**Split**

The split keyword can be used to extract data out of a string. Say there is an email address inside a string and you want to get it, or you want to extract some useful information out of a text string.

Let's go through an example to understand it more. Let's extract the first name and last name from an email address string. You will use split because it can split the string on any character and convert it into array. You will split the email address on the dot (.). After that, in element 0 of the array you get the first name, but for last name you must split again at character @.

$email = "Vikas.Sukhija@labtest.com"

$emsplit = $email.split(".")

$firstname = $emsplit[0]

$lastname = ($emsplit[1] -split "@")

$lastn = $lastname[0]

$emsplit[0] and $lastname[0]

See the step-by-step split operation in [Figure 9-1](https://cdn2.percipio.com/1651640039.ab7862f216d102e2482406b4b912b6636c073c64/eod/books/158621/OEBPS/section-45.xhtml#ch09Fig1) for a better understanding of how to use it.

Figure 9-1: Showing the split operation



**Replace**

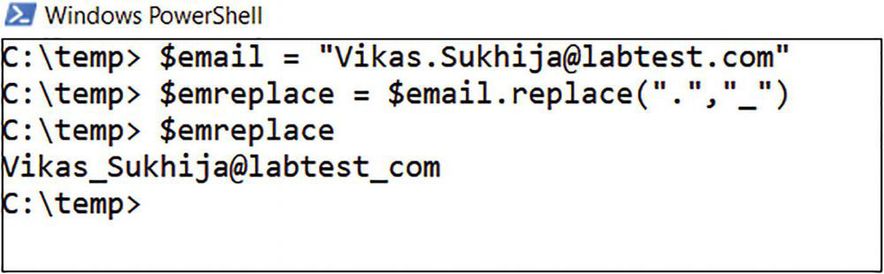
Another keyword is replace. Instead of splitting the string, you can replace the content of the string with other content.

You can use replace when you want to replace data in a string. Say you want to add an underscore instead of a dot because you want to update a secondary address. You can use this code and see the result in [Figure 9-2](https://cdn2.percipio.com/1651640039.ab7862f216d102e2482406b4b912b6636c073c64/eod/books/158621/OEBPS/section-46.xhtml#ch09Fig2):

$email = "Vikas.Sukhija@labtest.com"

$emreplace = $email.replace(".","\_")

Figure 9-2: Showing a replace operation



**Select-String**

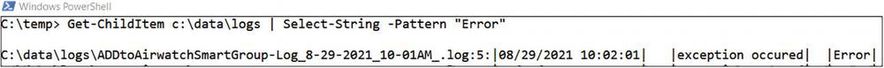
Select-String can do wonders because you can use it to find strings inside files. Here is a practical use for it, which I have used many times (while others have struggled and spent ample hours trying to solve): finding the right date and time of an operation from a large number of log files.

Say you have a large number of files inside the logs folder and you just want to find the files where the string error is present:

Get-ChildItem c:\data\logs | Select-String -Pattern "Error"

This simple one-liner will search for the string error in all of the log files inside the logs folder. [Figure 9-3](https://cdn2.percipio.com/1651640039.ab7862f216d102e2482406b4b912b6636c073c64/eod/books/158621/OEBPS/section-47.xhtml#ch09Fig3) show how it extracted the file name of the file that has the error.

Figure 9-3: Showing a Select-String operation



**Compare-Object**

Compare-Object (alias Compare) is used many times to compare two files or two arrays. It is faster than comparing the arrays or files using for loops. I use it many times for fetching members from a group and comparing them with a text file that has user IDs. This approach fetches only members that are not already part of the group and adds them, instead of processing all members.

[Listing 9-1](https://cdn2.percipio.com/1651640039.ab7862f216d102e2482406b4b912b6636c073c64/eod/books/158621/OEBPS/section-48.xhtml#ls9-1) adds members from one group to another.

**Note**

The Active Directory module is required for this to work.

**Listing 9-1: Cheat Code for Adding Members Using Compare-Object**

##########################fetching group1 ##################

$collgroup1 = Get-ADGroup -id "group1" -Properties member |

Select-Object -ExpandProperty member |

Get-ADUser |

Select-Object -ExpandProperty samaccountname

##########################fetching group2 ##################

$collgroup2 = Get-ADGroup -id "group2" -Properties member |

Select-Object -ExpandProperty member |

Get-ADUser |

Select-Object -ExpandProperty samaccountname

####################compare two groups#####################

$change = Compare-Object -ReferenceObject $collgroup1 -DifferenceObject $collgroup2

$Addition = $change |

Where-Object -FilterScript {$\_.SideIndicator -eq "<="} |

Select-Object -ExpandProperty InputObject

#######adding only members that are missing in group2##############

$Addition | ForEach-Object{

$sam = $\_

Add-ADGroupMember -identity "group2" -Members $sam

}

##############################################################

Similarly, you can do a remove operation by using Compare-Object, as shown in [Listing 9-2](https://cdn2.percipio.com/1651640039.ab7862f216d102e2482406b4b912b6636c073c64/eod/books/158621/OEBPS/section-48.xhtml#ls9-2).

**Listing 9-2: Cheat Code for Removing Members Using Compare-Object**

##########################fetching group1 ##################

$collgroup1 = Get-ADGroup -id "group1" -Properties member |

Select-Object -ExpandProperty member |

Get-ADUser |

Select-Object -ExpandProperty samaccountname

##########################fetching group2 ##################

$collgroup2 = Get-ADGroup -id "group2" -Properties member |

Select-Object -ExpandProperty member |

Get-ADUser |

Select-Object -ExpandProperty samaccountname

####################compare two groups####################

$change = Compare-Object -ReferenceObject $collgroup1 -DifferenceObject $collgroup2

$Removal = $change |

Where-Object -FilterScript {$\_.SideIndicator -eq "=>"} |

Select-Object -ExpandProperty InputObject

####Removing members that are in group2 but not in group1########

$Removal | ForEach-Object{

$sam = $\_

Remove-ADGroupMember -identity "group2" -Members $sam -confirm:$false

}

##############################################################

You can combine both operations in one script and synchronize two groups based on group1 as the anchor. [Listing 9-3](https://cdn2.percipio.com/1651640039.ab7862f216d102e2482406b4b912b6636c073c64/eod/books/158621/OEBPS/section-48.xhtml#ls9-3) shows this operation.

**Listing 9-3: Cheat Code for Synchronizing Two Groups Using Compare-Object (Based on group1 as the Anchor)**

##########################fetching group1 ##################

$collgroup1 = Get-ADGroup -id "group1" -Properties member |

Select-Object -ExpandProperty member |

Get-ADUser |

Select-Object -ExpandProperty samaccountname

##########################fetching group2 ##################

$collgroup2 = Get-ADGroup -id "group2" -Properties member |

Select-Object -ExpandProperty member |

Get-ADUser |

Select-Object -ExpandProperty samaccountname

####################compare two groups####################

$change = Compare-Object -ReferenceObject $collgroup1 -DifferenceObject $collgroup2

$Addition = $change |

Where-Object -FilterScript {$\_.SideIndicator -eq "<="} |

Select-Object -ExpandProperty InputObject

$Removal = $change |

Where-Object -FilterScript {$\_.SideIndicator -eq "=>"} |

Select-Object -ExpandProperty InputObject

#######adding only members that are missing in group2########

$Addition | ForEach-Object{

$sam = $\_

Add-ADGroupMember -identity "group2" -Members $sam

}

####Removing members that are in group2 but not in group1########

$Removal | ForEach-Object{

$sam = $\_

Remove-ADGroupMember -identity "group2" -Members $sam -confirm:$false

}

##############################################################

You can also use the other approach, so instead of removing from group2 you just use ADD-Groupmember for group1 so you can truly synchronize both groups. Any user object that is not present in group2 but is in group1 should be added to group2, and any user object not present in group1 but in group2 should be added to group1:

ADD-ADGroupMember -identity "group1" -Members $sam

instead of

Remove-ADGroupMember -identity "group2" -Members $sam -confirm:$false

There are other nice tricks you can perform with Compare-Object. Say you have two CSV files. One just has email addresses of users; the other has email addresses and other properties. You want all details from CSV file 2 for the users in CSV file one.

[Listing 9-4](https://cdn2.percipio.com/1651640039.ab7862f216d102e2482406b4b912b6636c073c64/eod/books/158621/OEBPS/section-48.xhtml#ls9-4) shows an example for OneDrive properties. There are two CSV files. One contains user email addresses and the other contains email addresses and other properties in other columns.

**Listing 9-4: Cheat Code for Merging Two CSV Files Using Compare-Object**

$importallonedrivesites = import-csv "c:\importonedrives.csv" # onedrive file with other attributes

$importspofile = import-csv "c:\users.csv" #users email addresses

$change = Compare-Object -ReferenceObject $importallonedrivesites -DifferenceObject $importspofile -Property owner -IncludeEqual -PassThru #owner is the column name for users email addreses

$change | where{$\_.SideIndicator -eq "==" -or $\_.SideIndicator -eq "=>"} |

select Owner, Title, url, StorageUsageCurrent, StorageQuota, StorageQuotaWarningLevel |

Export-Csv "c:\newfile.csv" -NoTypeInformation