PowerShell and Import-Csv CmdLet, Adding Headers – Part 3

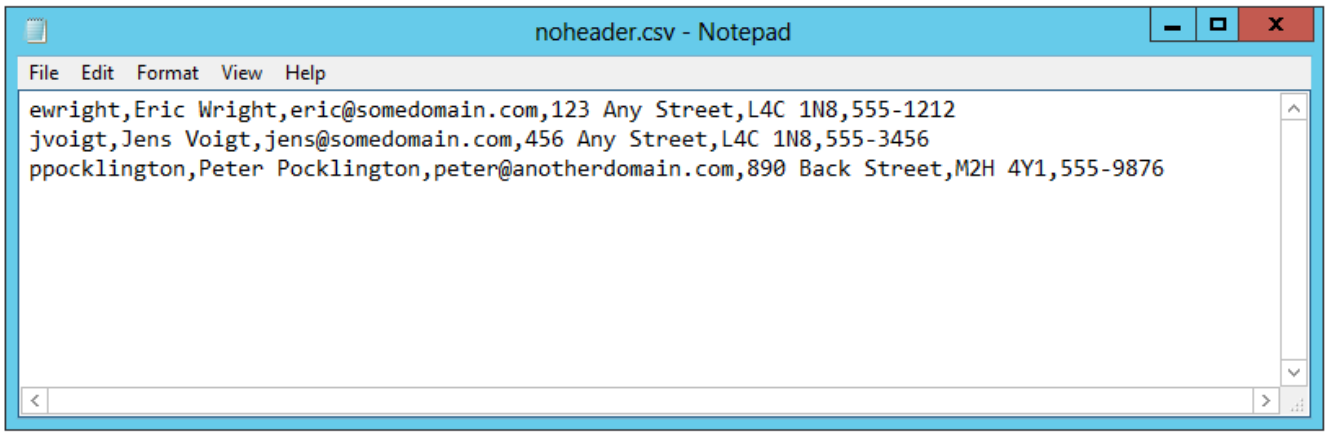
This was a long overdue post, so thanks for sticking with me while I finally got back on track with our CSV, yeah you know me series (Here are Part 1 and Part 2). As I'd mentioned in the closing of Part 2, we want to be able to deal with files that have no header in the CSV which we haven't encountered yet. Also, we want to expand the use of our CSV content by running the Get-Contact query and creating a contact in the event that one doesn't exist already.

PowerShell Import-CSV and Headers – The Head(er)less Horseman

In our previous posts, we have been using a file which contained some field data in the file and a header row containing the field names which made our job nice and easy with parsing out the data.

Let's take a look at the same source file, but this time we have no header row and our task will be to insert a header. This isn't necessarily a common task, but we may find that we receive some batch data from a vendor and they have either no header, or the field names would have been useless to us, so we opt to create our own header row for ease of use.

Here is our starter file named **NoHeader.CSV**



We know that the field names that we wish to apply are as follows: **username,full**

name,email,address,postalcode,phone so this will become part of our routine before parsing the data.

I'm a big fan of leaving source data intact when manipulation of the content occurs, so the steps I'm going to use include working with a new file which is populated from the clean source file and my custom inserted header row. Getting back to basics, this was the command to import the data to begin with.

```
$userobjects = Import-CSV x:ImportDataUserList.CSV
```

Now that we want to start with a clean file though, we change the tactic to reading the content using the **Get-Content** CmdLet and then piping the output into a new file. But first, let's setup our new target file.

I'm also going to start referring to our script as **CSVProcess.PS1** rather than just pasting all of my content in the command line each time.

When using temporary files we need to do some simple steps:

1. Define the file name and location (best done in a variable)
2. Empty out the file if it exists because we are creating the content on the fly
3. Send the header row data to the new empty file
4. Append the content from our header-less file
5. Revel in the joy of having done a cool task

So our new filename will be called **ImportData.CSV** and I will assign it a variable **$ImportFile** at the start of our script. While we are at it, we will also define the NoHeader.CSV file as $ImportFileNoHeader at the same time:

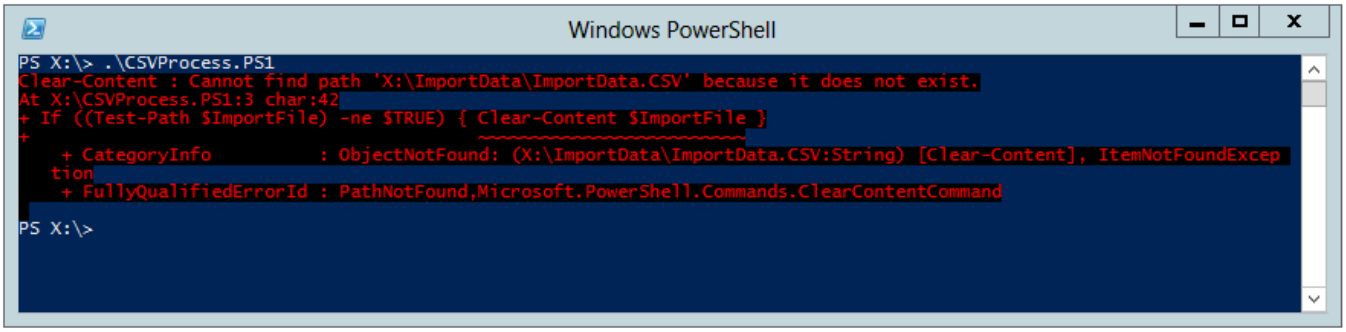
```
$ImportFile = "x:ImportDataImportData.CSV"

$ImportFileNoHeader = "x:ImportDataNoHeader.CSV"
```

Next we will use the **Clear-Content** CmdLet to empty out the file. The first time this line is executed it will generate an error because the file doesn't exist, but that's not a big problem because it will be created during the first script run:

```
Clear-Content $ImportFile
```

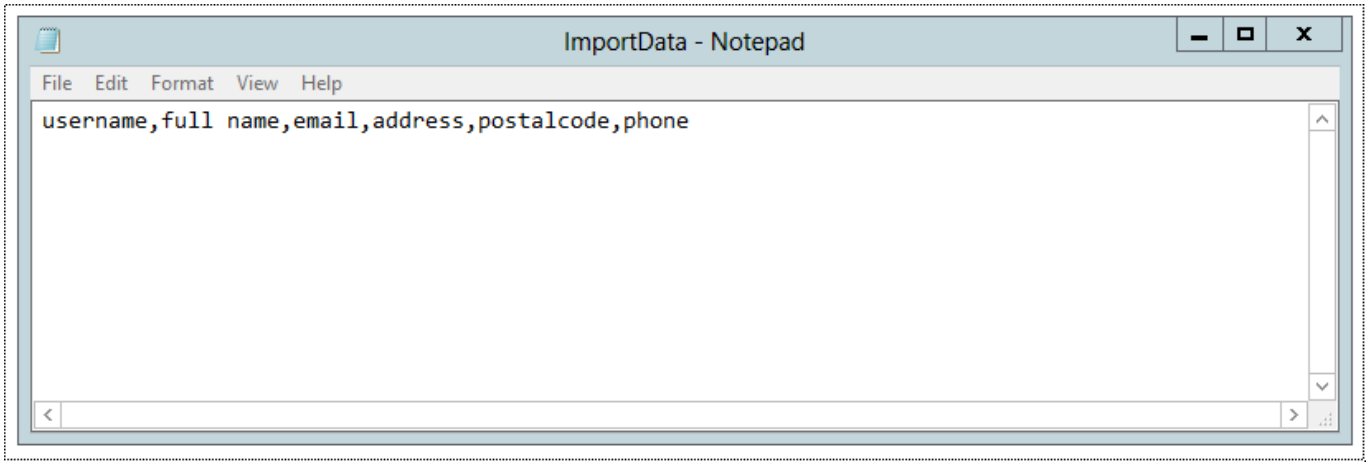
This is the error because if the missing file. We could handle the error and be fancy, but because this is a one-time error I would rather skip on lots of coding. I've used the **Test-Path** CmdLet for checking the file location, but it's overkill for the simple process we are running here.



So now we have our empty file (**ImportData.CSV**) and we will use the **Add-Content** CmdLet to append our header row:

```
Add-Content $ImportFile “username,full name,email,address,postalcode,phone”
```

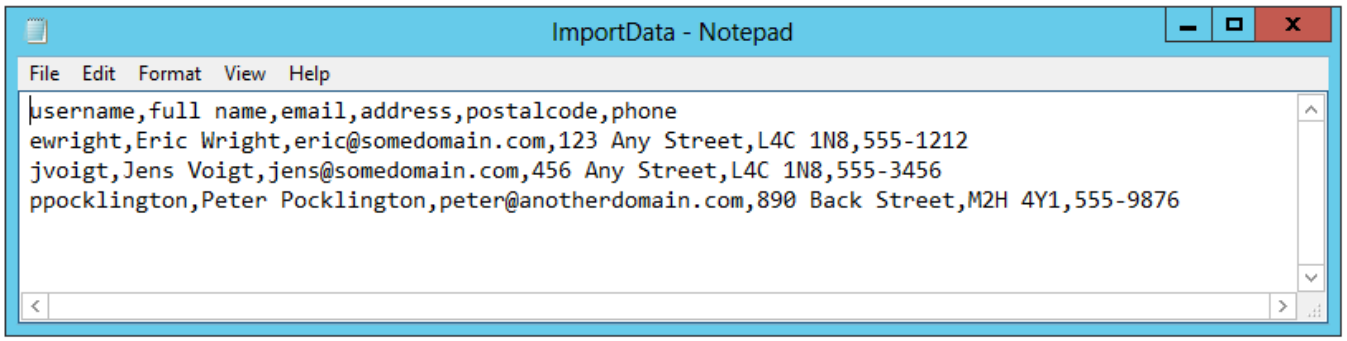
The file now contains our header row. Yay!



Now we read our header-less file (NoHeader.CSV) using the **Get-Content** CmdLet and pipe it into an **Add-Content** CmdLet to append to our **ImportData.CSV** file:

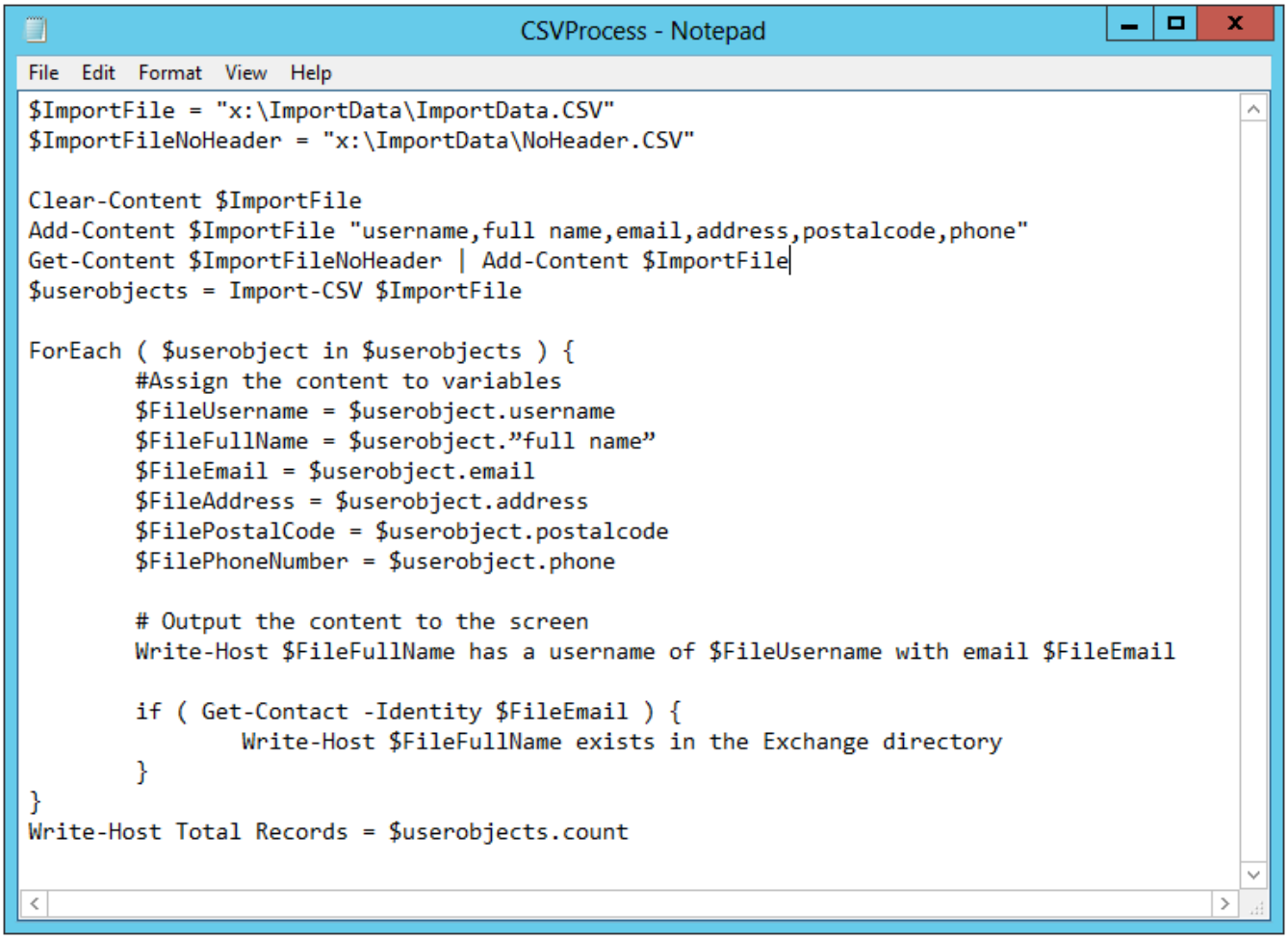
```
Get-Content $ImportFileNoHeader | Add-Content $ImportFile
```

Yup, it’s just that easy. And here is our new file to be used for the contact management section of the script:



So we now have our full process defined to add the header data, import the CSV data into the new

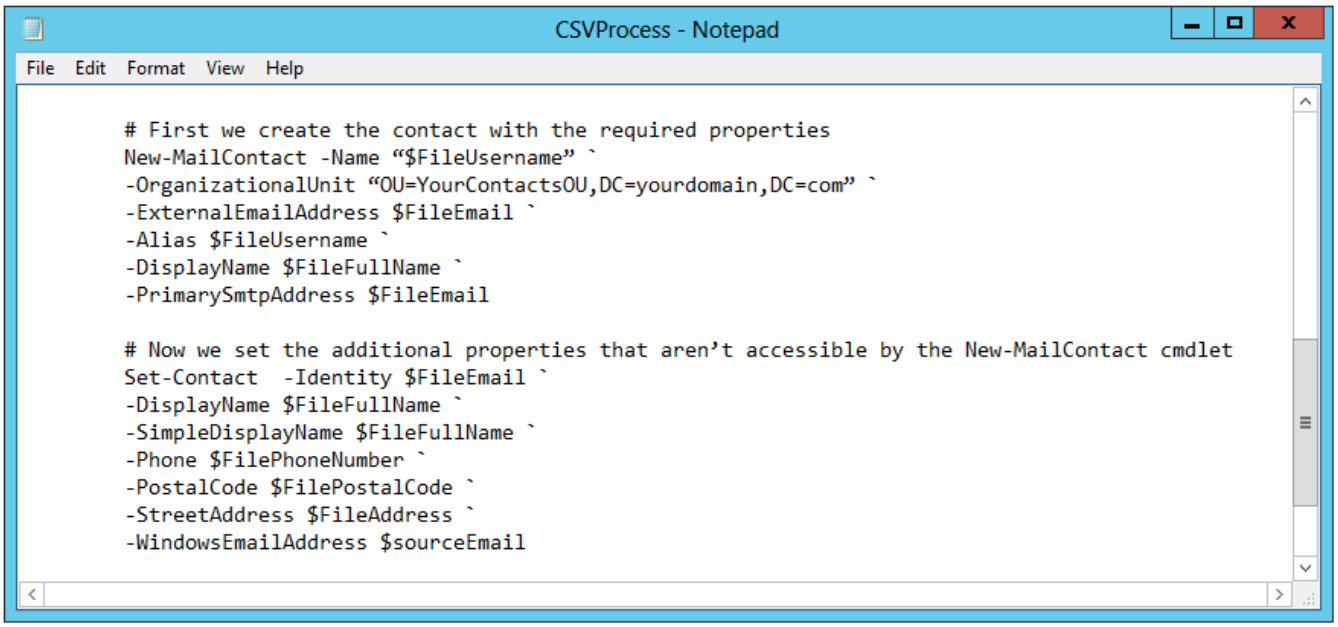
import file, import the data into an array and loop through to use the Get-Contact process to check if they exist.



Adding Contacts if they Don’t Exist, Adding a Line to CSV File

Adding Exchange contacts is easily done (sort of) by using the **New-MailContact** CmdLet but there is a catch: Only some field can be added using the **New-MailContact** CmdLet and the rest have to be set after creation using the **Set-MailContact** CmdLet. Confused? Don’t worry, it takes a little getting used to, but this becomes second nature once you build the process once.

I’m not going to dive into the features of the **New-MailContact** and **Set-MailContact** section here, so I’m doing the old Martha Stewart trick by putting the raw turkey in the top oven, and showing you the cooked one a minute later from the bottom oven. Here is the cooked version of the contact creation section of the script:



So we have now reached the end of the quick 3 part series on using the **Import-CSV** CmdLet and how CSV data is easy to manage and manipulate for using in other processes.

If you want to really dive deep, you can go over to my [Importing, and Updating Exchange 2010 Contacts from CSV] post which is probably the most popular overall post on the site. Hope that this has helped and if you have any questions, drop me a a comment below!

Here is the final copy of the whole script so that you can see it from end to end. Happy Scripting!

```
$ImportFile = "x:ImportDataImportData.CSV"
$ImportFileNoHeader = "x:ImportDataNoHeader.CSV"

Clear-Content $ImportFile
Add-Content $ImportFile "username,full name,email,address,postalcode,phone"
Get-Content $ImportFileNoHeader | Add-Content $ImportFile
$userobjects = Import-CSV $ImportFile

ForEach ( $userobject in $userobjects ) {
#Assign the content to variables
$FileUsername = $userobject.username
$FileFullName = $userobject."full name"
$FileEmail = $userobject.email
$FileAddress = $userobject.address
$FilePostalCode = $userobject.postalcode
$FilePhoneNumber = $userobject.phone

# Output the content to the screen
Write-Host $FileFullName has a username of $FileUsername with email $FileEmail

if ( Get-Contact -Identity $FileEmail ) {
Write-Host $FileFullName exists in the Exchange directory
}

# First we create the contact with the required properties
```

```
New-MailContact -Name “$FileUsername” `
-OrganizationalUnit “OU=YourContactsOU,DC=yourdomain,DC=com” `
-ExternalEmailAddress $FileEmail `
-Alias $FileUsername `
-DisplayName $FileFullName `
-PrimarySmtpAddress $FileEmail

# Now we set the additional properties that aren’t accessible by the New-MailContact cmdlet
Set-Contact -Identity $FileEmail `
-DisplayName $FileFullName `
-SimpleDisplayName $FileFullName `
-Phone $FilePhoneNumber `
-PostalCode $FilePostalCode `
-StreetAddress $FileAddress `
-WindowsEmailAddress $sourceEmail
}
Write-Host Total Records = $userobjects.count
```

This post is part of a series on using the PowerShell Import-CSV Command.

Part 1 – Importing a CSV Into PowerShell with Import-CSV

Part 2 – Looping when Importing a CSV Into PowerShell with Import-CSV foreach

Part 3 – Importing a CSV Into PowerShell with Import-CSV and Adding Headers