

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 'Select-Object'

PS:\>Get-HELP Select-Object -Full

NAME

Select-Object

SYNOPSIS

Selects objects or object properties.

SYNTAX

Select-Object [[-Property] <System.Object[]>] [-ExcludeProperty <System.String[]>] [-ExpandProperty <System.String>] [-First <System.Int32>] [-InputObject

<System.Management.Automation.PSObject>] [-Last <System.Int32>] [-Skip <System.Int32>] [-Unique] [-Wait]
[<CommonParameters>]

Select-Object [[-Property] <System.Object[]>] [-ExcludeProperty <System.String[]>] [-ExpandProperty <System.String>] [-InputObject

<System.Management.Automation.PSObject>] [-SkipLast <System.Int32>] [-Unique] [<CommonParameters>]

Select-Object [-Index <System.Int32[]>] [-InputObject <System.Management.Automation.PSObject>] [-Unique] [-Wait] [<CommonParameters>]

DESCRIPTION

The `Select-Object` cmdlet selects specified properties of an object or set of objects. It can also select unique objects, a specified number of objects, or objects

in a specified position in an array.

To select objects from a collection, use the First , Last , Unique , Skip , and Index parameters. To select object properties, use the Property parameter. When you

select properties, `Select-Object` returns new objects that have only the specified properties.

Beginning in Windows PowerShell 3.0, `Select-Object` includes an optimization feature that prevents commands from creating and processing objects that aren't used.

When you use `Select-Object` with the First or Index parameters in a command pipeline, PowerShell stops the command that generates the objects as soon as the selected

number of objects is reached. To turn off this optimizing behavior, use the Wait parameter.

PARAMETERS

-ExcludeProperty <System.String[]>

Specifies the properties that this cmdlet excludes from the operation. Wildcards are permitted. This parameter is effective only when the command also includes

the Property parameter.

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

Accept wildcard characters? true

-ExpandProperty <System.String>

Specifies a property to select, and indicates that an attempt should be made to expand that property. If the input object

- If the specified property is an array, each value of the array is included in the output.

- If the specified property is an object, the objects properties are expanded for every InputObject In either case, the output objects' Type matches the expanded

property's Type .

If the Property parameter is specified, `Select-Object` attempts to add each selected property as a NoteProperty to every outputted object.

> [!WARNING] > If you receive an error that a property can't be processed because a property with that name > already exists, consider the following. Note that

when using ExpandProperty, `Select-Object` > can't replace an existing property. This means: > > - If the expanded object has a property of the same name, the

command returns an error. > - If the Selected object has a property of the same name as an Expanded object's property, the > command returns an error.

Required?	false
Position?	named
Default value	None
Accept pipeline inp	out? False

Accept wildcard characters? false

-First <System.Int32>

Specifies the number of objects to select from the beginning of an array of input objects.

Required?	false
Position?	named
Default value	None
Accept pipeline ir	put? False
Accept wildcard of	characters? false

-Index <System.Int32[]>

Selects objects from an array based on their index values. Enter the indexes in a comma-separated list. Indexes in an array begin with 0, where 0 represents the

first value and (n-1) represents the last value.

Required?	false
Position?	named
Default value	None
Accept pipeline in	iput? False
Accept wildcard c	haracters? false

-InputObject <System.Management.Automation.PSObject>

Specifies objects to send to the cmdlet through the pipeline. This parameter enables you to pipe objects to `Select-Object`.

When you pass objects to the InputObject parameter, instead of using the pipeline, `Select-Object` treats the InputObject as a single object, even if the value is

a collection. It is recommended that you use the pipeline when passing collections to `Select-Object`.

Required?	false
Position?	named
Default value	None
Accept pipeline input	? True (ByValue)
Accept wildcard chara	acters? false

-Last <System.Int32>

Specifies the number of objects to select from the end of an array of input objects.

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

-Property <System.Object[]>

Specifies the properties to select. These properties are added as NoteProperty members to the output objects. Wildcards are permitted. If the input object doesn't

have the property named, the value of the new NoteProperty is set to `\$null`.

The value of the Property parameter can be a new calculated property. To create a calculated, property, use a hash table.

Valid keys are:

- Name (or Label) - `<string>`

- Expression - `<string>` or `<script block>`

	For	more	information,	see	about_Calculated_Properties
/ /Miercooft DowerChall Core/About/about C		Droportio	a md)		

(../Microsoft.PowerShell.Core/About/about_Calculated_Properties.md).

Required? false

Position? 0

Default value None

Accept pipeline input? False

Accept wildcard characters? true

-Skip <System.Int32>

Skips (doesn't select) the specified number of items. By default, the Skip parameter counts from the beginning of the collection of objects. If the command uses

the Last parameter, it counts from the end of the collection.

Unlike the Index parameter, which starts counting at 0, the Skip parameter begins at 1.

Required?	false
Position?	named
Default value	None
Accept pipeline ir	put? False
Accept wildcard of	characters? false

-SkipLast <System.Int32>

Skips (doesn't select) the specified number of items from the end of the list or array. Works in the same way as using Skip together with Last parameter.

Unlike the Index parameter, which starts counting at 0, the SkipLast parameter begins at 1.

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

-Unique <System.Management.Automation.SwitchParameter>

Specifies that if a subset of the input objects has identical properties and values, only a single member of the subset should be selected. Unique selects values

after other filtering parameters are applied.

This parameter is case-sensitive. As a result, strings that differ only in character casing are considered to be unique.

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

-Wait <System.Management.Automation.SwitchParameter>

Indicates that the cmdlet turns off optimization. PowerShell runs commands in the order that they appear for the

command pipeline and lets them generate all

objects. By default, if you include a `Select-Object` command with the First or Index parameters in a command

pipeline, PowerShell stops the command that

generates the objects as soon as the selected number of objects is generated.

This parameter was introduced in Windows PowerShell 3.0.

Required?	false
Position?	named
Default value	False
Accept pipeline in	nput? 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.Management.Automation.PSObject

You can pipe objects to this cmdlet.

OUTPUTS

System.Management.Automation.PSObject

This cmdlet returns the input objects with only the selected properties.

NOTES

- `select`

The optimization feature of `Select-Object` is available only for commands that write objects to the pipeline as they're processed. It has no effect on commands

that buffer processed objects and write them as a collection. Writing objects immediately is a cmdlet design best practice. For more information, see Write Single

Records to the Pipeline in Strongly Encouraged Development Guidelines (/powershell/scripting/developer/windows-powershell).

----- Example 1: Select objects by property ------

Get-Process | Select-Object - Property ProcessName, Id, WS

- Example 2: Select objects by property and format the results -

Get-Process Explorer |

Select-Object - Property ProcessName - ExpandProperty Modules |

Format-List

- ProcessName : explorer
- ModuleName : explorer.exe
- FileName : C:\WINDOWS\explorer.exe
- BaseAddress : 140697278152704

ModuleMemorySize : 3919872

EntryPointAddress: 140697278841168

FileVersionInfo : File: C:\WINDOWS\explorer.exe

InternalName: explorer

OriginalFilename: EXPLORER.EXE.MUI

FileVersion: 10.0.17134.1 (WinBuild.160101.0800)

FileDescription: Windows Explorer

Product: Microsoft Windows Operating System

----- Example 3: Select processes using the most memory -----

Get-Process | Sort-Object -Property WS | Select-Object -Last 5

Handle	es NPN	И(K) РМ((K) WS(K) \	/S(M) (CPU(s)	Id ProcessNam	e
2866	320	33432	45764 203	222.41	 I 1292 s	svchost	
577	17	23676	50516 265	50.58	4388 W	INWORD	
826	11	75448	76712 188	19.77	3780 Ps		
1367	14	73152	88736 216	61.69	676 Ps		
1612	44	66080	92780 380	900.59	6132 II	NFOPATH	

----- Example 4: Select unique characters from an array ------

"a","b","c","a","a","a" | Select-Object -Unique

а

...

b

С

------ Example 5: Using `-Unique` with other parameters ------

"a","a","b","c" | Select-Object -First 2 -Unique

а

unique value.

- Example 6: Select newest and oldest events in the event log -

\$a = Get-EventLog -LogName "Windows PowerShell"

\$a | Select-Object -Index 0, (\$A.count - 1)

----- Example 7: Select all but the first object ------

New-PSSession -ComputerName (Get-Content Servers.txt | Select-Object -Skip 1)

----- Example 8: Rename files and select several to review -----

Get-ChildItem *.txt -ReadOnly |

Rename-Item -NewName {\$_.BaseName + "-ro.txt"} -PassThru |

Select-Object -First 5 -Wait

Example 9: Show the intricacies of the -ExpandProperty parameter

Create a custom object to use for the Select-Object example.

\$object = [pscustomobject]@{Name="CustomObject";Expand=@(1,2,3,4,5)}

Use the ExpandProperty parameter to Expand the property.

\$object | Select-Object -ExpandProperty Expand -Property Name

1

·

2

3

4

5

Use Get-Member to confirm the Name property was added and populated.\$object | Select-Object -ExpandProperty Expand -Property Name | Get-Member

TypeName: System.Int32

Name MemberType Definition
CompareTo Method int CompareTo(System.Object value), int CompareTo(int value),
Equals Method bool Equals(System.Object obj), bool Equals(int obj), bool IEq
GetHashCode Method int GetHashCode()
GetType Method type GetType()
GetTypeCode Method System.TypeCode GetTypeCode(), System.TypeCode IConvertible.Ge
ToBoolean Method bool IConvertible.ToBoolean(System.IFormatProvider provider)
ToByte Method byte IConvertible.ToByte(System.IFormatProvider provider)
ToChar Method char IConvertible.ToChar(System.IFormatProvider provider)
ToDateTime Method datetime IConvertible.ToDateTime(System.IFormatProvider provider)
ToDecimal Method decimal IConvertible.ToDecimal(System.IFormatProvider provider)
ToDouble Method double IConvertible.ToDouble(System.IFormatProvider provider)
ToInt16 Method int16 IConvertible.ToInt16(System.IFormatProvider provider)
ToInt32 Method int IConvertible.ToInt32(System.IFormatProvider provider)
ToInt64 Method long IConvertible.ToInt64(System.IFormatProvider provider)
ToSByte Method sbyte IConvertible.ToSByte(System.IFormatProvider provider)
ToSingle Method float IConvertible.ToSingle(System.IFormatProvider provider)
ToString Method string ToString(), string ToString(string format), string ToS
ToType Method System.Object IConvertible.ToType(type conversionType, System
ToUInt16 Method uint16 IConvertible.ToUInt16(System.IFormatProvider provider)
ToUInt32 Method uint32 IConvertible.ToUInt32(System.IFormatProvider provider)
ToUInt64 Method uint64 IConvertible.ToUInt64(System.IFormatProvider provider)
Name NoteProperty string Name=CustomObject

------ Example 10: Create custom properties on objects ------

\$customObject = 1 | Select-Object -Property MyCustomProperty \$customObject.MyCustomProperty = "New Custom Property" \$customObject

MyCustomProperty

New Custom Property

Example 11: Create calculated properties for each InputObject

Create a calculated property called \$_.StartTime.DayOfWeek

Get-Process | Select-Object -Property ProcessName,{\$_.StartTime.DayOfWeek}

ProcessName \$_.StartTime.DayOfWeek

alg Wednesday

ati2evxx Wednesday

ati2evxx Thursday

•••

Add a custom property to calculate the size in KiloBytes of each FileInfo

object you pass in. Use the pipeline variable to divide each file's length by

1 KiloBytes

\$size = @{label="Size(KB)";expression={\$_.length/1KB}}

Create an additional calculated property with the number of Days since the

file was last accessed. You can also shorten the key names to be 'l', and 'e',

or use Name instead of Label.

\$days = @{l="Days";e={((Get-Date) - \$_.LastAccessTime).Days}}

You can also shorten the name of your label key to 'l' and your expression key

to 'e'.

Get-ChildItem \$PSHOME -File | Select-Object Name, \$size, \$days

Name	Size(KE	3)	Days		
Certificate.format.ps	1xml 12	2.5244	140625	223	3
Diagnostics.Format.	ps1xml	4.955	078125	22	3
DotNetTypes.format	.ps1xml	134.9	98339843	75	223

Example 12: Selecting the keys of a hashtable with calculated properties

```
@{ name = 'a' ; weight = 7 } | Select-Object -Property @(
  @{ label = 'Name' ; expression = { $_.name } }
  @{ label = 'Weight' ; expression = { $_.weight } }
)
```

Name Weight

a 7

RELATED LINKS

Online

Version:

https://learn.microsoft.com/powershell/module/microsoft.powershell.utility/select-object?view=powershell-5.1&WT.mc_id=ps-

gethelp

about_Calculated_Properties

Group-Object

Sort-Object

Where-Object