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

PowerShell Get-Help on command 'Add-Member'

PS C:\Users\wahid> Get-Help Add-Member

NAME

Add-Member

SYNOPSIS

Adds custom properties and methods to an instance of a PowerShell object.

SYNTAX

Add-Member [-MemberType] {AliasProperty | CodeMethod | CodeProperty |

NoteProperty | ScriptMethod | ScriptProperty} [-Name] <System.String>

[[-Value] <System.Object>] [[-SecondValue] <System.Object>] [-Force]

-InputObject <System.Management.Automation.PSObject> [-PassThru] [-TypeName

<System.String>] [<CommonParameters>]

Add-Member [-NotePropertyName] <System.String> [-NotePropertyValue]

<System.Object> [-Force] -InputObject <System.Management.Automation.PSObject>

[-PassThru] [-TypeName <System.String>] [<CommonParameters>]

Add-Member [-NotePropertyMembers] <System.Collections.IDictionary> [-Force]

-InputObject <System.Management.Automation.PSObject> [-PassThru] [-TypeName

<System.String>] [<CommonParameters>]

Add-Member -InputObject <System.Management.Automation.PSObject> [-PassThru] [-TypeName <System.String>] [<CommonParameters>]

DESCRIPTION

The `Add-Member` cmdlet lets you add members (properties and methods) to an instance of a PowerShell object. For instance, you can add a NoteProperty member that contains a description of the object or a ScriptMethod member that runs a script to change the object.

To use `Add-Member`, pipe the object to `Add-Member`, or use the InputObject parameter to specify the object.

The MemberType parameter indicates the type of member that you want to add. The Name parameter assigns a name to the new member, and the Value parameter sets the value of the member.

The properties and methods that you add are added only to the particular instance of the object that you specify. `Add-Member` doesn't change the object type. To create a new object type, use the `Add-Type` cmdlet.

You can also use the `Export-Clixml` cmdlet to save the instance of the object, including the additional members, in a file. Then you can use the `Import-Clixml` cmdlet to re-create the instance of the object from the information that's stored in the exported file.

Beginning in Windows PowerShell 3.0, `Add-Member` has new features that make it easier to add note properties to objects. You can use the NotePropertyName and NotePropertyValue parameters to define a note property or use the NotePropertyMembers parameter, which takes a hash table of note property names and values.

Also, beginning in Windows PowerShell 3.0, the PassThru parameter, which generates an output object, is needed less frequently. `Add-Member` now adds the new members directly to the input object of more types. For more information, see the PassThru parameter description.

PARAMETERS

-Force <System.Management.Automation.SwitchParameter>

By default, `Add-Member` can't add a new member if the object already has a member with the same. When you use the Force parameter, `Add-Member` replaces the existing member with the new member. You can't use the Force parameter to replace a standard member of a type.

-InputObject <System.Management.Automation.PSObject>

Specifies the object to which the new member is added. Enter a variable that contains the objects, or type a command or expression that gets the objects.

-MemberType <System.Management.Automation.PSMemberTypes>

Specifies the type of the member to add. This parameter is required. The acceptable values for this parameter are:

- AliasProperty

- CodeMethod

- CodeProperty

- NoteProperty

- ScriptMethod

- ScriptProperty

For information about these values, see PSMemberTypes Enumeration (xref:System.Management.Automation.PSMemberTypes)in the PowerShell SDK. Not all objects have every type of member. If you specify a member type that the object doesn't have, PowerShell returns an error.

-Name <System.String>

Specifies the name of the member that this cmdlet adds.

-NotePropertyMembers <System.Collections.IDictionary>

Specifies a hashtable or ordered dictionary that contains key-value pair representing NoteProperty names and their values. For more information about hash tables and ordered dictionaries in PowerShell, see about_Hash_Tables (../Microsoft.PowerShell.Core/About/about_Hash_Tables.md).

This parameter was introduced in Windows PowerShell 3.0.

-NotePropertyName <System.String>

Specifies the note property name.

Use this parameter with the NotePropertyValue parameter. This parameter is optional.

This parameter was introduced in Windows PowerShell 3.0.

-NotePropertyValue <System.Object>

Specifies the note property value.

Use this parameter with the NotePropertyName parameter. This parameter is optional.

This parameter was introduced in Windows PowerShell 3.0.

-PassThru <System.Management.Automation.SwitchParameter>

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

For most objects, `Add-Member` adds the new members to the input object. However, when the input object is a string, `Add-Member` can't add the member to the input object. For these objects, use the PassThru parameter to create an output object.

In Windows PowerShell 2.0, `Add-Member` added members only to the PSObject wrapper of objects, not to the object. Use the PassThru parameter to create an output object for any object that has a PSObject wrapper.

-SecondValue <System.Object>

Specifies optional additional information about AliasProperty , ScriptProperty , or CodeProperty members.

If used when adding an AliasProperty , this parameter must be a data type. A conversion to the specified data type is added to the value of the AliasProperty . For example, if you add an AliasProperty that provides an alternate name for a string property, you can also specify a SecondValue parameter of System.Int32 to indicate that the value of that string property should be converted to an integer when accessed using the corresponding AliasProperty .

For a CodeProperty , the value must be a reference to a method that implements a Set accessor. Use the `GetMethod()` method of a type reference to get a reference to a method. The method must take a single parameter that's a PSObject . The Get accessor is assigned using the Value parameter.

For a ScriptProperty , the value must be a script block that implements a Set accessor. The Get accessor is assigned using the Value parameter.

-TypeName <System.String>

Specifies a name for the type.

When the type is a class in the System namespace or a type that has a type accelerator, you can enter the short name of the type. Otherwise, the full type name is required. This parameter is effective only when the InputObject is a PSObject .

This parameter was introduced in Windows PowerShell 3.0.

-Value <System.Object>

Specifies the initial value of the added member. If you add an AliasProperty , CodeProperty , or ScriptProperty member, you can supply additional information using the SecondValue parameter.

- For an AliasProperty , the value must be the name of the property being aliased. - For a CodeMethod , the value must be a reference to a method. Use the `GetMethod()` method of a type reference to get a reference to a method. - For a CodeProperty , the value must be a reference to a method that implements a Get accessor. Use the `GetMethod()` method of a type reference to get a reference to a method. reference. The method must take a single parameter that's a PSObject . The Set accessor is assigned using the SecondValue parameter. - For a ScriptMethod , the value must be a script block. - For a ScriptProperty , the value must be a script block that implements a Get accessor. The Set accessor is assigned using the SecondValue parameter.

<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 note property to a PSObject ---------

```
$A = Get-ChildItem c:\ps-test\test.txt
$A | Add-Member -NotePropertyName Status -NotePropertyValue Done
$A.Status
```

Done

-------- Example 2: Add an alias property to a PSObject --------

```
$A = Get-ChildItem C:\Temp\test.txt
$A | Add-Member -MemberType AliasProperty -Name Size -Value Length
$A.Size
```

2394

----- Example 3: Add a StringUse note property to a string -----

```
$A = "A string"
$A = $A | Add-Member -NotePropertyMembers @{StringUse="Display"} -PassThru
$A.StringUse
```

Display

----- Example 4: Add a script method to a FileInfo object -----

```
$A = Get-ChildItem C:\Temp\test.txt
$S = {[math]::Round(($this.Length / 1MB), 2)}
$A | Add-Member -MemberType ScriptMethod -Name "SizeInMB" -Value $S
$A.SizeInMB()

0.43
```

---------- Example 5: Create a custom object ----------

```
$Asset = New-Object -TypeName PSObject
$Asset | Add-Member -NotePropertyMembers @{Name="Server30"} -TypeName Asset
$Asset | Add-Member -NotePropertyMembers @{System="Server Core"}
$Asset | Add-Member -NotePropertyMembers @{PSVersion="4.0"}
$Asset | Get-Member -MemberType Properties

TypeName: Asset

Name        MemberType   Definition
----        ----------   ----------
Name        NoteProperty string Name=Server30
PSVersion   NoteProperty string PSVersion=4.0
System      NoteProperty string System=Server Core

$Asset.PSObject.Properties | Format-Table Name, MemberType, TypeNameOfValue,
Value

Name        MemberType TypeNameOfValue Value
----        ---------- --------------- -----
Name      NoteProperty System.String   Server30
System    NoteProperty System.String   Server Core
PSVersion NoteProperty System.String   4.0
```

Inspecting the raw list of properties shows the properties in the order that
they were added to the object. `Format-Table` is used in this example to
create output similar to `Get-Member`.
--------- Example 6: Add an AliasProperty to an object ---------

```
PS> $obj = [pscustomobject]@{
    Name = 'Doris'
    Age = '20'
}
PS> $obj | Add-Member -MemberType AliasProperty -Name 'intAge' -Value age
-SecondValue uint32
PS> $obj | Get-Member

   TypeName: System.Management.Automation.PSCustomObject

Name        MemberType    Definition
----        ----------    ----------
intAge      AliasProperty intAge = (System.UInt32)age
Equals      Method        bool Equals(System.Object obj)
GetHashCode Method        int GetHashCode()
GetType     Method        type GetType()
ToString    Method        string ToString()
Age         NoteProperty  string Age=20
Name        NoteProperty  string Name=Doris

PS> $obj

Name   : Doris
Age    : 20
intAge : 20

PS> $obj.Age + 1
```

```
201

PS> $obj.intAge + 1

21
```

The intAge property is an AliasProperty for the Age property, but the type is guaranteed to be uint32 .

---- Example 7: Add get and set methods to a custom object ----

```
$user = [pscustomobject]@{
    Name      = 'User1'
    Age       = 29
    StartDate = [datetime]'2019-05-05'
    Position  = [pscustomobject]@{
        DepartmentName = 'IT'
        Role = 'Manager'
    }
}
$addMemberSplat = @{
    MemberType = 'ScriptProperty'
    Name = 'Title'
    Value = { $this.Position.Role }                  # getter
    SecondValue = { $this.Position.Role = $args[0] } # setter
}
$user | Add-Member @addMemberSplat
$user | Get-Member

TypeName: System.Management.Automation.PSCustomObject

Name        MemberType     Definition
----        ----------     ----------
Equals      Method         bool Equals(System.Object obj)
```

```
GetHashCode Method         int GetHashCode()
GetType     Method         type GetType()
ToString    Method         string ToString()
Age         NoteProperty   int Age=29
Name        NoteProperty   string Name=User1
Position    NoteProperty   System.Management.Automation.PSCustomObject
Position=@{DepartmentName=IT; Role=Manager}
StartDate   NoteProperty   datetime StartDate=5/5/2019 12:00:00 AM
Title       ScriptProperty System.Object Title {get= $this.Position.Role ;set=
$this.Position.Role = $args[0] ;}

$user.Title = 'Dev Manager'

Name      : User1
Age       : 29
StartDate : 5/5/2019 12:00:00 AM
Position  : @{DepartmentName=IT; Role=Dev Manager}
Title     : Dev Manager
```

Notice that the Title property is a ScriptProperty that has a Get and Set method. When we assign a new value to the Title property, the Set method is called and changes the value of the Role property in the Position property.

REMARKS

```
To see the examples, type: "get-help Add-Member -examples".
For more information, type: "get-help Add-Member -detailed".
For technical information, type: "get-help Add-Member -full".
For online help, type: "get-help Add-Member -online"
```