

# U01 Windows Administración SO PowerShell

 [Descargar PDF](#)

## ÍNDICE

- [CONSOLA POWERSHELL](#)
- ▼ [CMDLETS](#)
  - [Consulta de cmdlets y funciones](#)
  - ▼ [Módulos](#)
    - [Mostrar módulos](#)
    - [Actualizar módulos](#)
    - [Importar módulos en sesión](#)
  - [Obtener ayuda](#)
  - [Truncar comandos](#)
  - [Tuberías \(|\)](#)
  - [Objetos](#)
  - [Formatear salida](#)
  - ▼ [Convertir salida a otros formatos](#)
    - ▼ [Cadena de texto](#)
      - [Fichero TXT](#)
      - [Fichero CSV](#)
      - [Fichero JSON](#)
      - [Fichero XML](#)
- ▼ [SCRIPTS](#)
  - ▼ [Políticas de ejecución](#)
    - [Tipos](#)
    - [Ámbitos](#)
    - [Administración de las políticas de ejecución](#)

# CONSOLA POWERSHELL

La consola Powershell ( `powershell.exe` ) permite mediante el uso de objetos del sistema operativo consultar y/o modificar cualquier parámetro del sistema, incluidos evidentemente los de red. **Los objetos tienen propiedades que los definen y métodos u operaciones que se pueden realizar sobre los objetos.** También hay objetos que además incluyen a otros objetos o colecciones de ellos.

## CMDLETS

En [Powershell](#), además del uso de órdenes sencillas existen los **cmdlet** y las **funciones**. Desde el punto de vista funcional un cmdlet y una función son similares, por lo que de ahora en adelante nos referiremos a ambos como cmdlet. **Los cmdlet son órdenes o métodos que permiten manipular los objetos y se suelen agrupar en módulos funcionales.** Utiliza el patrón de denominación **verbo-objeto**, donde verbo es el nombre del método a utilizar para manipular el objeto. Son comunes los verbos: Get, Set, New, Add, Remove, Rename, Disable, Enable, Start, Stop ...

Las diferentes opciones y parámetros para la ejecución de un cmdlet se indican mediante el uso de un guion `-` y después un posible valor para dicha opción. Cuando ejecutamos un cmdlet la salida resultante es un objeto o una lista de objetos del sistema, aunque el sistema nos lo muestre como texto por la salida estándar en realidad se tratan de objetos.



### Tip

Usa `TAB` para completar automáticamente nombres de cmdlets o parámetros.

## Consulta de cmdlets y funciones

Se puede obtener información de los comandos instalados en powershell con los cmdlet [Get-Command](#).



### Uso de Get-Command

Obtener todos los cmdlet del objeto que contiene el patrón `-Net`.

```
PS C:\> Get-Command -Name *-Net
```

CommandType	Name	Version	Source
-----	----	-----	-----
Function	Get-Net6to4Configuration	1.0.0.0	NetworkTransition
Function	Get-NetAdapter	2.0.0.0	NetAdapter
Function	Get-NetAdapterAdvancedProperty	2.0.0.0	NetAdapter
Function	Get-NetAdapterBinding	2.0.0.0	NetAdapter
Function	Get-NetAdapterChecksumOffload	2.0.0.0	NetAdapter
Function	Get-NetAdapterEncapsulatedPacketTaskOffload	2.0.0.0	NetAdapter
Function	Get-NetAdapterHardwareInfo	2.0.0.0	NetAdapter
Function	Get-NetAdapterIPsecOffload	2.0.0.0	NetAdapter
Function	Get-NetAdapterLso	2.0.0.0	NetAdapter
Function	Get-NetAdapterPacketDirect	2.0.0.0	NetAdapter

## Uso de Get-Command

Obtener todos los cmdlet que comienzan con el verbo *Format*.

```
PS C:\> Get-Command -Verb Format
```

CommandType	Name	Version	Source
-----	----	-----	-----
Function	Format-Hex	3.1.0.0	Microsoft.PowerShell.Utility
Function	Format-Volume	2.0.0.0	Storage
Cmdlet	Format-Custom	3.1.0.0	Microsoft.PowerShell.Utility
Cmdlet	Format-List	3.1.0.0	Microsoft.PowerShell.Utility
Cmdlet	Format-SecureBootUEFI	2.0.0.0	SecureBoot
Cmdlet	Format-Table	3.1.0.0	Microsoft.PowerShell.Utility
Cmdlet	Format-Wide	3.1.0.0	Microsoft.PowerShell.Utility

## Uso de Get-Command

Obtener todos los cmdlet del módulo de nombre `NetTCPIP`

```
PS C:\> Get-Command -Module NetTCPIP
```

CommandType	Name	Version	Source
-----	----	-----	-----
Function	Get-Net6to4Configuration	1.0.0.0	NetworkTransition
Function	Get-NetAdapter	2.0.0.0	NetAdapter
Function	Get-NetAdapterAdvancedProperty	2.0.0.0	NetAdapter
Function	Get-NetAdapterBinding	2.0.0.0	NetAdapter
Function	Get-NetAdapterChecksumOffload	2.0.0.0	NetAdapter
Function	Get-NetAdapterEncapsulatedPacketTaskOffload	2.0.0.0	NetAdapter
Function	Get-NetAdapterHardwareInfo	2.0.0.0	NetAdapter
Function	Get-NetAdapterIPsecOffload	2.0.0.0	NetAdapter
Function	Get-NetAdapterLso	2.0.0.0	NetAdapter
Function	Get-NetAdapterPacketDirect	2.0.0.0	NetAdapter



## Uso de Get-Command

Obtener todos los cmdlet relacionados con el tratamiento de módulos.

```
PS C:\> Get-Command -Name "*Module*"

ommandType Name                      Version Source
-----
Function    Find-Module                        1.0.0.1 PowerShellGet
Function    Get-InstalledModule              1.0.0.1 PowerShellGet
Function    ImportSystemModules
Function    InModuleScope                   3.4.0    Pester
Function    Install-Module                  1.0.0.1 PowerShellGet
Function    Publish-Module                  1.0.0.1 PowerShellGet
Function    Save-Module                     1.0.0.1 PowerShellGet
Function    Uninstall-Module                1.0.0.1 PowerShellGet
Function    Update-Module                   1.0.0.1 PowerShellGet
Function    Update-ModuleManifest           1.0.0.1 PowerShellGet
Cmdlet      Export-ModuleMember              3.0.0.0 Microsoft.PowerShell.Core
Cmdlet      Get-Module                      3.0.0.0 Microsoft.PowerShell.Core
Cmdlet      Import-Module                   3.0.0.0 Microsoft.PowerShell.Core
Cmdlet      New-Module                      3.0.0.0 Microsoft.PowerShell.Core
Cmdlet      New-ModuleManifest              3.0.0.0 Microsoft.PowerShell.Core
Cmdlet      Remove-Module                   3.0.0.0 Microsoft.PowerShell.Core
Cmdlet      Test-ModuleManifest              3.0.0.0 Microsoft.PowerShell.Core
```

## Módulos

### Mostrar módulos

Para mostrar los módulos instalados usamos el cmdlet [Get-Module](#).



## Uso de Get-Module

Mostrar todos los módulos disponibles en la sesión actual.

```
PS C:\> Get-Module

ModuleType Version      Name                               ExportedCommands
-----
Manifest   3.1.0.0      Microsoft.PowerShell.Management    {Add-Computer, Add-Content, Checkpoint-Computer, .
Manifest   3.1.0.0      Microsoft.PowerShell.Utility       {Add-Member, Add-Type, Clear-Variable,...
Manifest   1.0.0.0      NetTCPIP                           {Find-NetRoute, Get-NetCompartment,...
Script     2.0.0        PSReadLine                         {Get-PSReadLineKeyHandler,...
```



## Uso de Get-Module

Mostrar todos los módulos disponibles en el sistema.

```
PS C:\> Get-Module -ListAvailable
```

No se ha mostrado la salida ya que es bastante extensa.

## Actualizar módulos

Se pueden actualizar los módulos del sistema a partir de repositorios como *Powershell Gallery*. Podemos Buscar, Instalar, Actualizar o Desinstalar módulos del sistema. Para ello se utilizan los cmdlets [Find-Module](#), [Install-Module](#), [Update-Module](#) y [Uninstall-Module](#) respectivamente.

## Importar módulos en sesión

A partir de Powershell 3.0 no es necesario importar módulos instalados a la sesión actual, ya que el sistema lo hace automáticamente al utilizar cualquier cmdlet definido en un módulo. No obstante, se puede realizar manualmente con los cmdlets [Import-Module](#) y [Remove-Module](#) respectivamente.

## Obtener ayuda

Para obtener ayuda sobre el uso y las opciones que se pueden utilizar en un cmdlet se utiliza [Get-Help](#).

### Ayuda en línea

Obtener la ayuda sobre `Get-NetIPAddress`.

```
PS C:\> Get-Help -Name Get-NetIPAddress
```

NOMBRE

`Get-NetIPAddress`

SINOPSIS

Gets the IP address configuration.

SINTAXIS

```
Get-NetIPAddress [-IPAddress] <String[]> [-AddressFamily <AddressFamily[]>] [-AddressState <AddressSt
[-AssociatedIPInterface <CimInstance>] [-CimSession <CimSession[]>] [-IncludeAllCompartments] [-Interfa
[-InterfaceIndex <UInt32[]>] [-PolicyStore <String>] [-PreferredLifetime <TimeSpan>] [-PrefixLength <
[-PrefixOrigin <PrefixOrigin[]>] [-SkipAsSource <Boolean[]>] [-SuffixOrigin <SuffixOrigin[]>] [-Thrott
[-Type <Type[]>] [<CommonParameters>]
```



### Tip

Pulsa `TAB` tras para completar el nombre de un cmdlet, una opción, valores de las opciones, etc ...

Por ejemplo, si escribimos `Get-NetIPAddress -` y pulsamos la tecla `TAB` se nos mostrarán cada una de las posibles opciones.

## Truncar comandos

A veces, un cmdlet puede ser demasiado largo para escribirse en una sola línea. En estos casos, puedes dividirlo en varias líneas utilizando el carácter de escape ``` al final de cada línea y pulsando la tecla `Enter`. Al hacerlo, aparecerá el símbolo `>>` en la línea siguiente, indicando que puedes continuar escribiendo.

### Cmdlet en más de una línea

En el siguiente ejemplo después de escribir el carácter ``` pulsamos la tecla `Enter` (no se escribe el literal `<Enter>`) y nos aparece el literal `>>` en la siguiente línea.

```
PS C:\> Set-DnsClientServerAddress -InterfaceAlias MiRed -ServerAddress `<Enter>
>> 1.1.1.1 -Validate:$false
```

## Tuberías (|)

Se denomina tubería a la posibilidad de encadenar la salida de un cmdlet como entrada de otro. Para realizar tuberías se utiliza el operador pipe `|`.

Es muy común utilizar tuberías para dar formato a la salida, por ejemplo, con `Format-Table`, `Format-List` o `Format-Wide`.

### Tuberías

Obtener los usuarios locales de nombre *admin* y *guest* en formato de lista y mostrar solamente las propiedades `Name` y `Enabled`.

```
PS C:\> Get-LocalUser -Name admin,guest | Format-List -Property Name,Enabled
Name      : admin
Enabled   : True

Name      : Guest
Enabled   : False
```

## Objetos

Como la salida de los cmdlets es también un objeto o una lista de objetos se pueden utilizar el pipe (tubería) y cmdlets adicionales para manipular los objetos. Algunos cmdlets para manipular objetos son: [ForEach-Object](#), [Where-Object](#), [Compare-Object](#), [Group-Object](#), [Select-Object](#), [Sort-Object](#) ...

### Objetos

Obtener todas las interfaces de red, pero solamente el índice y el nombre ordenadas por el número de índice.

```
PS C:\> Get-NetIPInterface | Select-Object -Property IfIndex,InterfaceAlias | Sort-Object ifIndex
ifIndex InterfaceAlias
-----
1 Loopback Pseudo-Interface 1
1 Loopback Pseudo-Interface 1
3 Conexión de área local* 12
3 Conexión de área local* 12
5 MiRed
5 MiRed
14 Wi-Fi
14 Wi
```

También podemos asignar la salida de un cmdlet a una variable.

## Objetos y variables

Asignar a la variable `$i` el objeto obtenido por el cmdlet `Get-NetIPInterface`.

```
PS C:\> $i = Get-NetIPInterface -InterfaceAlias "MiRed" -AddressFamily IPv4
```

Para ver todas las propiedades y métodos de un objeto podemos usar el cmdlet [Get-Member](#). Además podemos ver el valor de una propiedad, modificarlo o invocar un método con la sintaxis **objeto.propiedad** u **objeto.metodo()**.

## Objetos y variables

Mostrar las propiedades y métodos del objeto `$i` del ejemplo anterior.

```
PS C:\> $i | Get-Member
```

La ejecución del comando muestra una gran cantidad de propiedades y métodos.

## Objetos y variables

Mostrar la propiedad `ifIndex` del objeto `$i` del ejemplo anterior.

```
PS C:\> $i.ifIndex
6
```

# Formatear salida

PowerShell tiene un conjunto de cmdlets que permite controlar cómo se muestra la salida de un cmdlet. Estos cmdlet comienzan con el verbo **Format** y además permiten seleccionar las propiedades que se quieren mostrar mediante la opción `-Property`.

Los cmdlets de formateo de la salida son:

Cmdlet	Descripción
<code>Format-Wide</code>	Muestra la salida en formato amplio
<code>Format-List</code>	Muestra la salida en formato lista
<code>Format-Table</code>	Muestra la salida en formato tabla

## Formateo de la salida

Mostrar el nombre de los cmdlets del Module `Win*` en Formato ancho a 3 columnas.

```
PS C:\> Get-Command -Module "Win*" | Format-Wide -Property name -Column 3
Get-WindowsUpdateLog          Get-WUAVersion                Get-WUIPendingReboot
Get-WULastInstallationDate    Get-WULastScanSuccessDate     Install-WUUpdates
Start-WUScan                  Disable-WindowsErrorReporting Enable-WindowsErrorReporting
Get-WindowsErrorReporting
```

## Formateo de la salida

Mostrar la ayuda de `Format-Wide` en Formato Lista indicando solo nombre, el módulo y sinopsis.

```
PS C:\> Get-Help -Name "Format-Wide" | Format-List -Property ,Synopsis
ModuleName : Microsoft.PowerShell.Utility
Synopsis    : Formats objects as a wide table that displays only one property of each object.
```

## Formateo de la salida

Mostrar los usuarios del sistema en Formato *Tabla* indicando solo el nombre y si está o no habilitado.



```
PS C:\> Get-LocalUser | Format-Table -Property name, enabled -AutoSize
Name                Enabled
-----
Administrador        True
DefaultAccount       False
Invitado             False
Profesor             True
WDAGUtilityAccount   False
```

## Convertir salida a otros formatos

Los cmdlet operan sobre objetos y devuelven como salida objetos. A veces, es necesario obtener la salida en otros formatos.

### Cadena de texto

Para producir la salida como una cadena de texto se utilizarán el operador pipe y el cmdlet [Out-String](#). Por ejemplo, un uso común es utilizar la salida de un cmdlet para asignar el valor a una variable (las variables se definen en powershell utilizando el símbolo `$` ).

#### Conversión a cadena de texto

Obtener el nombre de la interfaz de red de nombre *MiRed* en formato cadena y asignarlo a una variable de nombre `$MiRedName` .

```
PS C:\> $MiRedName = (Get-NetAdapter -Name MiRed | Format-Wide -Property ifIndex | Out-String).Trim()
```

El método `Trim()` elimina los espacios a izquierda y derecha de una cadena de texto.

### Fichero TXT

Se utiliza el cmdlet [Out-File](#).

#### Conversión a fichero de texto

Obtener todos los procesos del sistema y guardarlos en un fichero `.txt` .

```
PS C:\> Get-Process | Out-File -FilePath MisProcesos.txt
```

### Fichero CSV

Se utiliza los cmdlets [ConvertTo-CSV](#) y [Out-File](#).



## Conversión a fichero CSV

Obtener todos los procesos del sistema y guardarlos en un fichero `csv` .

```
PS C:\> Get-Process | ConvertTo-CSV -NoTypeInfoation | Out-File -FilePath MisProcesos.csv
```

## Fichero JSON

Se utiliza los cmdlets [ConvertTo-JSON](#) y [Out-File](#).



## Conversión a fichero JSON

Obtener todos los procesos del sistema y guardarlos en un fichero `json` .

```
PS C:\> Get-Process | ConvertTo-Json | Out-File -FilePath MisProcesos.json
```

## Fichero XML

Se utiliza los cmdlets [ConvertTo-XML](#) y [Out-File](#).



## Conversión a fichero XML

Obtener todos los procesos del sistema y guardarlos en un fichero `xml` .

```
PS C:\> Get-Process | ConvertTo-XML -NoTypeInfoation -As String | Out-File -FilePath MisProcesos.xml
```

# SCRIPTS

Los scripts de Powershell son ficheros de texto que contienen una serie de comandos y/o funciones que se ejecutan en el orden en que aparecen. Los scripts pueden contener comentarios, variables, estructuras de control, funciones y cmdlets.

Los ficheros de scripts suelen tener las siguientes extensiones.

Extensión	Descripción
<code>.ps1</code>	Script
<code>.psm1</code>	Script de módulo
<code>.ps1xml</code>	Script de configuración

# Políticas de ejecución

La ejecución de scripts en Powershell está restringida por las [políticas de ejecución](#). Estas políticas de ejecución son una característica de seguridad que controla las condiciones en que Powershell carga ficheros de configuración y ejecuta los scripts.

## Tipos

- **Restricted**
  - No se puede ejecutar ningún script, solamente se permiten órdenes o cmdlets individuales por línea de comandos.
  - Es la política predeterminada para los sistemas Windows de tipo desktop o cliente.
  - Seguridad muy alta.
- **AllSigned**
  - Se pueden ejecutar solamente los scripts y ficheros de configuración locales firmados por un editor de confianza.
  - Seguridad alta.
- **RemoteSigned**
  - Se pueden ejecutar los scripts y ficheros de configuración bajados de internet y firmados por un editor de confianza.
  - Se pueden ejecutar todos los scripts y ficheros de configuración locales, aunque no estén firmados.
  - Seguridad media.
- **UnRestricted**
  - Se puede ejecutar cualquier script y fichero de configuración.
  - Es la política predeterminada para los sistemas no Windows que incluyen Powershell y hoy en día no se puede cambiar.
  - Avisa al usuario antes de ejecutar scripts y ficheros de configuración que no son de la zona de la intranet local.
  - Seguridad baja.
- **Bypass**
  - Se pueden ejecutar cualquier script y fichero de configuración.
  - No hay ningún aviso, petición o permiso para la ejecución.
  - Seguridad muy baja.
- **Default**
  - Establece la política de ejecución predeterminada.
  - `Restricted` para los sistemas Windows de tipo desktop o cliente.
  - `RemoteSigned` para los sistemas Windows de tipo servidor.
  - `UnRestricted` para los sistemas no Windows y no se puede cambiar.
- **Undefined**
  - El administrador no ha configurado ninguna política de seguridad.
  - En este caso se aplican las políticas de tipo `Default`.

## Ámbitos

Las políticas de ejecución se pueden configurar y ejecutar en diferentes **ámbitos** ( `scopes` ). Los ámbitos por orden de preferencia son: *MachinePolicy*, *UserPolicy*, *Process*, *CurrentUser* y *LocalMachine*

## Administración de las políticas de ejecución

Las políticas de ejecución se pueden administrar a través del objeto `ExecutionPolicy` y los cmdlets [Get-ExecutionPolicy](#) y [Set-ExecutionPolicy](#).

### Políticas de ejecución

Obtener la configuración actual de las políticas de ejecución.

```
PS C:\> Get-ExecutionPolicy -List
```

Scope	ExecutionPolicy
MachinePolicy	Undefined
UserPolicy	Undefined
Process	Undefined
CurrentUser	Undefined
LocalMachine	RemoteSigned

### Políticas de ejecución

Obtener configuración actual de políticas de ejecución para el ámbito `CurrentUser` .

```
PS C:\> Get-ExecutionPolicy -Scope CurrentUser
```

### Políticas de ejecución

Configurar `ByPass` para el ámbito `LocalMachine` .

```
PS C:\> Set-ExecutionPolicy -ExecutionPolicy Bypass -Scope LocalMachine
```

También es válido el siguiente cmdlet:

```
PS C:\> Set-ExecutionPolicy -ExecutionPolicy Bypass
```

### Políticas de ejecución

Configurar ByPass para el ámbito `CurrentUser` .

```
PS C:\> Set-ExecutionPolicy -ExecutionPolicy Bypass -Scope CurrentUser
```



## Políticas de ejecución

Borrar la configuración del ámbito `CurrentUser` .

```
PS C:\> Set-ExecutionPolicy -ExecutionPolicy Undefined -Scope CurrentUser
```



## Tip

Recordad, que si la configuración es `Undefined` entonces se aplica la configuración por defecto para el ámbito en función de que el sistema sea un cliente, un servidor o un sistema no Windows.