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

PowerShell Get-Help on command 'Add-Content'

PS C:\Users\wahid> Get-Help Add-Content

NAME

Add-Content

SYNOPSIS

Adds content to the specified items, such as adding words to a file.

SYNTAX

Add-Content [-Value] <System.Object[]> [-Credential
<System.Management.Automation.PSCredential>] [-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[]> [-NoNewline] [-PassThru] [-Stream <System.String>]
[-UseTransaction] [-Confirm] [-WhatIf] [<CommonParameters>]

Add-Content [-Path] <System.String[]> [-Value] <System.Object[]> [-Credential
<System.Management.Automation.PSCredential>] [-Encoding {ASCII |
BigEndianUnicode | BigEndianUTF32 | Byte | Default | OEM | String | Unicode |
Unknown | UTF7 | UTF8 | UTF32}] [-Exclude <System.String[]>] [-Filter
<System.String>] [-Force] [-Include <System.String[]>] [-NoNewline]

[-PassThru] [-Stream <System.String>] [-UseTransaction] [-Confirm] [-WhatIf] [<CommonParameters>]

DESCRIPTION

The `Add-Content` cmdlet appends content to a specified item or file. Content can be passed in from the pipeline or specified by using the Value parameter.

If you need to create files or directories for the following examples, see New-Item (New-Item.md).

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).

-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 `Add-Content` cmdlet. This parameter works only in file system drives.

-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.

-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.

-Force <System.Management.Automation.SwitchParameter>

Overrides the read-only attribute, allowing you to add content to a read-only file. For example, Force overrides the read-only attribute but it doesn't change file permissions.

-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.

-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).

-NoNewline <System.Management.Automation.SwitchParameter>

Indicates that this cmdlet doesn't add a new line or carriage return to the content.

The string representations of the input objects are concatenated to form the output. No spaces or newlines are inserted between the output strings. No newline is added after the last output string.

-PassThru <System.Management.Automation.SwitchParameter>

Returns an object representing the added content. By default, this cmdlet doesn't generate any output.

-Path <System.String[]>

Specifies the path to the items that receive the additional 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. If you specify multiple paths, use commas to separate the paths.

-Stream <System.String>

Specifies an alternative data stream for content. If the stream doesn't exist, this cmdlet creates it. Wildcard characters aren't supported. Stream is a dynamic parameter that the FileSystem provider adds to `Add-Content`. This parameter works only in file system drives.

You can use the `Add-Content` cmdlet to change the content of any alternate data stream, such as `Zone.Identifier`. However, we don't recommend this as a way to eliminate security checks that block files that are downloaded from the Internet. If you verify that a downloaded file is safe, use the `Unblock-File` cmdlet.

This parameter was introduced in PowerShell 3.0.

-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).

-Value <System.Object[]>
Specifies the content to be added. Type a quoted string, such as **This
data is for internal use only , or specify an object that contains
content, such as the DateTime** object that `Get-Date` generates.

You can't specify the contents of a file by typing its path, because the
path is just a string. You can use a `Get-Content` command to get the
content and pass it to the Value parameter.

-Confirm <System.Management.Automation.SwitchParameter>
Prompts you for confirmation before running the cmdlet.

-WhatIf <System.Management.Automation.SwitchParameter>
Shows what would happen if the cmdlet runs. The cmdlet isn't run.

<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: Add a string to all text files with an exception -

Add-Content -Path .*.txt -Exclude help* -Value 'End of file'

The Path parameter specifies all `.txt` files in the current directory, but
the Exclude parameter ignores file names that match the specified pattern. The
Value parameter specifies the text string that is written to the files.

Use Get-Content (Get-Content.md)to display the contents of these files.

--- Example 2: Add a date to the end of the specified files ---

```
Add-Content -Path .\DateTimeFile1.log, .\DateTimeFile2.log -Value (Get-Date)
-PassThru
Get-Content -Path .\DateTimeFile1.log

Tuesday, May 14, 2019 8:24:27 AM
Tuesday, May 14, 2019 8:24:27 AM
5/14/2019 8:24:27 AM
```

The `Add-Content` cmdlet creates two new files in the current directory. The Value parameter contains the output of the `Get-Date` cmdlet. The PassThru parameter outputs the added contents to the pipeline. Because there is no other cmdlet to receive the output, it is displayed in the PowerShell console. The `Get-Content` cmdlet displays the updated file, `DateTimeFile1.log`.
Example 3: Add the contents of a specified file to another file

```
$From = Get-Content -Path .\CopyFromFile.txt
Add-Content -Path .\CopyToFile.txt -Value $From
Get-Content -Path .\CopyToFile.txt
```

- The `Get-Content` cmdlet gets the contents of `CopyFromFile.txt` and stores the contents in the `$From` variable. - The `Add-Content` cmdlet updates the `CopyToFile.txt` file using the contents of the `$From` variable. - The `Get-Content` cmdlet displays CopyToFile.txt.
Example 4: Add the contents of a specified file to another file using the pipeline

```
Get-Content -Path .\CopyFromFile.txt | Add-Content -Path .\CopyToFile.txt
Get-Content -Path .\CopyToFile.txt
```

The `Get-Content` cmdlet gets the contents of `CopyFromFile.txt`. The results are piped to the `Add-Content` cmdlet, which updates the `CopyToFile.txt`. The

last `Get-Content` cmdlet displays `CopyToFile.txt`.

-------- Example 5: Create a new file and copy content --------

```
Add-Content -Path .\NewFile.txt -Value (Get-Content -Path .\CopyFromFile.txt)
Get-Content -Path .\NewFile.txt
```

- The `Add-Content` cmdlet uses the Path and Value parameters to create a new file in the current directory. - The `Get-Content` cmdlet gets the contents of an existing file, `CopyFromFile.txt` and passes it to the Value parameter. The parentheses around the `Get-Content` cmdlet ensure that the command finishes before the `Add-Content` command begins. - The `Get-Content` cmdlet displays the contents of the new file, `NewFile.txt`.

---------- Example 6: Add content to a read-only file ----------

```
New-Item -Path .\IsReadOnlyTextFile.txt -ItemType File
Set-ItemProperty -Path .\IsReadOnlyTextFile.txt -Name IsReadOnly -Value $True
Get-ChildItem -Path .\IsReadOnlyTextFile.txt
Add-Content -Path .\IsReadOnlyTextFile.txt -Value 'Add value to read-only text
file' -Force
Get-Content -Path .\IsReadOnlyTextFile.txt
```

```
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-ar--         1/28/2019     13:35              0 IsReadOnlyTextFile.txt
```

- The `New-Item` cmdlet uses the Path and ItemType parameters to create the file `IsReadOnlyTextFile.txt` in the current directory. - The `Set-ItemProperty` cmdlet uses the Name and Value parameters to change the file's IsReadOnly property to True. - The `Get-ChildItem` cmdlet shows the file is empty (`0`) and has the read-only attribute (`r`).

- The `Add-Content` cmdlet uses the Path parameter to specify the file. The Value parameter includes the text string to append to the file. The Force

parameter writes the text to the read-only file. - The `Get-Content` cmdlet uses the Path parameter to display the file's contents.

To remove the read-only attribute, use the `Set-ItemProperty` command with the Value parameter set to `False`.

REMARKS

To see the examples, type: "get-help Add-Content -examples".

For more information, type: "get-help Add-Content -detailed".

For technical information, type: "get-help Add-Content -full".

For online help, type: "get-help Add-Content -online"