



**Full credit is given to all the above companies including the Operating System that this PDF file was generated!**

### ***Windows PowerShell Get-Help on Cmdlet 'New-Module'***

**PS:\>Get-HELP New-Module -Full**

#### **NAME**

New-Module

#### **SYNOPSIS**

Creates a new dynamic module that exists only in memory.

#### **SYNTAX**

```
New-Module [-Name] <System.String> [-ScriptBlock] <System.Management.Automation.ScriptBlock> [-ArgumentList  
<System.Object[]>] [-AsCustomObject] [-Cmdlet  
<System.String[]>] [-Function <System.String[]>] [-ReturnResult] [<CommonParameters>]
```

#### **DESCRIPTION**

The `New-Module` cmdlet creates a dynamic module from a script block. The members of the dynamic module, such as functions and variables, are immediately available in the session and remain available until you close the session.

Like static modules, by default, the cmdlets and functions in a dynamic module are exported and the variables and aliases are not. However, you can use the

Export-ModuleMember cmdlet and the parameters of `New-Module` to override the defaults.

You can also use the AsCustomObject parameter of `New-Module` to return the dynamic module as a custom object. The members of the modules, such as functions, are

implemented as script methods of the custom object instead of being imported into the session.

Dynamic modules exist only in memory, not on disk. Like all modules, the members of dynamic modules run in a private module scope that is a child of the global scope.

Get-Module cannot get a dynamic module, but Get-Command can get the exported members.

To make a dynamic module available to `Get-Module`, pipe a `New-Module` command to Import-Module, or pipe the module object that `New-Module` returns to

`Import-Module`. This action adds the dynamic module to the `Get-Module` list, but it does not save the module to disk or make it persistent.

## PARAMETERS

-ArgumentList <System.Object[]>

Specifies an array of arguments which are parameter values that are passed to the script block. For more information about the behavior of ArgumentList , see

about\_Splatting (about/about\_Splatting.md#splatting-with-arrays).

Required? false

Position? named

Default value None

Accept pipeline input? False

Accept wildcard characters? false

-AsCustomObject <System.Management.Automation.SwitchParameter>

Indicates that this cmdlet returns a custom object that represents the dynamic module. The module members are implemented as script methods of the custom object,

but they are not imported into the session. You can save the custom object in a variable and use dot notation to invoke the members.

If the module has multiple members with the same name, such as a function and a variable that are both named A, only one member with each name can be accessed from the custom object.

Required?	false
Position?	named
Default value	False
Accept pipeline input?	False
Accept wildcard characters?	false

#### -Cmdlet <System.String[]>

Specifies an array of cmdlets that this cmdlet exports from the module into the current session. Enter a comma-separated list of cmdlets. Wildcard characters are permitted. By default, all cmdlets in the module are exported.

You cannot define cmdlets in a script block, but a dynamic module can include cmdlets if it imports the cmdlets from a binary module.

Required?	false
Position?	named
Default value	None
Accept pipeline input?	False
Accept wildcard characters?	false

#### -Function <System.String[]>

Specifies an array of functions that this cmdlet exports from the module into the current session. Enter a comma-separated list of functions. Wildcard characters are permitted. By default, all functions defined in a module are exported.

Required?	false
Position?	named
Default value	None

Accept pipeline input? False

Accept wildcard characters? true

#### -Name <System.String>

Specifies a name for the new module. You can also pipe a module name to New-Module.

The default value is an autogenerated name that starts with ` \_DynamicModule ` and is followed by a GUID that specifies the path of the dynamic module.

Required? true

Position? 0

Default value None

Accept pipeline input? True (ByValue)

Accept wildcard characters? false

#### -ReturnResult <System.Management.Automation.SwitchParameter>

Indicates that this cmdlet runs the script block and returns the script block results instead of returning a module object.

Required? false

Position? named

Default value False

Accept pipeline input? False

Accept wildcard characters? false

#### -ScriptBlock <System.Management.Automation.ScriptBlock>

Specifies the contents of the dynamic module. Enclose the contents in braces (`{}`) to create a script block. This parameter is required.

Required? true

Position? 1

Default value None

Accept pipeline input? False

Accept wildcard characters? false

## <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>).

## INPUTS

System.String

You can pipe a module name to this cmdlet.

## OUTPUTS

System.Management.Automation.PSModuleInfo

This cmdlet generates a PSModuleInfo object by default.

System.Management.Automation.PSCustomObject

If you use the AsCustomObject parameter, it generates a PSCustomObject object.

System.Object

If you use the ReturnResult parameter, this cmdlet returns the result of evaluating the script block in the dynamic module.

## NOTES

Windows PowerShell includes the following aliases for `New-Module`:

- `nmo`

You can also refer to `New-Module` by its alias, `nmo`. For more information, see about\_Aliases (About/about\_Aliases.md).

----- Example 1: Create a dynamic module -----

```
New-Module -ScriptBlock {function Hello {"Hello!"}}
```

```
Name      : __DynamicModule_2ceb1d0a-990f-45e4-9fe4-89f0f6ead0e5
Path     : 2ceb1d0a-990f-45e4-9fe4-89f0f6ead0e5
Description   :
Guid      : 00000000-0000-0000-000000000000
Version    : 0.0
ModuleBase   :
ModuleType   : Script
PrivateData   :
AccessMode   : ReadWrite
ExportedAliases : {}
ExportedCmdlets : {}
ExportedFunctions : {[Hello, Hello]}
ExportedVariables : {}
NestedModules   : {}
```

Example 2: Working with dynamic modules and Get-Module and Get-Command

```
new-module -scriptblock {function Hello {"Hello!"}}
```

```
Name      : __DynamicModule_2ceb1d0a-990f-45e4-9fe4-89f0f6ead0e5
Path     : 2ceb1d0a-990f-45e4-9fe4-89f0f6ead0e5
Description   :
Guid      : 00000000-0000-0000-000000000000
Version    : 0.0
ModuleBase   :
ModuleType   : Script
PrivateData   :
```

```
AccessMode      : ReadWrite
ExportedAliases : {}
ExportedCmdlets : {}
ExportedFunctions : {[Hello, Hello]}
ExportedVariables : {}
NestedModules    : {}
```

Get-Module

Get-Command Hello

CommandType	Name	Definition
-------------	------	------------

-----

Function	Hello	"Hello!"
----------	-------	----------

---- Example 3: Export a variable into the current session ----

```
New-Module -ScriptBlock {$SayHelloHelp="Type 'SayHello', a space, and a name."; function SayHello ($name) { "Hello, $name" }; Export-ModuleMember -function SayHello -Variable SayHelloHelp}
$SayHelloHelp
```

Type 'SayHello', a space, and a name.

SayHello Jeffrey

Hello, Jeffrey

The output shows that both the variable and the function were exported into the session.

--- Example 4: Make a dynamic module available to Get-Module ---

```
New-Module -ScriptBlock {function Hello {"Hello!"}} -name GreetingModule | Import-Module
```

```
Get-Module
```

```
Name      : GreetingModule
Path      : d54dfdac-4531-4db2-9dec-0b4b9c57a1e5
Description   :
Guid      : 00000000-0000-0000-0000-000000000000
Version    : 0.0
ModuleBase   :
ModuleType   : Script
PrivateData   :
AccessMode   : ReadWrite
ExportedAliases : {}
ExportedCmdlets : {}
ExportedFunctions : {[Hello, Hello]}
ExportedVariables : {}
NestedModules   : {}
```

```
Get-Command hello
```

CommandType	Name	Definition
Function	Hello	"Hello!"

The `Get-Command` cmdlet shows the `Hello` function that the dynamic module exports.

Example 5: Generate a custom object that has exported functions

```
$m = New-Module -ScriptBlock {

function Hello ($name) {"Hello, $name"}

function Goodbye ($name) {"Goodbye, $name"}

} -AsCustomObject

$m

$m | 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()
----------	--------	-------------------

Goodbye	ScriptMethod	System.Object Goodbye();
---------	--------------	--------------------------

Hello	ScriptMethod	System.Object Hello();
-------	--------------	------------------------

```
$m.goodbye("Jane")
```

Goodbye, Jane

```
$m.hello("Manoj")
```

Hello, Manoj

Piping `\$m` to the `Get-Member` cmdlet displays the properties and methods of the custom object. The output shows that the object has script methods that represent

the `Hello` and `Goodbye` functions. Finally, we call these script methods and display the results.

----- Example 6: Get the results of the script block -----

```
New-Module -ScriptBlock {function SayHello {"Hello, World!"}; SayHello} -ReturnResult
```

Hello, World!

## RELATED LINKS

Online

Version:

gethelp

Export-ModuleMember

Get-Module

Import-Module

Remove-Module

about\_Modules