

How to use PowerShell Replace to replace a String

 lazyadmin.nl/powershell/powershell-replace

February 10, 2022

Do you need to replace a string or character in PowerShell? Or do you want to insert complete text blocks into a file? Then the replace function in PowerShell is what you are looking for.

With Replace method or `-replace` operator in PowerShell, we can find and replace any character or (part of) strings with other data.

In this article, we are going to take a look at the different methods to replace a string in PowerShell, and I will also give you some useful examples when it comes to the replace function.

Powershell Replace Character in String

In PowerShell we can use the `Replace()` method on any string or variable that is a string. The method needs two arguments, the text or character that you want to find and the with what you want to replace it with.

Explaining the PowerShell Replace function is best done with examples, so let's start with some simple examples, like replacing a character or word in a string.

Let's take the following string:

```
$str = "Welcome to LazyAdmin"
```

To replace the word `to` into `at` we can use the following method:

```
$str = $str.Replace('to','at')
```



```
PS C:\> $str = "Welcome to LazyAdmin"
PS C:\> $str = $str.Replace('to','at')
PS C:\> $str
Welcome at LazyAdmin
PS C:\> |
```

I have stored the result of the replace method into the same variable. You don't need to use variables to use the replace method in PowerShell, you can also directly replace the string if you want.

```
"Welcome to LazyAdmin".Replace('to','at')
```

Another example that I often use is to replace the slashes in a folder path:

```
$Paths = "folder\subfolder"
$Paths.replace('\','\\')
# Result
folder\\subfolder
```

Removing Characters or Words from a String

We can also use the replace method to remove words or characters from strings. To do this we only need to replace the content with an empty string, like this:

```
# Replace the word 'to' with nothing
$str.Replace("to", "")
```

Let's take a list of email addresses and remove the domain name and @ symbol from it, so we are only left with the usernames. We are going to loop through the array with mail addresses and store the result in \$usernames:

```
$mailAddresses =
@("AdeleV@lazydev.com", "GradyA@lazydev.com", "JoniS@lazydev.com")
$usernames = @()
$mailAddresses | ForEach{
    $usernames += $_.Replace("@lazydev.com", "")
}
# Result
AdeleV
GradyA
JoniS
```

Insert text into templates with Replace

Some scripts that I have created sent an email with information, for example, the username and password of a new user, or an overview of all users without MFA enabled.

For those emails, I have created an HTML email template and inside this template, I have placed placeholders for the username and password.

```
<p>Hi {{manager.firstname}},</p>
<p>The account for {{user.fullname}} is ready. Below you will find the login credentials for {{user.firstname}}. The password is temporary and needs to be changed after the first login.</p>
<p>
<table class="details">
<tr>
<td style="font-family:sans-serif;font-size:14px;">Username: </td>
<td style="color:#000000;"><strong>{{user.UserPrincipalName}}</strong></td>
</tr>
<tr>
<td style="font-family:sans-serif;font-size:14px;">Password: </td>
```

```
<td><strong>{{user.Password}}</strong></td>
</tr>
</table>
</p>
```

```
<p>If you have any questions, please let us know</p>
<p>IT</p>
```

As you can see in the HTML template, I have created placeholders with curly brackets `{{ }}`. With PowerShell we can load the content of the HTML file and use `replace` to replace all the placeholders with the correct value:

```
#Get the mailtemplate
$mailTemplate = (Get-Content ($rootPath + '\ ' + $config.Settings.mailTemplateManager)) |
ForEach-Object {
$_ -replace '{{manager.firstname}}', $manager.GivenName `
-replace '{{user.UserPrincipalName}}', $user.UserPrincipalname `
-replace '{{user.Password}}', $config.Settings.password `
-replace '{{user.fullname}}', $user.fullName `
-replace '{{user.firstname}}', $user.givenName
} | Out-String
return $mailTemplate
```

In this case, I have used the `-replace` operator but you can also use the `replace()` method to replace multiple instances:

```
$str = "Welcome to LazyAdmin"
$str.Replace('Welcome','Hello').Replace('to ','')
# Result
Hello LazyAdmin
```

FREE EMAIL SERIES!

Level Up with PowerShell

5 Emails, Endless Skills



Powershell Replace Regex

Besides the `Replace()` method, we also have the `-replace` operator. Basically, they can do the same, replace any character or word in a string with something else. But there is a big difference between the two, the `-replace` operator uses regex to match the find part.

Note

The `-replace` operator only uses regex to find the keyword. It doesn't use regex on the replace part of the operator.

Let's take the following example, I have a list of log files that I want to rename from `.log` to `.txt` for example.

```
PS7 x + v
PS C:\> Get-ChildItem c:\temp\files

Directory: C:\temp\files

Mode                LastWriteTime         Length Name
----                -
-a---             8-2-2022   15:55             0 la-ams-ad01-log-20221.log
-a---             8-2-2022   15:55             0 la-ams-ad01-log-20222.log
-a---             8-2-2022   15:56             0 la-ams-file02-log-20221.log
-a---             8-2-2022   15:56             0 la-osl-ad01-log-20221.log
-a---             8-2-2022   15:56             0 la-osl-file01-log-20221.log

PS C:\> |
```

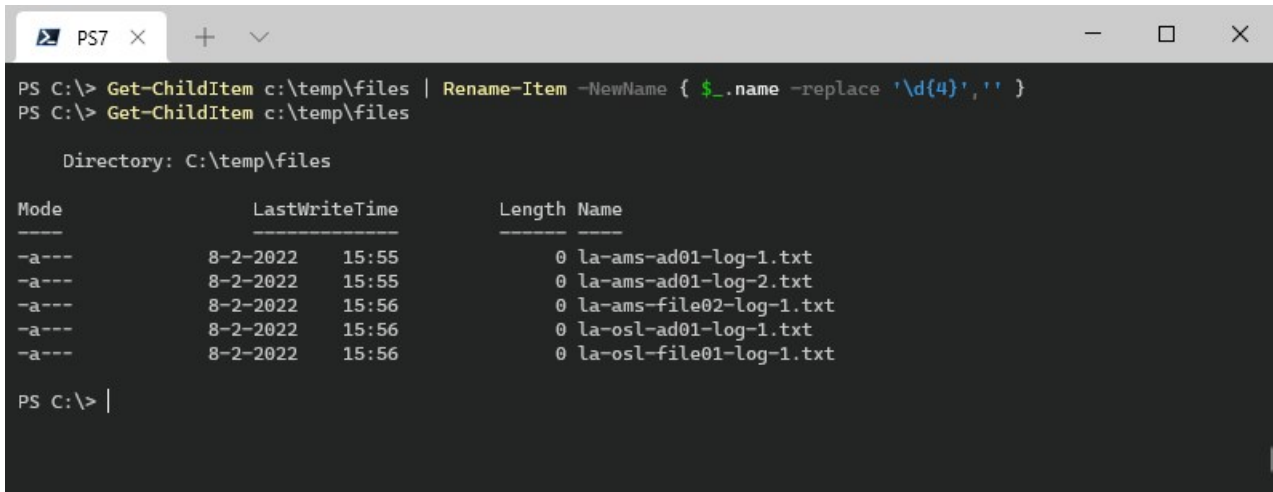
To do this we can use the `-replace` operator in PowerShell to select the `.log` from the end of the string and replace it with `.txt`.

```
Get-ChildItem c:\temp\files | Rename-Item -NewName { $_.name -replace '\.log$', '.txt' }
```

Now in this case we could also use the `replace()` method because `.log` is unique enough to select and replace. But if we want to remove the year from the file name then we will need to use regular expressions for this:

```
# Find 4 digits with \d{4}
```

```
Get-ChildItem c:\temp\files | Rename-Item -NewName { $_.name -replace '\d{4}', '' }
```



```
PS C:\> Get-ChildItem c:\temp\files | Rename-Item -NewName { $_.name -replace '\d{4}', '' }
PS C:\> Get-ChildItem c:\temp\files

Directory: C:\temp\files

Mode                LastWriteTime         Length Name
----                -
-a---             8-2-2022   15:55             0 la-ams-ad01-log-1.txt
-a---             8-2-2022   15:55             0 la-ams-ad01-log-2.txt
-a---             8-2-2022   15:56             0 la-ams-file02-log-1.txt
-a---             8-2-2022   15:56             0 la-osl-ad01-log-1.txt
-a---             8-2-2022   15:56             0 la-osl-file01-log-1.txt

PS C:\> |
```

Creating regular expression is always a bit challenging if you don't do it every day. A good site to test your expressions is regex101.com.

Case Sensitive and Case Insensitive Replacement

The `-Replace` operator is by default case insensitive. You can make it case sensitive by using the `-creplace` operator instead of `-replace`. For example:

```
$str = "Hello LazyAdmin"
# Case insensitive
$str -replace 'hello', 'Goodbye';
# Result : Goodbye LazyAdmin
# Case sensitive
$str -creplace 'hello', 'Goodbye';
# Result : Hello LazyAdmin
```

Powershell Replace Special Characters

With the `replace` operator in PowerShell we can easily replace all special characters in a string. Regex allows us to select all non-word characters, everything except `a-z`, `A-Z`, and `0-9`, to something else.

For example:

```
$str = "Long [String] with _special@character#123$% and (spaces)"
$str -replace ('\W', '')
# Result
LongStringwith_specialcharacter123andspaces
```

If you want to remove all special characters, except the spaces then we need to use a slightly different regex expression:

```
$str = "Long [String] with _special@character#123$% and (spaces)"
```

```
$str -replace ('[^a-zA-Z\d\s:]', '')
```

Result

Long String with specialcharacter123 and spaces

Wrapping Up

Replacing characters or words in a string with PowerShell is easily done using either the replace method or `-replace` operator. When working with special characters, like `[]`, `\` or `$` symbols, it's often easier to use the `replace()` method than the operator variant. Because this way you don't need to escape the special character.

And keep in mind, always test your script before running it against real data or files.

If you have any questions, then just drop a comment below.

Did you **Liked** this **Article**?

Get the latest articles like this **in your mailbox**

or share this article

I hate spam to, so you can unsubscribe at any time.