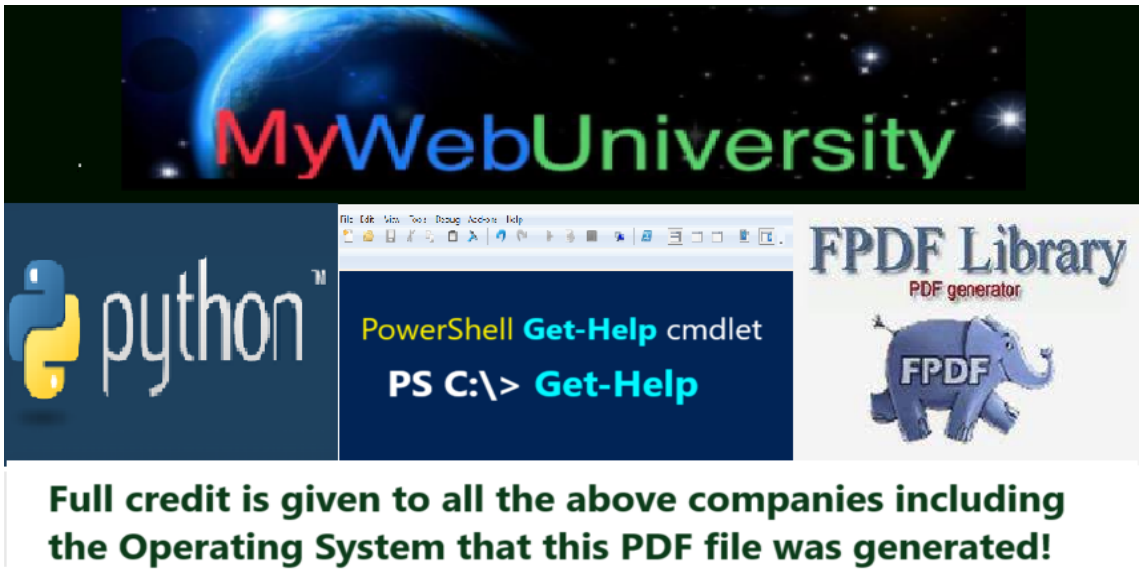

Windows PowerShell Get-Help on Cmdlet 'Get-Content'

PS:\>Get-HELP Get-Content -Full

NAME

Get-Content

SYNOPSIS

Gets the content of the item at the specified location.

SYNTAX

Get-Content [-Credential <System.Management.Automation.PSCredential>] [-Delimiter <System.String>] [-Encoding {ASCII | BigEndianUnicode | BigEndianUTF32 | Byte |

Default | OEM | String | Unicode | Unknown | UTF7 | UTF8 | UTF32}] [-Exclude <System.String[]>] [-Filter <System.String>] [-Force] [-Include <System.String[]>]

-LiteralPath <System.String[]> [-Raw] [-ReadCount <System.Int64>] [-Stream <System.String>] [-Tail <System.Int32>] [-TotalCount <System.Int64>] [-UseTransaction]

[-Wait] [<CommonParameters>]

Get-Content [-Path] <System.String[]> [-Credential <System.Management.Automation.PSCredential>] [-Delimiter <System.String>] [-Encoding {ASCII | BigEndianUnicode |

BigEndianUTF32 | Byte | Default | OEM | String | Unicode | Unknown | UTF7 | UTF8 | UTF32}] [-Exclude <System.String[]>] [-Filter <System.String>] [-Force] [-Include

<System.String[]>] [-Raw] [-ReadCount <System.Int64>] [-Stream <System.String>] [-Tail <System.Int32>] [-TotalCount <System.Int64>] [-UseTransaction] [-Wait]

[<CommonParameters>]

DESCRIPTION

The `Get-Content` cmdlet gets the content of the item at the location specified by the path, such as the text in a file or the content of a function. For files, the

content is read one line at a time and returns a collection of objects, each representing a line of content.

Beginning in PowerShell 3.0, `Get-Content` can also get a specified number of lines from the beginning or end of an item.

PARAMETERS

-Credential <System.Management.Automation.PSCredential>

> [!NOTE] > This parameter isn't supported by any providers installed with PowerShell. To impersonate another > user, or elevate your credentials when running

this cmdlet, use > Invoke-Command (../Microsoft.PowerShell.Core/Invoke-Command.md).

Required? false

Position? named

Default value Current user

Accept pipeline input? True (ByPropertyName)

Accept wildcard characters? false

-Delimiter <System.String>

Specifies the delimiter that `Get-Content` uses to divide the file into objects while it reads. The default is `\n`, the end-of-line character. When reading a

text file, `Get-Content` returns a collection of string objects, each ending with an end-of-line character. When you enter a delimiter that doesn't exist in the

file, `Get-Content` returns the entire file as a single, undelimited object.

You can use this parameter to split a large file into smaller files by specifying a file separator, as the delimiter. The

delimiter is preserved (not discarded)

and becomes the last item in each file section. Delimiter is a dynamic parameter that the FileSystem provider adds to the `Get-Content` cmdlet. This parameter

works only in file system drives.

> [!NOTE] > Currently, when the value of the Delimiter parameter is an empty string, `Get-Content` does > not return anything. This is a known issue. To force

`Get-Content` to return the entire file as > a single, undelimited string. Enter a value that doesn't exist in the file.

Required? false

Position? named

Default value End-of-line character

Accept pipeline input? False

Accept wildcard characters? false

-Encoding <Microsoft.PowerShell.Commands.FileSystemCmdletProviderEncoding>

Specifies the type of encoding for the target file. The default value is `Default`.

The acceptable values for this parameter are as follows:

- `Ascii` Uses ASCII (7-bit) character set.

- `BigEndianUnicode` Uses UTF-16 with the big-endian byte order.

- `BigEndianUTF32` Uses UTF-32 with the big-endian byte order.

- `Byte` Encodes a set of characters into a sequence of bytes.

- `Default` Uses the encoding that corresponds to the system's active code page (usually ANSI).

- `Oem` Uses the encoding that corresponds to the system's current OEM code page.

- `String` Same as `Unicode`.

- `Unicode` Uses UTF-16 with the little-endian byte order.

- `Unknown` Same as `Unicode`.

- `UTF7` Uses UTF-7.

- `UTF8` Uses UTF-8.

- `UTF32` Uses UTF-32 with the little-endian byte order.

Encoding is a dynamic parameter that the FileSystem provider adds to the `Get-Content` cmdlet. This parameter works only in file system drives.

When reading from and writing to binary files, use a value of Byte for the Encoding dynamic parameter and a value of 0 for the ReadCount parameter. A ReadCount

value of 0 reads the entire file in a single read operation and converts it into a single object (PSObject). The default ReadCount value, 1, reads one byte in

each read operation and converts each byte into a separate object, which causes errors when you use the `Set-Content` cmdlet to write the bytes to a file.

Required? false

Position? named

Default value Default

Accept pipeline input? False

Accept wildcard characters? false

-Exclude <System.String[]>

Specifies, as a string array, an item or items that this cmdlet excludes in the operation. The value of this parameter qualifies the Path parameter.

Enter a path element or pattern, such as `*.txt`. Wildcard characters are permitted.

The Exclude parameter is effective only when the command includes the contents of an item, such as `C:\Windows*`, where the wildcard character specifies the

contents of the `C:\Windows` directory.

Required? false

Position? named

Default value None

Accept pipeline input? False

Accept wildcard characters? true

-Filter <System.String>

Specifies a filter to qualify the Path parameter. The FileSystem (../Microsoft.PowerShell.Core/About/about_FileSystem_Provider.md)provider is the only installed

PowerShell provider that supports the use of filters. You can find the syntax for the FileSystem filter language in about_Wildcards

(../Microsoft.PowerShell.Core/About/about_Wildcards.md). Filters are more efficient than other parameters, because the provider applies them when the cmdlet gets

the objects rather than having PowerShell filter the objects after they're retrieved.

Required? false

Position? named

Default value None

Accept pipeline input? False

Accept wildcard characters? true

-Force <System.Management.Automation.SwitchParameter>

Force can override a read-only attribute or create directories to complete a file path. The Force parameter doesn't attempt to change file permissions or override

security restrictions.

Required? false

Position? named

Default value False

Accept pipeline input? False

Accept wildcard characters? false

-Include <System.String[]>

Specifies, as a string array, an item or items that this cmdlet includes in the operation. The value of this parameter qualifies the Path parameter. Enter a path

element or pattern, such as `" .txt"`. Wildcard characters are permitted. The Include * parameter is effective only when the command includes the contents of an

item, such as `C:\Windows*`, where the wildcard character specifies the contents of the `C:\Windows` directory.

Required? false

Position? named

Default value None

Accept pipeline input? False

Accept wildcard characters? true

-LiteralPath <System.String[]>

Specifies a path to one or more locations. The value of LiteralPath is used exactly as it's typed. No characters are interpreted as wildcards. If the path

includes escape characters, enclose it in single quotation marks. Single quotation marks tell PowerShell not to interpret any characters as escape sequences.

For more information, see about_Quoting_Rules (../Microsoft.Powershell.Core/About/about_Quoting_Rules.md).

Required? true

Position? named

Default value None

Accept pipeline input? True (ByPropertyName)

Accept wildcard characters? false

-Path <System.String[]>

Specifies the path to an item where `Get-Content` gets the content. Wildcard characters are permitted. The paths must

be paths to items, not to containers. For

example, you must specify a path to one or more files, not a path to a directory.

Required? true

Position? 0

Default value None

Accept pipeline input? True (ByPropertyName)

Accept wildcard characters? true

-Raw <System.Management.Automation.SwitchParameter>

Ignores newline characters and returns the entire contents of a file in one string with the newlines preserved. By default, newline characters in a file are used

as delimiters to separate the input into an array of strings. This parameter was introduced in PowerShell 3.0. Raw is a dynamic parameter that the FileSystem

provider adds to the `Get-Content` cmdlet This parameter works only in file system drives.

Required? false

Position? named

Default value False

Accept pipeline input? False

Accept wildcard characters? false

-ReadCount <System.Int64>

Specifies how many lines of content are sent through the pipeline at a time. The default value is 1. A value of 0 (zero) or negative numbers sends all the content

at one time.

This parameter doesn't change the content displayed, but it does affect the time it takes to display the content. As the value of ReadCount increases, the time it

takes to return the first line increases, but the total time for the operation decreases. This can make a perceptible difference in large items.

Required? false

Position? named

Default value 1

Accept pipeline input? True (ByPropertyName)

Accept wildcard characters? false

-Stream <System.String>

Gets the contents of the specified alternate NTFS file stream from the file. Enter the stream name. Wildcards aren't supported. Stream is a dynamic parameter that

the FileSystem provider adds to the `Get-Content` cmdlet. This parameter works only in file system drives on Windows systems.

This parameter was introduced in Windows PowerShell 3.0.

Required? false

Position? named

Default value None

Accept pipeline input? False

Accept wildcard characters? false

-Tail <System.Int32>

Specifies the number of lines from the end of a file or other item. You can use the Tail parameter name or its alias, Last . Negative values cause the cmdlet to

return the entire contents.

This parameter was introduced in PowerShell 3.0.

Required? false

Position? named

Default value None

Accept pipeline input? True (ByPropertyName)

Accept wildcard characters? false

-TotalCount <System.Int64>

Specifies the number of lines from the beginning of a file or other item. Negative values cause the cmdlet to return the entire contents.

You can use the TotalCount parameter name or its aliases, First or Head .

Required? false

Position? named

Default value None

Accept pipeline input? True (ByPropertyName)

Accept wildcard characters? false

-UseTransaction <System.Management.Automation.SwitchParameter>

Includes the command in the active transaction. This parameter is valid only when a transaction is in progress. For more information, see about_Transactions

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

Required? false

Position? named

Default value False

Accept pipeline input? False

Accept wildcard characters? false

-Wait <System.Management.Automation.SwitchParameter>

Causes the cmdlet to wait indefinitely, keeping the file open, until interrupted. While waiting, `Get-Content` checks the file once per second and outputs new

lines if present. When used with the TotalCount parameter, `Get-Content` waits until the specified number of lines are available in the specified file. For

example, if you specify a TotalCount of 10 and the file already has 10 or more lines, `Get-Content` returns the 10 lines and exits. If the file has fewer than 10

lines, `Get-Content` outputs each line as it arrives, but waits until the tenth line arrives before exiting.

You can interrupt Wait by pressing <kbd>Ctrl</kbd>+<kbd>C</kbd>. Deleting the file causes a non-terminating error that also interrupts the waiting. Wait is a

dynamic parameter that the FileSystem provider adds to the `Get-Content` cmdlet. This parameter works only in file system drives. Wait can't be combined with Raw .

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.Int64

You can pipe the read count or total count to this cmdlet.

System.String[]

You can pipe paths to this cmdlet.

System.Management.Automation.PSCredential

You can pipe credentials to this cmdlet.

OUTPUTS

System.Byte

When you use the AsByteStream parameter, this cmdlet returns the content as bytes.

System.String

By default, this cmdlet returns the content as an array of strings, one per line. When you use the Raw parameter, it returns a single string containing every line

in the file.

NOTES

Windows PowerShell includes the following aliases for `Get-Content`:

- `cat`

- `gc`

- `type`

The `Get-Content` cmdlet is designed to work with the data exposed by any provider. To get the providers in your session, use the `Get-PSProvider` cmdlet. For

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

---------- Example 1: Get the content of a text file ----------

```
1..100 | ForEach-Object { Add-Content -Path .\LineNumbers.txt -Value "This is line $_." }
Get-Content -Path .\LineNumbers.txt

This is Line 1
This is Line 2
...
This is line 99.
This is line 100.
```

The array values 1-100 are sent down the pipeline to the `ForEach-Object` cmdlet. `ForEach-Object` uses a script block with the `Add-Content` cmdlet to create the

`LineNumbers.txt` file. The variable `$_` represents the array values as each object is sent down the pipeline. The `Get-Content` cmdlet uses the Path parameter to

specify the `LineNumbers.txt` file and displays the content in the PowerShell console.

--- Example 2: Limit the number of lines Get-Content returns ---

```
Get-Content -Path .\LineNumbers.txt -TotalCount 5
```

```
This is Line 1
This is Line 2
This is Line 3
This is Line 4
This is Line 5
```

-- Example 3: Get a specific line of content from a text file --

```
(Get-Content -Path .\LineNumbers.txt -TotalCount 25)[-1]
```

```
This is Line 25
```

The `Get-Content` command is wrapped in parentheses so that the command completes before going to the next step. `Get-Content`returns an array of lines, this allows

you to add the index notation after the parenthesis to retrieve a specific line number. In this case, the `[-1]` index specifies the last index in the returned array

of 25 retrieved lines.

--------- Example 4: Get the last line of a text file ---------

```
Get-Item -Path .\LineNumbers.txt | Get-Content -Tail 1
```

```
This is Line 100
```

This example uses the `Get-Item` cmdlet to demonstrate that you can pipe files to `Get-Content`. The Tail parameter gets the last line of the file. This method is

faster than retrieving all the lines in a variable and using the `[-1]` index notation.

---- Example 5: Get the content of an alternate data stream ----

```
Set-Content -Path .\Stream.txt -Value 'This is the content of the Stream.txt file'
# Specify a wildcard to the Stream parameter to display all streams of the recently created file.
Get-Item -Path .\Stream.txt -Stream *

PSPath        : Microsoft.PowerShell.Core\FileSystem::C:\Test\Stream.txt::$DATA
PSParentPath  : Microsoft.PowerShell.Core\FileSystem::C:\Test
PSChildName   : Stream.txt::$DATA
PSDrive       : C
PSProvider    : Microsoft.PowerShell.Core\FileSystem
PSIsContainer : False
FileName      : C:\Test\Stream.txt
Stream        : :$DATA
Length        : 44

# Retrieve the content of the primary stream.
# Note the single quotes to prevent variable substitution.
Get-Content -Path .\Stream.txt -Stream ':$DATA'

This is the content of the Stream.txt file

# Alternative way to get the same content.
Get-Content -Path .\Stream.txt -Stream ""
# The primary stream doesn't need to be specified to get the primary stream of the file.
# This gets the same data as the prior two examples.
Get-Content -Path .\Stream.txt

This is the content of the Stream.txt file

# Use the Stream parameter of Add-Content to create a new Stream containing sample content.
$addContentSplat = @{
```

```
    Path = '.\Stream.txt'
    Stream = 'NewStream'
    Value = 'Added a stream named NewStream to Stream.txt'
}
Add-Content @addContentSplat

# Use Get-Item to verify the stream was created.
Get-Item -Path .\Stream.txt -Stream *

PSPath        : Microsoft.PowerShell.Core\FileSystem::C:\Test\Stream.txt::$DATA
PSParentPath  : Microsoft.PowerShell.Core\FileSystem::C:\Test
PSChildName   : Stream.txt::$DATA
PSDrive       : C
PSProvider    : Microsoft.PowerShell.Core\FileSystem
PSIsContainer : False
FileName      : C:\Test\Stream.txt
Stream        : :$DATA
Length        : 44

PSPath        : Microsoft.PowerShell.Core\FileSystem::C:\Test\Stream.txt:NewStream
PSParentPath  : Microsoft.PowerShell.Core\FileSystem::C:\Test
PSChildName   : Stream.txt:NewStream
PSDrive       : C
PSProvider    : Microsoft.PowerShell.Core\FileSystem
PSIsContainer : False
FileName      : C:\Test\Stream.txt
Stream        : NewStream
Length        : 46

# Retrieve the content of your newly created Stream.
Get-Content -Path .\Stream.txt -Stream NewStream

Added a stream named NewStream to Stream.txt
```

The Stream parameter is a dynamic parameter of the FileSystem provider (../microsoft.powershell.core/about/about_filesystem_provider.md#stream-string). By default

`Get-Content` only retrieves data from the default, or `:$DATA` stream. Streams can be used to store hidden data such as attributes, security settings, or other data.

---------- Example 6: Get raw content ----------

```
$raw = Get-Content -Path .\LineNumbers.txt -Raw
$lines = Get-Content -Path .\LineNumbers.txt
Write-Host "Raw contains $($raw.Count) lines."
Write-Host "Lines contains $($lines.Count) lines."

Raw contains 1 lines.
Lines contains 100 lines.
```

---------- Example 7: Use Filters with Get-Content ----------

```
Get-Content -Path C:\Temp\* -Filter *.log
```

--------- Example 8: Get file contents as a byte array ---------

```
$byteArray = Get-Content -Path C:\temp\test.txt -Encoding Byte -Raw
Get-Member -InputObject $bytearray

TypeName: System.Byte[]

Name          MemberType           Definition
----          ----------           ----------
Count         AliasProperty        Count = Length
Add           Method               int IList.Add(System.Object value)
```

The first command uses the Encoding parameter to get the stream of bytes from the file. The Raw parameter ensures that the bytes are returned as a `[System.Byte[]]`.

If the Raw parameter was absent, the return value is a stream of bytes, which is interpreted by PowerShell as `[System.Object[]]`.

RELATED LINKS

Online Version: https://learn.microsoft.com/powershell/module/microsoft.powershell.management/get-content?view=powershell-5.1&WT.mc_id=ps-gethelp

about_Automatic_Variables

about_Providers

Add-Content

Clear-Content

ForEach-Object

Get-PSProvider

Set-Content