



## ***Windows PowerShell Get-Help on Cmdlet 'Format-Table'***

***PS:\>Get-HELP Format-Table -Full***

### **NAME**

Format-Table

### **SYNOPSIS**

Formats the output as a table.

### **SYNTAX**

```
Format-Table [[-Property] <System.Object[]> [-AutoSize] [-DisplayError] [-Expand {CoreOnly | EnumOnly | Both}] [-Force]
[-GroupBy <System.Object>]
[-HideTableHeaders] [-InputObject <System.Management.Automation.PSObject>] [-RepeatHeader] [-ShowError] [-View
<System.String>] [-Wrap] [<CommonParameters>]
```

### **DESCRIPTION**

The `Format-Table` cmdlet formats the output of a command as a table with the selected properties of the object in each column. The object type determines the default

layout and properties that are displayed in each column. You can use the Property parameter to select the properties that you want to display.

PowerShell uses default formatters to define how object types are displayed. You can use `.ps1xml` files to create custom views that display an output table with

specified properties. After a custom view is created, use the `View` parameter to display the table with your custom view. For more information about views, see `about_Format.ps1xml` (`../Microsoft.PowerShell.Core/About/about_Format.ps1xml.md`).

You can use a hash table to add calculated properties to an object before displaying it and to specify the column headings in the table. To add a calculated property,

use the `Property` or `GroupBy` parameter. For more information about hash tables, see `about_Hash_Tables` (`../Microsoft.PowerShell.Core/About/about_Hash_Tables.md`).

## PARAMETERS

`-AutoSize` <System.Management.Automation.SwitchParameter>

Indicates that the cmdlet adjusts the column size and number of columns based on the width of the data. By default, the column size and number are determined by the view.

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

`-DisplayError` <System.Management.Automation.SwitchParameter>

Indicates that the cmdlet displays errors on the command line. This parameter can be used as a debugging aid when you're formatting expressions in a ``Format-Table`` command and need to troubleshoot the expressions.

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

Accept wildcard characters? false

#### -Expand <System.String>

Specifies the format of the collection object and the objects in the collection. This parameter is designed to format objects that support the ICollection

([/dotnet/api/system.collections.ICollection](#))([System.Collections](#) ([/dotnet/api/system.collections](#)))interface. The default value is EnumOnly . The acceptable values

for this parameter are as follows:

- EnumOnly : Displays the properties of the objects in the collection. - CoreOnly : Displays the properties of the collection object. - Both : Displays the properties of the collection object and the properties of objects in the collection.

Required? false

Position? named

Default value None

Accept pipeline input? False

Accept wildcard characters? false

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

Indicates that the cmdlet directs the cmdlet to display all the error information. Use with the DisplayError or ShowError parameter. By default, when an error

object is written to the error or display streams, only some error information is displayed.

Also required when formatting certain .NET types. For more information, see the Notes (#notes)section.

Required? false

Position? named

Default value False

Accept pipeline input? False

Accept wildcard characters? false

#### -GroupBy <System.Object>

Specifies sorted output in separate tables based on a property value. For example, you can use GroupBy to list services in separate tables based on their status.

Enter an expression or a property. The GroupBy parameter expects that the objects are sorted. Use the `Sort-Object` cmdlet before using `Format-Table` to group the objects.

The value of the GroupBy parameter can be a new calculated property. The calculated property can be a script block or a hash table. Valid key-value pairs are:

- Name (or Label) - ``
- Expression - `` or `

Accept pipeline input? False

Accept wildcard characters? false

-InputObject <System.Management.Automation.PSObject>

Specifies the objects to format. Enter a variable that contains the objects, or type a command or expression that gets the objects.

Required? false

Position? named

Default value None

Accept pipeline input? True (ByValue)

Accept wildcard characters? false

-Property <System.Object[]>

Specifies the object properties that appear in the display and the order in which they appear. Type one or more property names, separated by commas, or use a hash

table to display a calculated property. Wildcards are permitted.

If you omit this parameter, the properties that appear in the display depend on the first object's properties. For example, if the first object has PropertyA and

PropertyB but subsequent objects have PropertyA , PropertyB , and PropertyC , then only the PropertyA and PropertyB headers are displayed.

The Property parameter is optional. You can't use the Property and View parameters in the same command.

The value of the Property parameter can be a new calculated property. The calculated property can be a script block or a hash table. Valid key-value pairs are:

- Name (or Label) `<string>`

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

- FormatString - `<string>`

- Width - `<int32>` - must be greater than `0`

- Alignment - value can be `Left`, `Center`, or `Right`

For more information, see [about\\_Calculated\\_Properties](#)  
(../Microsoft.PowerShell.Core/About/about\_Calculated\_Properties.md).

Required?	false
Position?	0
Default value	None
Accept pipeline input?	False
Accept wildcard characters?	true

**-RepeatHeader** `<System.Management.Automation.SwitchParameter>`

Repeats displaying the header of a table after every screen full. The repeated header is useful when the output is piped to a pager such as `less` or `more` or paging with a screen reader.

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

**-ShowError** `<System.Management.Automation.SwitchParameter>`

This parameter sends errors through the pipeline. This parameter can be used as a debugging aid when you're formatting expressions in a `Format-Table` command and need to troubleshoot the expressions.

Required?	false
-----------	-------

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

-View <System.String>

In PowerShell 5.1 and earlier versions, the default views are defined in `*.format.ps1xml`` files stored in `$PSHOME``.

The View parameter lets you specify an alternate format or custom view for the table. You can use the default PowerShell views or create custom views. For more information about how to create a custom view, see `about_Format.ps1xml` (`../Microsoft.PowerShell.Core/About/about_Format.ps1xml.md`).

The alternate and custom views for the View parameter must use the table format, otherwise, ``Format-Table`` fails. If the alternate view is a list, use the

``Format-List`` cmdlet. If the alternate view isn't a list or a table, use the ``Format-Custom`` cmdlet.

You can't use the Property and View parameters in the same command.

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

-Wrap <System.Management.Automation.SwitchParameter>

Displays text that exceeds the column width on the next line. By default, text that exceeds the column width is truncated.

Required?                false  
Position?                named  
Default value            False  
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.Management.Automation.PSObject

You can pipe any object to this cmdlet.

#### OUTPUTS

Microsoft.PowerShell.Commands.Internal.Format

This cmdlet returns format objects that represent the table.

#### NOTES

Windows PowerShell includes the following aliases for ``Format-Table``:

- ``ft``

If you want to use ``Format-Table`` with the Property parameter, you need to include the Force parameter under any of the following conditions:

- The input objects are normally formatted out-of-band using the ``ToString()`` method. This applies to ``[string]`` and .NET primitive types, which are a superset of the built-in numeric types such as ``[int]``, ``[long]``, and others.
- The input objects have no public properties.



- The input objects are instances of the wrapper types PowerShell uses for output streams other than the Success output stream. This applies only when these

wrapper types are sent to the Success output stream that requires either having captured them via common parameters such as ErrorVariable first or using a

redirection such as `\*>&1`.

- The wrapper types include:

- System.Management.Automation.ErrorRecord (/dotnet/api/System.Management.Automation.ErrorRecord)-  
System.Management.Automation.WarningRecord

(/dotnet/api/System.Management.Automation.WarningRecord)- System.Management.Automation.VerboseRecord  
(/dotnet/api/System.Management.Automation.VerboseRecord)-

System.Management.Automation.DebugRecord (/dotnet/api/System.Management.Automation.DebugRecord)-  
System.Management.Automation.InformationRecord

(/dotnet/api/System.Management.Automation.InformationRecord)

----- Example 1: Format PowerShell host -----

Get-Host | Format-Table -AutoSize

The `Get-Host` cmdlet gets System.Management.Automation.Internal.Host.InternalHost objects that represent the host. The objects are sent down the pipeline to

`Format-Table` and displayed in a table. The AutoSize parameter adjusts the column widths to minimize truncation.

----- Example 2: Format processes by BasePriority -----

Get-Process | Sort-Object -Property BasePriority | Format-Table -GroupBy BasePriority -Wrap

The `Get-Process` cmdlet gets objects that represent each process on the computer and sends them down the pipeline to `Sort-Object`. The objects are sorted in the order of their BasePriority property.

The sorted objects are sent down the pipeline to `Format-Table`. The GroupBy parameter arranges the process data into

groups based on their BasePriority property's

value. The Wrap parameter ensures that data isn't truncated.

----- Example 3: Format processes by start date -----

```
Get-Process | Sort-Object StartTime | Format-Table -View StartTime
```

`Get-Process` gets the System.Diagnostics.Process objects that represent the processes running on the computer. The objects are sent down the pipeline to

`Sort-Object`, and are sorted based on the StartTime property.

The sorted objects are sent down the pipeline to `Format-Table`. The View parameter specifies the StartTime view that's defined in the PowerShell

`DotNetTypes.format.ps1xml` file for System.Diagnostics.Process objects. The StartTime view converts each processes start time to a short date and then groups the processes by the start date.

The `DotNetTypes.format.ps1xml` file contains a Priority view for processes. You can create your own `format.ps1xml` files with customized views.

----- Example 4: Use a custom view for table output -----

```
Get-ChildItem -Path C:\Test | Format-Table -View mygciview
```

Directory: C:\Test

Mode	LastWriteTime		CreationTime		Length Name
----	-----		-----		-----
d----	11/4/2019	15:54	9/24/2019	15:54	Archives
d----	8/27/2019	14:22	8/27/2019	14:22	Drawings
d----	10/23/2019	09:38	2/25/2019	09:38	Files
-a----	11/7/2019	11:07	11/7/2019	11:07	11345 Alias.txt
-a----	2/27/2019	15:15	2/27/2019	15:15	258 alias_out.txt
-a----	2/27/2019	15:16	2/27/2019	15:16	258 alias_out2.txt

`Get-ChildItem` gets the contents of the current directory, `C:\Test`. The `System.IO.DirectoryInfo` and `System.IO.FileInfo` objects are sent down the pipeline.

`Format-Table` uses the `View` parameter to specify the custom view `mygciview` that includes the `CreationTime` column.

The default `Format-Table` output for `Get-ChildItem` doesn't include the `CreationTime` column.

----- Example 5: Use properties for table output -----

```
Get-Service | Format-Table -Property Name, DependentServices
```

`Get-Service` gets all the services on the computer and sends the `System.ServiceProcess.ServiceController` objects down the pipeline. `Format-Table` uses the `Property`

parameter to specify that the `Name` and `DependentServices` properties are displayed in the table. `Name` and `DependentServices` are two of the object type's properties. To

view all the properties: `Get-Service | Get-Member -MemberType Properties`.

-- Example 6: Format a process and calculate its running time --

```
Get-Process notepad |
```

```
Format-Table ProcessName, @{Label="TotalRunningTime"; Expression={{(Get-Date) - $_.StartTime}}
```

```
ProcessName TotalRunningTime
```

```
-----
```

```
notepad    03:20:00.2751767
```

```
notepad    00:00:16.7710520
```

`Get-Process` gets all the local computer's notepad processes and sends the objects down the pipeline. `Format-Table` displays a table with two columns: `ProcessName`,

a `Get-Process` property, and `TotalRunningTime`, a calculated property.

The `TotalRunningTime` property is specified by a hash table with two keys, `Label` and `Expression`. The `Label` key specifies the property name. The `Expression` key

specifies the calculation. The expression gets the `StartTime` property of each process object and subtracts it from the result of a `Get-Date` command, which gets the

current date and time.

----- Example 7: Format Notepad processes -----

```
$Processes = Get-CimInstance -Class win32_process -Filter "name='notepad.exe'"  
$Processes | Format-Table ProcessName, @{  
    Label = "Total Running Time"  
    Expression={(Get-Date) - $_.CreationDate}  
}
```

```
ProcessName Total Running Time
```

```
-----
```

```
notepad.exe 03:39:39.6260693
```

```
notepad.exe 00:19:56.1376922
```

`Get-CimInstance` gets instances of the WMI Win32\_Process class that describes all the local computer's processes named notepad.exe . The process objects are stored in the `\$Processes` variable.

The process objects in the `\$Processes` variable are sent down the pipeline to `Format-Table`, which displays the ProcessName property and a new calculated property, Total Running Time .

The command assigns the name of the new calculated property, Total Running Time , to the Label key. The Expression key's script block calculates how long the process

has been running by subtracting the processes creation date from the current date. The `Get-Date` cmdlet gets the current date. The creation date is subtracted from

the current date. The result is the value of Total Running Time .

----- Example 8: Troubleshooting format errors -----

```
Get-Date | Format-Table DayOfWeek,{ $_ / $null } -DisplayError
```

```
DayOfWeek $_ / $null
```

```
-----
```

```
Wednesday #ERR
```

Get-Date | Format-Table DayOfWeek,{ \$\_ / \$null } -ShowError

DayOfWeek \$\_ / \$null

-----

Wednesday

Failed to evaluate expression " \$\_ / \$null ".

- + CategoryInfo : InvalidArgument: (11/27/2019 12:53:41:PSObject) [], RuntimeException
- + FullyQualifiedErrorId : mshExpressionError

## RELATED LINKS

Online

Version:

[https://learn.microsoft.com/powershell/module/microsoft.powershell.utility/format-table?view=powershell-5.1&WT.mc\\_id=ps-gethelp](https://learn.microsoft.com/powershell/module/microsoft.powershell.utility/format-table?view=powershell-5.1&WT.mc_id=ps-gethelp)

about\_Calculated\_Properties

about\_Format.ps1xml

about\_Hash\_Tables

Export-FormatData

Format-Custom

Format-Hex

Format-List

Format-Wide

Get-FormatData

Get-Member

Get-CimInstance

Update-FormatData