PowerShell Get-Help on command 'Set-CimInstance'

PS C:\Users\wahid> Get-Help Set-CimInstance

```
NAME
    Set-CimInstance

SYNOPSIS
    Modifies a CIM instance on a CIM server by calling the ModifyInstance method
    of the CIM class.


SYNTAX
    Set-CimInstance [-Query] <System.String> -CimSession
    <Microsoft.Management.Infrastructure.CimSession[]> [-Namespace
    <System.String>] [-OperationTimeoutSec <System.UInt32>] [-PassThru] [-Property
    <System.Collections.IDictionary>] [-QueryDialect <System.String>] [-Confirm]
    [-WhatIf] [<CommonParameters>]

    Set-CimInstance [-InputObject]
    <Microsoft.Management.Infrastructure.CimInstance> -CimSession
    <Microsoft.Management.Infrastructure.CimSession[]> [-OperationTimeoutSec
    <System.UInt32>] [-PassThru] [-Property <System.Collections.IDictionary>]
    [-ResourceUri <System.Uri>] [-Confirm] [-WhatIf] [<CommonParameters>]
```

```
Set-CimInstance [-InputObject]
<Microsoft.Management.Infrastructure.CimInstance> [-ComputerName
<System.String[]>] [-OperationTimeoutSec <System.UInt32>] [-PassThru]
[-Property <System.Collections.IDictionary>] [-ResourceUri <System.Uri>]
[-Confirm] [-WhatIf] [<CommonParameters>]

Set-CimInstance [-Query] <System.String> [-ComputerName <System.String[]>]
[-Namespace <System.String>] [-OperationTimeoutSec <System.UInt32>]
[-PassThru] [-Property <System.Collections.IDictionary>] [-QueryDialect
<System.String>] [-Confirm] [-WhatIf] [<CommonParameters>]
```

DESCRIPTION

This cmdlet modifies a CIM instance on a CIM server.

If the InputObject parameter is not specified, the cmdlet works in one of the following ways:

- If neither the ComputerName parameter nor the CimSession parameter is specified, then this   cmdlet works on local Windows Management Instrumentation (WMI) using a Component Object Model   (COM) session. - If either the ComputerName parameter or the CimSession parameter is specified, then this   cmdlet works against the CIM server specified by either the ComputerName parameter or the CimSession parameter.

If the InputObject parameter is specified, the cmdlet works in one of the following ways:

- If neither the ComputerName parameter nor the CimSession parameter is specified, then this   cmdlet uses the CIM session or computer name from the input object. - If the either the ComputerName parameter or the CimSession parameter is specified, then   this cmdlet uses the either the CimSession parameter value or ComputerName parameter   value. This is not very common.

PARAMETERS

-CimSession <Microsoft.Management.Infrastructure.CimSession[]>

Runs the cmdlets on a remote computer. Enter a computer name or a session object, such as the output of a `New-CimSession` or `Get-CimSession` cmdlet.

-ComputerName <System.String[]>

Specifies the name of the computer on which you want to run the CIM operation. You can specify a fully qualified domain name (FQDN) or a NetBIOS name.

If you do not specify this parameter, the cmdlet performs the operation on the local computer using Component Object Model (COM).

If you specify this parameter, the cmdlet creates a temporary session to the specified computer using the WsMan protocol.

If multiple operations are being performed on the same computer, connecting using a CIM session gives better performance.

-InputObject <Microsoft.Management.Infrastructure.CimInstance>

Specifies a CIM instance object to use as input.

The InputObject parameter doesn't enumerate over collections. If a collection is passed, an error is thrown. When working with collections, pipe the input to enumerate the values.

-Namespace <System.String>

Specifies the namespace for the CIM operation. The default namespace is root/cimv2. You can use tab completion to browse the list of namespaces, because PowerShell gets a list of namespaces from the local WMI server to provide the list of namespaces.

-OperationTimeoutSec <System.UInt32>

Specifies the amount of time that the cmdlet waits for a response from the computer. By default, the value of this parameter is 0, which means that the cmdlet uses the default timeout value for the server.

If the OperationTimeoutSec parameter is set to a value less than the robust connection retry timeout of 3 minutes, network failures that last more than the value of the OperationTimeoutSec parameter are not recoverable, because the operation on the server times out before the client can reconnect.

-PassThru <System.Management.Automation.SwitchParameter>

Returns an object representing the item with which you are working. By default, this cmdlet does not generate any output.

-Property <System.Collections.IDictionary>

Specifies the properties of the CIM instance as a hash table (using name-value pairs). Only the properties specified using this parameter are changed. Other properties of the CIM instance are not changed.

-Query <System.String>

Specifies a query to run on the CIM server to retrieve CIM instances on which to run the cmdlet. You can specify the query dialect using the QueryDialect parameter.

If the value specified contains double quotes (`"`), single quotes (`'`), or a backslash (``), you must escape those characters by prefixing them with the backslash (``) character. If the value specified uses the WQL LIKE operator, then you must escape the following characters by enclosing them in square brackets (`[]`): percent (`%`), underscore (`_`), or opening square bracket (`[`).

-QueryDialect <System.String>

Specifies the query language used for the Query parameter. The acceptable values for this parameter are: WQL or CQL . The default value is WQL .

-ResourceUri <System.Uri>

Specifies the resource uniform resource identifier (URI) of the resource class or instance. The URI is used to identify a specific type of resource, such as disks or processes, on a computer.

A URI consists of a prefix and a path to a resource. For example:

- `http://schemas.microsoft.com/wbem/wsman/1/wmi/root/cimv2/Win32_LogicalDisk`

- `http://intel.com/wbem/wscim/1/amt-schema/1/AMT_GeneralSettings`

By default, if you do not specify this parameter, the DMTF standard resource URI `http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/` is used and the class name is appended to it.

ResourceURI can only be used with CIM sessions created using the WSMan protocol, or when specifying the ComputerName parameter, which creates a CIM session using WSMan. If you specify this parameter without specifying the ComputerName parameter, or if you specify a CIM session created using DCOM protocol, you will get an error, because the DCOM protocol does not support the ResourceURI parameter.
If both the ResourceUri parameter and the Filter parameter are specified, the Filter parameter is ignored.

-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: Set the CIM instance ---------------

```
Set-CimInstance -Query 'Select * from Win32_Environment where name LIKE
"testvar%"' -Property @{VariableValue="abcd"}
```

--- Example 2: Set the CIM instance property using pipeline ---

```
Get-CimInstance -Query 'Select * from Win32_Environment where name LIKE
"testvar%"' |
  Set-CimInstance -Property @{VariableValue="abcd"}
```

- Example 3: Set the CIM instance property using input object -

```
$x = Get-CimInstance -Query 'Select * from Win32_Environment where
Name="testvar"'
Set-CimInstance -InputObject $x -Property @{VariableValue="somevalue"}
-PassThru
```

This example retrieves the CIM instance objects filtered by the Query parameter in to a variable `$x` using `Get-CimInstance`, and then passes the contents of the variable to the `Set-CimInstance` cmdlet. `Set-CimInstance` then modifies the VariableValue property to somevalue . Because the Passthru parameter is used, This example returns a modified CIM instance object.

----------- Example 4: Set the CIM instance property -----------

```
$x = Get-CimInstance -Query 'Select * from Win32_Environment where
name="testvar"'
$x.VariableValue = "Change"
Set-CimInstance -CimInstance $x -PassThru
```

Example 5: Show the list of CIM instances to modify using WhatIf

```
Set-CimInstance -Query 'Select * from Win32_Environment where name LIKE
"testvar%"' -Property @{VariableValue="abcd"} -WhatIf
```

Example 6: Set the CIM instance after confirmation from the user

```
Set-CimInstance -Query 'Select * from Win32_Environment where name LIKE
"testvar%"' -Property @{VariableValue="abcd"} -Confirm
```

----------- Example 7: Set the created CIM instance -----------

```
$x = New-CimInstance -ClassName Win32_Environment -Property
@{Name="testvar";UserName="domain\user"} -Key Name,UserName -ClientOnly
Set-CimInstance -CimInstance $x -Property @{VariableValue="somevalue"}
-PassThru
```

REMARKS

To see the examples, type: "get-help Set-CimInstance -examples".

For more information, type: "get-help Set-CimInstance -detailed".

For technical information, type: "get-help Set-CimInstance -full".

For online help, type: "get-help Set-CimInstance -online"