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

PowerShell Get-Help on command 'New-CimInstance'

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

NAME

New-CimInstance

SYNOPSIS

Creates a CIM instance.

SYNTAX

New-CimInstance [-CimClass] <Microsoft.Management.Infrastructure.CimClass>
[[-Property] <System.Collections.IDictionary>] -CimSession
<Microsoft.Management.Infrastructure.CimSession[]> [-ClientOnly]
[-OperationTimeoutSec <System.UInt32>] [-Confirm] [-WhatIf]
[<CommonParameters>]

New-CimInstance [-CimClass] <Microsoft.Management.Infrastructure.CimClass>
[[-Property] <System.Collections.IDictionary>] [-ClientOnly] [-ComputerName
<System.String[]>] [-OperationTimeoutSec <System.UInt32>] [-Confirm] [-WhatIf]
[<CommonParameters>]

New-CimInstance [-ClassName] <System.String> [[-Property]
<System.Collections.IDictionary>] -CimSession

<Microsoft.Management.Infrastructure.CimSession[]> [-ClientOnly] [-Key <System.String[]>] [-Namespace <System.String>] [-OperationTimeoutSec <System.UInt32>] [-Confirm] [-WhatIf] [<CommonParameters>]

New-CimInstance [[-Property] <System.Collections.IDictionary>] -CimSession <Microsoft.Management.Infrastructure.CimSession[]> [-Key <System.String[]>] [-Namespace <System.String>] [-OperationTimeoutSec <System.UInt32>] -ResourceUri <System.Uri> [-Confirm] [-WhatIf] [<CommonParameters>]

New-CimInstance [-ClassName] <System.String> [[-Property] <System.Collections.IDictionary>] [-ClientOnly] [-ComputerName <System.String[]>] [-Key <System.String[]>] [-Namespace <System.String>] [-OperationTimeoutSec <System.UInt32>] [-Confirm] [-WhatIf] [<CommonParameters>]

New-CimInstance [[-Property] <System.Collections.IDictionary>] [-ComputerName <System.String[]>] [-Key <System.String[]>] [-Namespace <System.String>] [-OperationTimeoutSec <System.UInt32>] -ResourceUri <System.Uri> [-Confirm] [-WhatIf] [<CommonParameters>]

DESCRIPTION

The `New-CimInstance` cmdlet creates an instance of a CIM class based on the class definition on either the local computer or a remote computer. By default, the `New-CimInstance` cmdlet creates an instance on the local computer.

PARAMETERS

-CimClass <Microsoft.Management.Infrastructure.CimClass>

Specifies a CIM class object that represents the type of the instance. Use the `Get-CimClass` cmdlet to retrieve the class declaration from a computer. Using this parameter results in better client side schema

validations.

-CimSession <Microsoft.Management.Infrastructure.CimSession[]>
Runs the command using the specified CIM session. Enter a variable that contains the CIM session, or a command that creates or gets the CIM session, such as the `New-CimSession` or `Get-CimSession` cmdlets. For more information, see about_CimSession (../Microsoft.PowerShell.Core/About/about_CimSession.md).

-ClassName <System.String>
Specifies the name of the CIM class of which the operation creates an instance. NOTE: You can use tab completion to browse the list of classes, because PowerShell gets a list of classes from the local WMI server to provide a list of class names.

-ClientOnly <System.Management.Automation.SwitchParameter>
Indicates that the instance is only created in PowerShell without going to the CIM server. You can use this parameter to create an in-memory CIM instance for use in subsequent PowerShell operations.

-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), a NetBIOS name, or an IP address.

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

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

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

-Key <System.String[]>

Specifies the properties that are used as keys. CimSession and ComputerName cannot be used when Key is specified.

-Namespace <System.String>

Specifies the namespace of the class for the new instance. 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 CIM server. 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.

-Property <System.Collections.IDictionary>

Specifies the properties of the CIM instance using a hash table (name-value pairs).

If you specify the CimClass parameter, then the `New-CimInstance` cmdlet performs a property validation on the client to make sure that the properties specified are consistent with the class declaration on the server. If the CimClass parameter is not specified, then the property validation is done on the server.

-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: Create an instance of a CIM class ---------

```
New-CimInstance -ClassName Win32_Environment -Property
@{Name="testvar";VariableValue="testvalue";UserName="domain\user"}
```

No client side validation is performed if the class does not exist, the properties are wrong, or if the server rejects the call. If the instance is created successfully, the cmdlet outputs the newly created instance.

Example 2: Create an instance of a CIM class using a class schema

```
$class = Get-CimClass -ClassName Win32_Environment
New-CimInstance -CimClass $class -Property
@{Name="testvar";VariableValue="testvalue";UserName="Contoso\User1"}
```

------ Example 3: Create a dynamic instance on the client ------

```
$a = New-CimInstance -ClassName Win32_Process -Property @{Handle=0} -Key
Handle -ClientOnly
Get-CimInstance -CimInstance $a
Invoke-CimMethod -CimInstance $a -MethodName GetOwner

ProcessId Name               HandleCount WorkingSetSize VirtualSize
--------- ----               ----------- -------------- -----------
0         System Idle Process 0          8192           8192

Domain         :
ReturnValue    : 2
User           :
PSComputerName :
```

The `Get-CimInstance` cmdlet then retrieves a particular single instance. The `Invoke-CimMethod` cmdlet calls the GetOwner method on the retrieved instance.

Example 4: Create an instance for a CIM class of a specific namespace

```
$class = Get-CimClass -ClassName MSFT_Something -Namespace root/somewhere
New-CimInstance -CimClass $class -Property @{"Prop1"=1;"Prop2"="value"}
-ClientOnly
```

In this example, using the CimClass parameter instead of the ClassName parameter validates that Prop1 and Prop2 actually exist and that the keys are marked correctly.

You cannot use the ComputerName or CimSession parameter with the ClientOnly parameter.

REMARKS

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

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

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

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