# Are you sure? Using the -WhatIf and -Confirm parameters in PowerShell

There are 2 very helpful parameters you can apply to many PowerShell CmdLets. These are the –**WhatIf** and the –**Confirm** options. There are many cases where you are processing a number of records which you would like to get an idea on how to "test drive" the process on a large chunk of data.

First, by utilizing the –**WhatIf** parameter you can preview the changes that will take place against the object without actually committing those changes. This is a great way to ensure that your process or script will give you the expected results.

The second very handy parameter is –**Confirm** which will pass information to the script where it prompts for confirmation before proceeding with the command. You will find that many, if not most, delete commands have this option attached so that a runaway delete process doesn't just blindly wipe out data.

The –**WhatIf** switch is self-explanatory. Here is my data, so What If I run a command against it? In the following script, I'm pushing some variables into a **Set-Contact** CmdLet for Exchange 2010. I want to ensure that the updates will go without issue, so I tag the command with the –**WhatIf** option and the result will show me that the command with either pass and perform updates, or if there is an issue, it will throw an error.

> **Set-Contact -Identity $sourceEmail -City $sourceAddressCity -Company $sourceBranchLegalName -CountryOrRegion $sourceAddressCountry -WhatIf**

Now we want to see a delete process, which in PowerShell are almost always "Remove" CmdLets. On a side note, I would definitely prefer that PowerShell followed the CRUD or Create Read Update Delete convention which we see referenced in application and web development. But I digress, so let's get back to the task at hand.

I want to remove a contact record which matches a variable which I've named $sourceEmail that I'm reading from somewhere. The TechNet help (http://technet.microsoft.com/en-us/library/bb123561.aspx) on the **Remove-MailContact** CmdLet shows the following description for the –**Confirm** parameter:

> The *Confirm* switch causes the command to pause processing and requires you to acknowledge what the command will do before processing continues. You don't have to specify a value with the *Confirm* switch.

So if we read this correctly, the default behaviour of the CmdLet is to *not* prompt for confirmation, but if you want it to prompt you before taking action you need to add the –**Confirm** to your command. So we run the command as-is so that it runs without interaction like this…right?:

> **Remove-MailContact -Identity "$sourceEmail"**

But here is the catch! The command still prompts you despite the fact that you have not explicitly required confirmation using the very information that Microsoft themselves have provided. As in the

magic world, this is the Turn where we specifically ask the CmdLet to prompt, but we pass data to it within the command:

**Remove-MailContact -Identity "$sourceEmail" -Confirm:$Y -WhatIf**

The result of this command is that we explicitly prompt for the confirmation, and the **$Y** sends a **Y** character to the prompt, thus answering the nagging question at the console for us.

I hope that you find these two little parameters to be friendly and helpful. Most importantly, make sure that you always test your processes in a non-production environment first for the best possible level of safety before you put them into production.

If only everything in life came with a -WhatIf option.