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

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

NAME

ConvertFrom-StringData

SYNOPSIS

Converts a string containing one or more key and value pairs to a hash table.

SYNTAX

ConvertFrom-StringData [-StringData] <System.String> [<CommonParameters>]

DESCRIPTION

The `ConvertFrom-StringData` cmdlet converts a string that contains one or more key and value pairs into a hash table. Because each key-value pair must be on a

separate line, here-strings are often used as the input format. By default, the key must be separated from the value by an equals sign (`=`) character.

The `ConvertFrom-StringData` cmdlet is considered to be a safe cmdlet that can be used in the DATA section of a script or function. When used in a DATA section, the

contents of the string must conform to the rules for a DATA section. For more information, see about_Data_Sections.

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

`ConvertFrom-StringData` supports escape character sequences that are allowed by conventional machine translation tools. That is, the cmdlet can interpret backslashes (``) as escape characters in the string data by using the Regex.Unescape Method (/dotnet/api/system.text.regularexpressions.regex.unescape), instead of the PowerShell backtick character (`` ` ``) that would normally signal the end of a line in a script. Inside the here-string, the backtick character does not work. You can also preserve a literal backslash in your results by escaping it with a preceding backslash, like this: `\`. Unescaped backslash characters, such as those that are commonly used in file paths, can render as illegal escape sequences in your results.

PARAMETERS

-StringData <System.String>

Specifies the string to be converted. You can use this parameter or pipe a string to `ConvertFrom-StringData`. The parameter name is optional.

The value of this parameter must be a string that contains one or more key-value pairs. Each key-value pair must be on a separate line, or each pair must be separated by newline characters (`` `n ``).

You can include comments in the string, but the comments cannot be on the same line as a key-value pair. `ConvertFrom-StringData` ignores single-line comments.

The `#` character must be the first non-whitespace character on the line. All characters on the line after the `#` are ignored. The comments are not included in the hash table.

A here-string is a string consisting of one or more lines. Quotation marks within the here-string are interpreted literally as part of the string data. For more information, see about_Quoting_Rules (../Microsoft.PowerShell.Core/About/about_Quoting_Rules.md).

Required? true

Position? 0

Default value None

Accept pipeline input? True (ByValue)

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 a string containing a key-value pair to this cmdlet.

OUTPUTS

System.Collections.Hashtable

This cmdlet returns a hash table that it creates from the key-value pairs.

NOTES

A here-string is a string consisting of one or more lines within which quotation marks are interpreted literally.

This cmdlet can be useful in scripts that display user messages in multiple spoken languages. You can use the dictionary-style hash tables to isolate text strings

from code, such as in resource files, and to format the text strings for use in translation tools.

Example 1: Convert a single-quoted here-string to a hash table

$Here = @'

```
Msg1 = The string parameter is required.
Msg2 = Credentials are required for this command.
Msg3 = The specified variable does not exist.
'@
ConvertFrom-StringData -StringData $Here

Name                           Value
----                           -----
Msg3                           The specified variable does not exist.
Msg2                           Credentials are required for this command.
Msg1                           The string parameter is required.


---- Example 2: Convert a here-string containing a comment ----

ConvertFrom-StringData -StringData @'
Name = Disks.ps1

# Category is optional.

Category = Storage
Cost = Free
'@

Name                           Value
----                           -----
Cost                           Free
Category                       Storage
Name                           Disks.ps1
```

The value of the StringData parameter is a here-string, instead of a variable that contains a here-string. Either format is valid. The here-string includes a comment

about one of the strings. `ConvertFrom-StringData` ignores single-line comments, but the `#` character must be the first

non-whitespace character on the line. All

characters on the line after the `#` are ignored.

--------- Example 3: Convert a string to a hash table ---------

```
$A = ConvertFrom-StringData -StringData "Top = Red `n Bottom = Blue"
$A

Name            Value
----            -----
Bottom          Blue
Top             Red
```

To satisfy the condition that each key-value pair must be on a separate line, the string uses the PowerShell newline character (`` `n ``) to separate the pairs.

Example 4: Use ConvertFrom-StringData in the DATA section of a script

```
$TextMsgs = DATA {
ConvertFrom-StringData @'
Text001 = The $Notebook variable contains the name of the user's system notebook.
Text002 = The $MyNotebook variable contains the name of the user's private notebook.
'@
}
$TextMsgs

Name            Value
----            -----
Text001         The $Notebook variable contains the name of the user's system notebook.
Text002         The $MyNotebook variable contains the name of the user's private notebook.
```

Because the text includes variable names, it must be enclosed in a single-quoted string so that the variables are interpreted literally and not expanded. Variables

are not permitted in the DATA section.

---- Example 5: Use the pipeline operator to pass a string ----

```
$Here = @'
Msg1 = The string parameter is required.
Msg2 = Credentials are required for this command.
Msg3 = The specified variable does not exist.
'@
$Hash = $Here | ConvertFrom-StringData
$Hash

Name     Value
----     -----
Msg3     The specified variable does not exist.
Msg2     Credentials are required for this command.
Msg1     The string parameter is required.
```

Example 6: Use escape characters to add new lines and return characters

```
ConvertFrom-StringData @"
 Vincentio = Heaven doth with us as we with torches do,\nNot light them for themselves; for if our virtues\nDid not go forth
of us, 'twere all alike\nAs if we had them
not.
Angelo = Let there be some more test made of my metal,\nBefore so noble and so great a figure\nBe stamp'd upon it.
"@ | Format-List

Name  : Angelo
Value : Let there be some more test made of my metal,
        Before so noble and so great a figure
        Be stamp'd upon it.

Name  : Vincentio
Value : Heaven doth with us as we with torches do,
        Not light them for themselves; for if our virtues
```

Did not go forth of us, 'twere all alike

As if we had them not.

Example 7: Use backslash escape character to correctly render a file path

ConvertFrom-StringData "Message=Look in c:\\Windows\\System32"

Name Value

---- -----

Message Look in c:\Windows\System32

RELATED LINKS

Online Version:

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