Join-String (Microsoft.PowerShell.Utility) - PowerShell

sdwheeler ⋮ 11-14 minutes

Join-String

- Reference

In this article

Combines objects from the pipeline into a single string.

Syntax

```
Join-String
    [[-Property] <PSPropertyExpression>]
    [[-Separator] <String>]
    [-OutputPrefix <String>]
    [-OutputSuffix <String>]
    [-UseCulture]
    [-InputObject <PSObject[]>]
    [<CommonParameters>]
```

```
Join-String
    [[-Property] <PSPropertyExpression>]
    [[-Separator] <String>]
    [-OutputPrefix <String>]
    [-OutputSuffix <String>]
    [-SingleQuote]
```

```
    [-UseCulture]
    [-InputObject <PSObject[]>]
    [<CommonParameters>]
```

```
Join-String
    [[-Property] <PSPropertyExpression>]
    [[-Separator] <String>]
    [-OutputPrefix <String>]
    [-OutputSuffix <String>]
    [-DoubleQuote]
    [-UseCulture]
    [-InputObject <PSObject[]>]
    [<CommonParameters>]
```

```
Join-String
    [[-Property] <PSPropertyExpression>]
    [[-Separator] <String>]
    [-OutputPrefix <String>]
    [-OutputSuffix <String>]
    [-FormatString <String>]
    [-UseCulture]
    [-InputObject <PSObject[]>]
    [<CommonParameters>]
```

Description

The `Join-String` cmdlet joins, or combines, text from pipeline objects into a single string.

If no parameters are specified, the pipeline objects are converted to a string and joined with the default separator `$OFS`.

Note

When you set `$OFS` its value is used to join arrays when they're converted to strings until the variable is reset to `$null`. Because using `$OFS` can have unintended effects elsewhere in your code, it's best to use the **Separator** parameter instead.

By specifying a property name, the property's value is converted to a string and joined into a string.

Instead of a property name, a script block can be used. The script block's result is converted to a string before it's joined to form the result. It can either combine the text of an object's property or the result of the object that was converted to a string.

This cmdlet was introduced in PowerShell 6.2.

Examples

Example 1: Join directory names

This example joins directory names, wraps the output in double-quotes, and separates the directory names with a comma and space (`, `). The output is a string object.

```
Get-ChildItem -Directory C:\ | Join-String -Property Name -DoubleQuote
-Separator ', '

"PerfLogs", "Program Files", "Program Files (x86)", "Users", "Windows"
```

`Get-ChildItem` uses the **Directory** parameter to get all the directory names for the `C:\` drive. The objects are sent down the pipeline to `Join-String`. The **Property** parameter specifies the directory names. The **DoubleQuote** parameter wraps the directory names with double-quote marks. The **Separator** parameter specifies to use a comma and space (`, `) to separate the directory names.

The `Get-ChildItem` objects are **System.IO.DirectoryInfo** and `Join-String` converts the objects to **System.String**.

Example 2: Use a property substring to join directory names

This example uses a substring method to get the first four letters of directory names, wraps the output in single-quotes, and separates the directory names with a semicolon (`;`).

```
Get-ChildItem -Directory C:\ | Join-String -Property
{$_.Name.SubString(0,4)} -SingleQuote -Separator ';'

'Perf';'Prog';'Prog';'User';'Wind'
```

`Get-ChildItem` uses the **Directory** parameter to get all the directory names for the `C:\` drive. The objects are sent down the pipeline to `Join-String`.

The **Property** parameter script block uses automatic variable (`$_`) to specify each object's **Name** property substring. The substring gets the first four letters of each directory name. The substring specifies the character start and end positions. The **SingleQuote** parameter wraps the directory names with single-quote marks. The **Separator** parameter specifies to use a semicolon (`;`) to separate the directory names.

For more information about automatic variables and substrings, see about_Automatic_Variables and Substring.

Example 3: Display join output on a separate line

This example joins service names with each service on a separate line and indented by a tab.

```
Get-Service -Name se* | Join-String -Property Name -Separator "`r`n`t"
-OutputPrefix "Services:`n`t"

Services:
    seclogon
    SecurityHealthService
    SEMgrSvc
```

```
    SENS
    Sense
    SensorDataService
    SensorService
    SensrSvc
    SessionEnv
```

`Get-Service` uses the **Name** parameter with to specify services that begin with `se*`. The asterisk (*) is a wildcard for any character.

The objects are sent down the pipeline to `Join-String` that uses the **Property** parameter to specify the service names. The **Separator** parameter specifies three special characters that represent a carriage return (`` `r ``), newline (`` `n ``), and tab (`` `t ``). The **OutputPrefix** inserts a label `Services:` with a new line and tab before the first line of output.

For more information about special characters, see about_Special_Characters.

Example 4: Create a class definition from an object

This example generates a PowerShell class definition using an existing object as a template.

This code sample uses splatting to reduce the line length and improve readability. For more information, see about_Splatting.

```
$obj = [pscustomobject] @{Name = "Joe"; Age = 42}
$parms = @{
  Property = "Name"
  FormatString = '  ${0}'
  OutputPrefix = "class {`n"
  OutputSuffix = "`n}`n"
  Separator = "`n"
}
$obj.PSObject.Properties | Join-String @parms

class {
  $Name
  $Age
}
```

Parameters

-DoubleQuote

Wraps the string value of each pipeline object in double-quotes.

Type:	SwitchParameter
Position:	Named
Default value:	False

Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

-FormatString

Specifies a format string that specifies how each pipeline object should be formatted before joining them. Use the `{0}` placeholder to represent the current object. If you need to keep the curly braces (`{}`) in the formatted string, you can escape them by doubling the curly braces (`{{` and `}}`).

For more information, see the String.Format method and Composite Formatting.

Type:	String
Position:	Named
Default value:	None
Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

-InputObject

Specifies the text to be joined. Enter a variable that contains the text, or type a command or expression that gets the objects to join into strings.

Type:	PSObject[]
Position:	Named
Default value:	None
Required:	False
Accept pipeline input:	True
Accept wildcard characters:	False

-OutputPrefix

Text that's inserted before the output string. The string can contain special characters such as carriage return (`` `r ``), newline (`` `n ``), and tab (`` `t ``).

Type:	String
Aliases:	op
Position:	Named
Default value:	None

Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

-OutputSuffix

Text that's appended to the output string. The string can contain special characters such as carriage return (`r), newline (`n), and tab (`t).

Type:	String
Aliases:	os
Position:	Named
Default value:	None
Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

-Property

The name of a property, or a property expression, to be converted to text.

Type:	PSPropertyExpression
Position:	0
Default value:	None
Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

-Separator

Text or characters such as a comma or semicolon that's inserted between the text for each pipeline object.

By default, the pipeline objects are joined without a separator. If the Output Field Separator preference variable ($OFS) is set, that value is used unless this parameter is specified.

Note

When you set $OFS its value is used to join arrays when they're converted to strings until the variable is reset to $null. Because using $OFS can have unintended effects elsewhere in your code, it's best to use the **Separator** parameter instead.

Type:	String
Position:	1
Default value:	None
Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

-SingleQuote

Wraps the string value of each pipeline object in single quotes.

Type:	SwitchParameter
Position:	Named
Default value:	False
Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

-UseCulture

Uses the list separator for the current culture as the item delimiter. To find the list separator for a culture, use the following command: `(Get-Culture).TextInfo.ListSeparator`.

Type:	SwitchParameter
Position:	Named
Default value:	False
Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

Inputs

PSObject

Outputs

String

- about_Automatic_Variables
- about_Special_Characters

- Substring

Collaborate with us on GitHub
The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see our contributor guide.

In this article

Previous Chapter Next Chapter