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

PowerShell Get-Help on command 'Remove-Module'

PS C:\Users\wahid> Get-Help Remove-Module

NAME

Remove-Module

SYNOPSIS

Removes modules from the current session.

SYNTAX

Remove-Module [-FullyQualifiedName]

<Microsoft.PowerShell.Commands.ModuleSpecification[]> [-Force] [-Confirm]

[-WhatIf] [<CommonParameters>]

Remove-Module [-ModuleInfo] <System.Management.Automation.PSModuleInfo[]>

[-Force] [-Confirm] [-WhatIf] [<CommonParameters>]

Remove-Module [-Name] <System.String[]> [-Force] [-Confirm] [-WhatIf]

[<CommonParameters>]

DESCRIPTION

The `Remove-Module` cmdlet removes the members of a module, such as cmdlets

and functions, from the current session.

If the module includes an assembly (`.dll`), all members that are implemented by the assembly are removed, but the assembly isn't unloaded.

This cmdlet doesn't uninstall the module or delete it from the computer. It affects only the current PowerShell session.

PARAMETERS

-Force <System.Management.Automation.SwitchParameter>

Indicates that this cmdlet removes read-only modules. By default, `Remove-Module` removes only read-write modules.

The ReadOnly and ReadWrite values are stored in AccessMode property of a module.

-FullyQualifiedName <Microsoft.PowerShell.Commands.ModuleSpecification[]>

The value can be a module name, a full module specification, or a path to a module file.

When the value is a path, the path can be fully qualified or relative. A relative path is resolved relative to the script that contains the using statement.

When the value is a name or module specification, PowerShell searches the PSModulePath for the specified module.

A module specification is a hashtable that has the following keys.

- `ModuleName` - Required Specifies the module name. - `GUID` - Optional Specifies the GUID of the module. - It's also Required to specify at least one of the three below keys. - `ModuleVersion` - Specifies a minimum

acceptable version of the module. - `MaximumVersion` - Specifies the maximum acceptable version of the module. - `RequiredVersion` - Specifies an exact, required version of the module. This can't be used with the other Version keys.

-ModuleInfo <System.Management.Automation.PSModuleInfo[]>
Specifies the module objects to remove. Enter a variable that contains a PSModuleInfo object or a command that gets a module object, such as a `Get-Module` command. You can also pipe module objects to `Remove-Module`.

-Name <System.String[]>
Specifies the names of modules to remove. Wildcard characters are permitted. You can also pipe name strings to `Remove-Module`.

-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 isn't 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: Remove a module ----------

Remove-Module -Name "BitsTransfer"

This command removes the BitsTransfer module from the current session.

---------- Example 2: Remove all modules ----------

Get-Module | Remove-Module

This command removes all modules from the current session.

------- Example 3: Remove modules by using the pipeline -------

"FileTransfer", "PSDiagnostics" | Remove-Module -Verbose

VERBOSE: Performing operation "Remove-Module" on Target "filetransfer (Path: 'C
:\Windows\system32\WindowsPowerShell\v1.0\Modules\filetransfer\filetransfer.psd
1')".
VERBOSE: Performing operation "Remove-Module" on Target
"Microsoft.BackgroundIntelligentTransfer.Management (Path: 'C:\Windows\assembly
\GAC_MSIL\Microsoft.BackgroundIntelligentTransfer.Management\1.0.0.0__31bf3856a
d364e35\Microsoft.BackgroundIntelligentTransfe
r.Management.dll')".
VERBOSE: Performing operation "Remove-Module" on Target "psdiagnostics (Path: '
C:\Windows\system32\WindowsPowerShell\v1.0\Modules\psdiagnostics\psdiagnostics.
psd1')".
VERBOSE: Removing imported function 'Start-Trace'.
VERBOSE: Removing imported function 'Stop-Trace'.
VERBOSE: Removing imported function 'Enable-WSManTrace'.
VERBOSE: Removing imported function 'Disable-WSManTrace'.
VERBOSE: Removing imported function 'Enable-PSWSManCombinedTrace'.
VERBOSE: Removing imported function 'Disable-PSWSManCombinedTrace'.
VERBOSE: Removing imported function 'Set-LogProperties'.
VERBOSE: Removing imported function 'Get-LogProperties'.
VERBOSE: Removing imported function 'Enable-PSTrace'.
VERBOSE: Removing imported function 'Disable-PSTrace'.
VERBOSE: Performing operation "Remove-Module" on Target "PSDiagnostics (Path: '
C:\Windows\system32\WindowsPowerShell\v1.0\Modules\psdiagnostics\PSDiagnostics.
psm1')".

This command removes the BitsTransfer and PSDiagnostics modules from the

current session.

The command uses a pipeline operator (`|`) to send the module names to `Remove-Module`. It uses the Verbose common parameter to get detailed information about the members that are removed.

The Verbose messages show the items that are removed. The messages differ because the BitsTransfer module includes an assembly that implements its cmdlets and a nested module with its own assembly. The PSDiagnostics module includes a module script file (`.psm1`) that exports functions.

--------- Example 4: Remove a module using ModuleInfo ---------

```
$a = Get-Module BitsTransfer
Remove-Module -ModuleInfo $a
```

This command uses the ModuleInfo parameter to remove the BitsTransfer module.

---------- Example 5: Using the OnRemove event ----------

```
$OnRemoveScript = {
    # perform cleanup
    $cachedSessions | Remove-PSSession
}
$ExecutionContext.SessionState.Module.OnRemove += $OnRemoveScript

$registerEngineEventSplat = @{
    SourceIdentifier = ([System.Management.Automation.PsEngineEvent]::Exiting)
    Action = $OnRemoveScript
}
Register-EngineEvent @registerEngineEventSplat
```

The `$OnRemoveScript` variable contains the script block that cleans up the resources. You register the script block by assigning it to `$ExecutionContext.SessionState.Module.OnRemove`. You can also use

`Register-EngineEvent` to have the script block execute when the PowerShell session ends.

For script-based modules, you would add this code to the `.PSM1` file or put it in a startup script that is listed in the ScriptsToProcess property of the module manifest.

REMARKS

To see the examples, type: "get-help Remove-Module -examples".

For more information, type: "get-help Remove-Module -detailed".

For technical information, type: "get-help Remove-Module -full".

For online help, type: "get-help Remove-Module -online"