python

PowerShell

FPDF Library

PDF generator

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

PowerShell Get-Help on command 'Set-Content'

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

NAME

Set-Content

SYNOPSIS

Writes new content or replaces existing content in a file.

SYNTAX

Set-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>]

Set-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

`Set-Content` is a string-processing cmdlet that writes new content or replaces the content in a file. `Set-Content` replaces the existing content and differs from the `Add-Content` cmdlet that appends content to a file. To send content to `Set-Content` you can use the Value parameter on the command line or send content through the pipeline.

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 is not 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>

This is a dynamic parameter made available by the FileSystem provider. For more information, see about_FileSystem_Provider (../Microsoft.PowerShell.Core/About/about_FileSystem_Provider.md).

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

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

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 `Set-Content`. 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 are retrieved.

-Force <System.Management.Automation.SwitchParameter>

Forces the cmdlet to set the contents of a file, even if the file is read-only. Implementation varies from provider to provider. For more information, see about_Providers (../Microsoft.PowerShell.Core/About/about_Providers.md). The Force parameter does not override security restrictions.

-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 is 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>

This is a dynamic parameter made available by the FileSystem provider. For more information, see about_FileSystem_Provider (../Microsoft.PowerShell.Core/About/about_FileSystem_Provider.md).

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 that represents the content. By default, this cmdlet does not generate any output.

-Path <System.String[]>

Specifies the path of the item that receives the content. Wildcard characters are permitted.

-Stream <System.String>

This is a dynamic parameter made available by the FileSystem provider. This Parameter is only available on Windows. For more information, see about_FileSystem_Provider

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

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

You can use the `Set-Content` cmdlet to create or update the content of any alternate data stream, such as `Zone.Identifier`. However, we do not 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 new content for the item.

-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 is not 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: Replace the contents of multiple files in a directory

```
Get-ChildItem -Path .\Test*.txt

Test1.txt
Test2.txt
Test3.txt

Set-Content -Path .\Test*.txt -Value 'Hello, World'
Get-Content -Path .\Test*.txt

Hello, World
Hello, World
Hello, World
```

The `Get-ChildItem` cmdlet uses the Path parameter to list .txt files that begin with `Test ` in the current directory. The `Set-Content` cmdlet uses the Path * parameter to specify the `Test .txt` files. The Value parameter provides the text string Hello, World * that replaces the existing content in each file. The `Get-Content` cmdlet uses the Path parameter to specify the `Test*.txt` files and displays each file's content in the PowerShell console.
-------- Example 2: Create a new file and write content --------

```
Set-Content -Path .\DateTime.txt -Value (Get-Date)
Get-Content -Path .\DateTime.txt

1/30/2019 09:55:08
```

`Set-Content` uses the Path and Value parameters to create a new file named DateTime.txt in the current directory. The Value parameter uses `Get-Date` to

get the current date and time. `Set-Content` writes the DateTime object to the file as a string. The `Get-Content` cmdlet uses the Path parameter to display the content of DateTime.txt in the PowerShell console.

---------- Example 3: Replace text in a file ----------

```
Get-Content -Path .\Notice.txt

Warning
Replace Warning with a new word.
The word Warning was replaced.

(Get-Content -Path .\Notice.txt) |
    ForEach-Object {$_ -Replace 'Warning', 'Caution'} |
        Set-Content -Path .\Notice.txt
Get-Content -Path .\Notice.txt

Caution
Replace Caution with a new word.
The word Caution was replaced.
```

The `Get-Content` cmdlet uses the Path parameter to specify the Notice.txt file in the current directory. The `Get-Content` command is wrapped with parentheses so that the command finishes before being sent down the pipeline.

The contents of the Notice.txt file are sent down the pipeline to the `ForEach-Object` cmdlet. `ForEach-Object` uses the automatic variable `$_` and replaces each occurrence of Warning with Caution . The objects are sent down the pipeline to the `Set-Content` cmdlet. `Set-Content` uses the Path parameter to specify the Notice.txt file and writes the updated content to the file.

The last `Get-Content` cmdlet displays the updated file content in the PowerShell console.

---------- Example 4: Use Filters with Set-Content ----------

Set-Content -Path C:\Temp* -Filter *.txt -Value "Empty"

REMARKS

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

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

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

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