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

PowerShell Get-Help on command 'Compare-Object'

PS C:\Users\wahid> Get-Help Compare-Object

NAME

Compare-Object

SYNOPSIS

Compares two sets of objects.

SYNTAX

Compare-Object [-ReferenceObject] <System.Management.Automation.PSObject[]>
[-DifferenceObject] <System.Management.Automation.PSObject[]> [-CaseSensitive]
[-Culture <System.String>] [-ExcludeDifferent] [-IncludeEqual] [-PassThru]
[-Property <System.Object[]>] [-SyncWindow <System.Int32>] [<CommonParameters>]

DESCRIPTION

The `Compare-Object` cmdlet compares two sets of objects. One set of objects is the reference , and the other set of objects is the difference .

`Compare-Object` checks for available methods of comparing a whole object. If it can't find a suitable method, it calls the ToString() methods of the input objects and compares the string results. You can provide one or more

properties to be used for comparison. When properties are provided, the cmdlet compares the values of those properties only.

The result of the comparison indicates whether a property value appeared only in the reference object (`<=`) or only in the difference object (`=>`). If the IncludeEqual parameter is used, (`==`) indicates the value is in both objects.

If the reference or the difference objects are null (`$null`), `Compare-Object` generates a terminating error.

Some examples use splatting to reduce the line length of the code samples. For more information, see about_Splatting (../Microsoft.PowerShell.Core/About/about_Splatting.md).

PARAMETERS

-CaseSensitive <System.Management.Automation.SwitchParameter>
Indicates that comparisons should be case-sensitive.

-Culture <System.String>
Specifies the culture to use for comparisons.

-DifferenceObject <System.Management.Automation.PSObject[]>
Specifies the objects that are compared to the reference objects.

-ExcludeDifferent <System.Management.Automation.SwitchParameter>
Indicates that this cmdlet displays only the characteristics of compared objects that are equal. The differences between the objects are discarded.

Use ExcludeDifferent with IncludeEqual to display only the lines that match between the reference and difference objects.

If ExcludeDifferent is specified without IncludeEqual , there's no output.

-IncludeEqual <System.Management.Automation.SwitchParameter>
    IncludeEqual displays the matches between the reference and difference
    objects.

    By default, the output also includes the differences between the reference
    and difference objects.

-PassThru <System.Management.Automation.SwitchParameter>
    When you use the PassThru parameter, `Compare-Object` omits the
    PSCustomObject wrapper around the compared objects and returns the
    differing objects, unchanged.

-Property <System.Object[]>
    Specifies an array of properties of the reference and difference objects
    to compare.

    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:

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

    For more information, see about_Calculated_Properties
    (../Microsoft.PowerShell.Core/About/about_Calculated_Properties.md).

-ReferenceObject <System.Management.Automation.PSObject[]>
    Specifies an array of objects used as a reference for comparison.

-SyncWindow <System.Int32>
    Specifies the number of adjacent objects that `Compare-Object` inspects
    while looking for a match in a collection of objects. `Compare-Object`
    examines adjacent objects when it doesn't find the object in the same

position in a collection. The default value is `[Int32]::MaxValue`, which means that `Compare-Object` examines the entire object collection.

When working with large collections, the default value might not be efficient but is accurate. Specifying a smaller value for SyncWindow can increase performance but could have lower accuracy.

<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 - Compare the content of two text files ------

```
Compare-Object -ReferenceObject (Get-Content -Path C:\Test\Testfile1.txt)
-DifferenceObject (Get-Content -Path C:\Test\Testfile2.txt)

InputObject SideIndicator
----------- -------------
cat         =>
racoon      =>
dog         <=
squirrel    <=
```

Example 2 - Compare each line of content and exclude the differences

```
$objects = @{
  ReferenceObject = (Get-Content -Path C:\Test\Testfile1.txt)
  DifferenceObject = (Get-Content -Path C:\Test\Testfile2.txt)
}
Compare-Object @objects -IncludeEqual -ExcludeDifferent
```

```
InputObject SideIndicator
----------- -------------
bird        ==
```

<a id="ex3" />

Example 3 - Show the difference when using the PassThru parameter

```
$a = $True
Compare-Object -IncludeEqual $a $a
(Compare-Object -IncludeEqual $a $a) | Get-Member

InputObject SideIndicator
----------- -------------
       True ==


   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()
InputObject   NoteProperty System.Boolean InputObject=True
SideIndicator NoteProperty string SideIndicator===

Compare-Object -IncludeEqual $a $a -PassThru
(Compare-Object -IncludeEqual $a $a -PassThru) | Get-Member

True


   TypeName: System.Boolean
Name          MemberType   Definition
```

---- ---------- ----------

CompareTo Method int CompareTo(System.Object obj), int

CompareTo(bool value), int IComparable.CompareTo(Syst

Equals Method bool Equals(System.Object obj), bool Equals(bool

obj), bool IEquatable[bool].Equals(bool ot

GetHashCode Method int GetHashCode()

GetType Method type GetType()

GetTypeCode Method System.TypeCode GetTypeCode(), System.TypeCode

IConvertible.GetTypeCode()

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 short 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(System.IFormatProvider provider), string IConvertible.To

ToType Method System.Object IConvertible.ToType(type conversionType, System.IFormatProvider provider)
ToUInt16 Method ushort IConvertible.ToUInt16(System.IFormatProvider provider)
ToUInt32 Method uint IConvertible.ToUInt32(System.IFormatProvider provider)
ToUInt64 Method ulong IConvertible.ToUInt64(System.IFormatProvider provider)
TryFormat Method bool TryFormat(System.Span[char] destination, [ref] int charsWritten)
SideIndicator NoteProperty string SideIndicator===

When using PassThru , the original object type (System.Boolean) is returned. Note how the output displayed by the default format for System.Boolean objects didn't display the SideIndicator property. However, the returned System.Boolean object has the added NoteProperty .
--- Example 4 - Compare two simple objects using properties ---

Compare-Object -ReferenceObject 'abc' -DifferenceObject 'xyz' -Property Length -IncludeEqual

Length SideIndicator
------ -------------
3 ==

---- Example 5 - Comparing complex objects using properties ----

PS> Get-Process pwsh

NPM(K) PM(M) WS(M) CPU(s) Id SI ProcessName
------ ----- ----- ------ -- -- -----------
101 123.32 139.10 35.81 11168 1 pwsh

```
    89   107.55      66.97      11.44   17600   1 pwsh

PS> $a = Get-Process -Id 11168
PS> $b = Get-Process -Id 17600
PS> $a.ToString()
System.Diagnostics.Process (pwsh)
PS> $b.ToString()
System.Diagnostics.Process (pwsh)
PS> Compare-Object $a $b -IncludeEqual

InputObject                    SideIndicator
-----------                    -------------
System.Diagnostics.Process (pwsh) ==

PS> Compare-Object $a $b -Property ProcessName, Id, CPU

ProcessName    Id       CPU SideIndicator
-----------    --       --- -------------
pwsh        17600   11.4375 =>
pwsh        11168 36.203125 <=
```

When you specify properties to be compared, the cmdlet shows the differences.

Example 6 - Comparing complex objects that implement IComparable

```
Compare-Object ([TimeSpan]"0:0:1") "0:0:1" -IncludeEqual

InputObject SideIndicator
----------- -------------
00:00:01    ==

Compare-Object "0:0:1" ([TimeSpan]"0:0:1")

InputObject SideIndicator
```

---------- ----------

00:00:01 =>

0:0:1 <=

In the second case, the TimeSpan is converted to a string so the object are different.

REMARKS

To see the examples, type: "get-help Compare-Object -examples".

For more information, type: "get-help Compare-Object -detailed".

For technical information, type: "get-help Compare-Object -full".

For online help, type: "get-help Compare-Object -online"