Windows PowerShell Get-Help on Cmdlet 'ConvertFrom-Csv'

PS:\>Get-HELP ConvertFrom-Csv -Full

NAME

ConvertFrom-Csv

SYNOPSIS

Converts object properties in character-separated value (CSV) format into CSV versions of the original objects.

SYNTAX

ConvertFrom-Csv [-InputObject] <System.Management.Automation.PSObject[]> [[-Delimiter] <System.Char>] [-Header <System.String[]>] [<CommonParameters>]

ConvertFrom-Csv [-InputObject] <System.Management.Automation.PSObject[]> [-Header <System.String[]>] -UseCulture [<CommonParameters>]

DESCRIPTION

The `ConvertFrom-Csv` cmdlet converts character-separated value (CSV) data to PSObject type objects for each line of CSV data. The new objects are written to the

pipeline in the order they are read from the CSV data. The values in column header row of the CSV become the names of the properties added to each new PSObject .

The objects that `ConvertFrom-Csv` creates are PSObject type object for each row in the CSV file. The property values of the CSV objects are string versions of the

property values of the original objects. The CSV versions of the objects don't have any methods.

You can also use the `Export-Csv` and `Import-Csv` cmdlets to convert objects to CSV strings in a file and back. These cmdlets are the same as the `ConvertTo-Csv` and

`ConvertFrom-Csv` cmdlets, except that they save the CSV strings in a file.

The PSObject type maintains the order of the properties in column header order. This means that you get the same column order when you convert the objects back into

CSV format.

PARAMETERS

-Delimiter <System.Char>

Specifies the delimiter that separates the property values in the CSV strings. The default is a comma (`,`). Enter a character, such as a colon (`:`). To specify

a semicolon (`;`) enclose it in single quotation marks.

If you specify a character other than the actual string delimiter in the file, `ConvertFrom-Csv` can't create the objects from the CSV strings and returns the CSV

strings.

Required? false

Position? 1

Default value comma (,)

Accept pipeline input? False

Accept wildcard characters? false

-Header <System.String[]>

Specifies an alternate column header row for the imported string. The column header determines the property names of the objects created by `ConvertFrom-Csv`.

Enter column headers as a character-separated list. don't enclose the header string in quotation marks. Enclose each column header in single quotation marks.

If you enter fewer column headers than there are data columns, the remaining data columns are discarded. If you enter more column headers than there are data

columns, the additional column headers are created with empty data columns.

When using the Header parameter, omit the column header string from the CSV strings. Otherwise, this cmdlet creates an extra object from the items in the header

row.

Required? false

Position? named

Default value None

Accept pipeline input? False

Accept wildcard characters? false

-InputObject <System.Management.Automation.PSObject[]>

Specifies the CSV strings to be converted to objects. Enter a variable that contains the CSV strings or type a command or expression that gets the CSV strings.

You can also pipe the CSV strings to `ConvertFrom-Csv`.

Required? true

Position? 0

Default value None

Accept pipeline input? True (ByPropertyName, ByValue)

Accept wildcard characters? false

-UseCulture <System.Management.Automation.SwitchParameter>

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`.

Required? true

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.String

You can pipe CSV strings to this cmdlet.

OUTPUTS

System.Management.Automation.PSObject

This cmdlet returns the objects described by the properties in the CSV strings.

NOTES

In CSV format, each object is represented by a character-separated list of the property values of the object. The property values are converted to strings, using

the `ToString()` method of the object. There is no way to export the methods of the object.

Example 1: Convert processes on the local computer to CSV format

$P = Get-Process | ConvertTo-Csv

```
$P | ConvertFrom-Csv
```

The `Get-Process` cmdlet sends the processes down the pipeline to `ConvertTo-Csv`. The `ConvertTo-Csv` cmdlet converts the process objects to a series of CSV strings.

The `ConvertFrom-Csv` cmdlet converts the CSV strings into CSV versions of the original process objects. The CSV strings are saved in the `$P` variable.

Example 2: Convert a data object to CSV format and then to CSV object format

```
$Date = Get-Date | ConvertTo-Csv -Delimiter ';'
ConvertFrom-Csv -InputObject $Date -Delimiter ';'
```

The first command uses `Get-Date` to send the current date and time down the pipeline to `ConvertTo-Csv`. The `ConvertTo-Csv` cmdlet converts the date object to a

series of CSV strings. The Delimiter parameter is used to specify a semicolon delimiter. The strings are saved in the `$Date` variable.

Example 3: Use the header parameter to change the names of properties

```
$J = Start-Job -ScriptBlock { Get-Process } | ConvertTo-Csv  -NoTypeInformation
$Header = 'State', 'MoreData', 'StatusMessage', 'Location', 'Command',
        'StateInfo', 'Finished', 'InstanceId', 'Id', 'Name', 'ChildJobs',
        'BeginTime', 'EndTime', 'JobType', 'Output', 'Error', 'Progress',
        'Verbose', 'Debug', 'Warning', 'Information'
# Delete the default header from $J
$J = $J[1..($J.count - 1)]
$J | ConvertFrom-Csv -Header $Header

State         : Running
MoreData      : True
StatusMessage :
Location      : localhost
Command       : Get-Process
StateInfo     : Running
Finished      : System.Threading.ManualResetEvent
```

InstanceId : a259eb63-6824-4b97-a033-305108ae1c2e

Id : 1

Name : Job1

ChildJobs : System.Collections.Generic.List`1[System.Management.Automation.Job]

BeginTime : 12/20/2018 18:59:57

EndTime :

JobType : BackgroundJob

Output : System.Management.Automation.PSDataCollection`1[System.Management.Automation.PSObject]

Error : System.Management.Automation.PSDataCollection`1[System.Management.Automation.ErrorRecord]

Progress : System.Management.Automation.PSDataCollection`1[System.Management.Automation.ProgressRecord]

Verbose : System.Management.Automation.PSDataCollection`1[System.Management.Automation.VerboseRecord]

Debug : System.Management.Automation.PSDataCollection`1[System.Management.Automation.DebugRecord]

Warning : System.Management.Automation.PSDataCollection`1[System.Management.Automation.WarningRecord]

Information : System.Management.Automation.PSDataCollection`1[System.Management.Automation.InformationRecord]

The `Start-Job` cmdlet starts a background job that runs `Get-Process`. A job object is sent down the pipeline to `ConvertTo-Csv` and converted to a CSV string. The

NoTypeInformation parameter removes the type information header from CSV output and is optional in PowerShell v6 and higher. The `$Header` variable contains a custom

header that replaces the following default values: HasMoreData , JobStateInfo , PSBeginTime , PSEndTime , and PSJobTypeName . The `$J` variable contains the CSV

string and is used to remove the default header. The `ConvertFrom-Csv` cmdlet converts the CSV string into a PSCustomObject and uses the Header parameter to apply the

`$Header` variable.

------ Example 4: Convert CSV strings of service objects ------

```
(Get-Culture).TextInfo.ListSeparator
$Services = (Get-Service | ConvertTo-Csv)
ConvertFrom-Csv -InputObject $Services -UseCulture
```

The `Get-Culture` cmdlet uses the nested properties TextInfo and ListSeparator to get the current culture's default list separator. The `Get-Service` cmdlet sends

service objects down the pipeline to `ConvertTo-Csv`. The `ConvertTo-Csv` converts the service objects to a series of CSV strings. The CSV strings are stored in the

`$Services` variable. The `ConvertFrom-Csv` cmdlet uses the InputObject parameter and converts the CSV strings from the `$Services` variable. The UseCulture parameter

uses the current culture's default list separator.

When the UseCulture parameter is used, be sure that the current culture's default list separator matches the delimiter used in the CSV strings. Otherwise,

`ConvertFrom-Csv` can't generate objects from the CSV strings.

RELATED LINKS

Online Version: https://learn.microsoft.com/powershell/module/microsoft.powershell.utility/convertfrom-csv?view=powershell-5.1&WT.mc_id=ps-gethelp

ConvertTo-Csv

Export-Csv

Import-Csv