type a user name, you are prompted for a password.

> [!NOTE] > This parameter is not supported by any providers installed with PowerShell. > To impersonate another user, or elevate your credentials when running this cmdlet, > use Invoke-Command (../Microsoft.PowerShell.Core/Invoke-Command.md).

-Exclude <System.String[]>

Specifies, as a string array, a property or property that this cmdlet excludes from the operation. The value of this parameter qualifies the Path parameter. Enter a path element or pattern, such as `*.txt`. Wildcard characters are permitted.

-Filter <System.String>

Specifies a filter in the format or language of the provider. The value of this parameter qualifies the Path parameter.

The syntax of the filter, including the use of wildcard characters, depends on the provider. Filters are more efficient than other parameters, because the provider applies them when the cmdlet gets the objects rather than having PowerShell filter the objects after they are retrieved.

-Force <System.Management.Automation.SwitchParameter>

Forces the cmdlet to create a property on an object that cannot otherwise be accessed by the user. Implementation varies from provider to provider. For more information, see about_Providers (../Microsoft.PowerShell.Core/About/about_Providers.md).

-Include <System.String[]>

Specifies, as a string array, an item or items that this cmdlet includes in the operation. The value of this parameter qualifies the Path parameter. Enter a path element or pattern, such as ` .txt`. Wildcard characters are permitted. The Include * parameter is effective only when the command includes the contents of an item, such as `C:\Windows*`, where the wildcard character specifies the contents of the `C:\Windows` directory.

-LiteralPath <System.String[]>

Specifies a path to one or more locations. The value of LiteralPath is used exactly as it is typed. No characters are interpreted as wildcards. If the path includes escape characters, enclose it in single quotation marks (`'`). Single quotation marks tell PowerShell not to interpret any characters as escape sequences.

For more information, see about_Quoting_Rules (../Microsoft.Powershell.Core/About/about_Quoting_Rules.md).

-Name <System.String>

Specifies a name for the new property. If the property is a registry entry, this parameter specifies the name of the entry.

-Path <System.String[]>

Specifies the path of the item. This parameter identifies the item to which this cmdlet adds the new property.

-PropertyType <System.String>

Specifies the type of property that this cmdlet adds. The acceptable values for this parameter are:

- `String`: Specifies a null-terminated string. Used for REG_SZ values. - `ExpandString`: Specifies a null-terminated string that contains unexpanded references to environment variables that are expanded when the value is retrieved. Used for REG_EXPAND_SZ values. - `Binary`: Specifies binary data in any form. Used for REG_BINARY values. - `DWord`: Specifies a 32-bit binary number. Used for REG_DWORD values. - `MultiString`: Specifies an array of null-terminated strings terminated by two null characters. Used for REG_MULTI_SZ values. - `Qword`: Specifies a 64-bit binary number. Used for REG_QWORD values. - `Unknown`: Indicates an unsupported registry data type, such as REG_RESOURCE_LIST values.

-UseTransaction <System.Management.Automation.SwitchParameter>

Includes the command in the active transaction. This parameter is valid only when a transaction is in progress. For more information, see about_Transactions (../Microsoft.PowerShell.Core/About/about_Transactions.md).

-Value <System.Object>

Specifies the property value. If the property is a registry entry, this parameter specifies the value of the entry.

-Confirm <System.Management.Automation.SwitchParameter>

Prompts you for confirmation before running the cmdlet.

-WhatIf <System.Management.Automation.SwitchParameter>

Shows what would happen if the cmdlet runs. The cmdlet is not run.

<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: Add a registry entry ----------

```
New-ItemProperty -Path "HKLM:\Software\MyCompany" -Name "NoOfEmployees" -Value
822
Get-ItemProperty "HKLM:\Software\MyCompany"

PSPath       :
Microsoft.PowerShell.Core\Registry::HKEY_LOCAL_MACHINE\software\mycompany
PSParentPath  : Microsoft.PowerShell.Core\Registry::HKEY_LOCAL_MACHINE\software
PSChildName   : mycompany
PSDrive       : HKLM
PSProvider    : Microsoft.PowerShell.Core\Registry
NoOfLocations : 2
NoOfEmployees : 822
```

---------- Example 2: Add a registry entry to a key ----------

```
Get-Item -Path "HKLM:\Software\MyCompany" | New-ItemProperty -Name
NoOfLocations -Value 3
```

This command works because the parameter-binding feature of Windows PowerShell associates the path of the RegistryKey object that `Get-Item` returns with the LiteralPath parameter of `New-ItemProperty`. For more information, see about_Pipelines (../Microsoft.PowerShell.Core/About/about_pipelines.md).

Example 3: Create a MultiString value in the registry using a Here-String

```
$newValue = New-ItemProperty -Path "HKLM:\SOFTWARE\ContosoCompany\" -Name
'HereString' -PropertyType MultiString -Value @"
This is text which contains newlines
It can also contain "quoted" strings
"@
$newValue.multistring
```

```
This is text which contains newlines
It can also contain "quoted" strings
```

Example 4: Create a MultiString value in the registry using an array

```
$newValue = New-ItemProperty -Path "HKLM:\SOFTWARE\ContosoCompany\" -Name
'MultiString' -PropertyType MultiString -Value ('a','b','c')
$newValue.multistring[0]
```

```
a
```

REMARKS

To see the examples, type: "get-help New-ItemProperty -examples".

For more information, type: "get-help New-ItemProperty -detailed".

For technical information, type: "get-help New-ItemProperty -full".

For online help, type: "get-help New-ItemProperty -online"