Get-Started: ii Captain! Using the Invoke-Item CmdLet

A nifty little PowerShell CmdLet to make use of is **Invoke-Item**, which also has the alias **ii**. This CmdLet can be used against files, URLs and other documents and items to invoke the default action against them.

An example where we would want to use this is where we generate some output into a file such as a .txt file, and then we want it to be rendered to the screen using the editor for the .txt extension which is most likely Notepad.exe on your system.

In this case, let's use a simple one-liner to output the path variable ($env:path) to a text file.

```
$env:path | Out-File MyPath.txt
```

Now that we have the file created, we simply run the Invoke-Item against the file to launch it using the default Open action based on the file type association.

```
Invoke-Item .MyPath.txt
```

The result of the **Invoke-Item** command is that you will see a Notepad windows launch (or whatever your associated application happens to be).

This was a very simple example, but the concept is identical for any registered file type in your system. When running as an interactive script, a very popular purpose for the **Invoke-Item** CmdLet is to launch Internet Explorer to view an HTML file which was created by some other process.

While this is a handy little tool in our PowerShell toolbox, you may find that you want to move to the **Invoke-Expression** if you require more complex command lines and parameters to be available. I'll be putting together a post on that CmdLet in the future which will go into much further detail.

The CmdLet supports relative and literal path references and requires double-quotes for long file and path names where spaces are in the path.

The Get-Help for the Invoke-Item is as follows:

The full TechNet help for the Invoke-Item can be found here:
http://technet.microsoft.com/en-us/library/hh849794.aspx

Happy invoking!

Get-Started: PowerShell 3.0 – Mastering parameters with the Show-Command CmdLet

Welcome to the first PowerShell 3.0 post in the Get-Started learning series. Today we are going to work with PowerShell 3.0 and the **Show-Command** CmdLet. Because this is a PowerShell 3.0 CmdLet I am assuming that you are working with Windows 8, Windows Server 2012 or that you have installed the PowerShell CTP for your existing environment.

The **Show-Command** CmdLet is new with PowerShell 3.0 and what it brings us is a really great way to explore and extend our understanding of parameters.

For our example we will use the Out-FileCmdLet as our target. In your PowerShell console you simply type ***Show-Command <CmdLet name here>***

```
Show-Command Out-File
```

This will bring up a GUI window that presents all of the parameters you can edit for this particular CmdLet. It is context sensitive so the parameters will change depending on which CmdLet you launch.

This is what we see with our **Out-File** CmdLet:

We can see that for **Out-File** we can use the following parameters:

- Append
- Confirm
- Encoding
- Force
- InputObject
- NoClobber
- WhatIf
- Width

You would find these same parameters listed if you were to use the ***Get-Help Out-File*** command also, but for many people it can be confusing to understand the help format and it is especially helpful for those who are used to working with the GUI.

One extra feature we have is that not only are the CmdLet specific parameters shown, but there are also Common Parameters available. These include:

- Debug

- ErrorAction
- ErrorVariable
- OutBuffer
- OutVariable
- Verbose
- WarningAction
- WarningVariable

Common Parameters are those which apply to all CmdLets so these will be more familiar once you get used to using them. You may not need to use them, but it is good to explore them as your skill level increases. They will add great features to your scripts when you begin to work with error handling.

In this case I will only add a couple of parameters which are ***-Append, -NoClobber*** and my file path which is ***c:tempMyOutputFile.txt***.

Once we populate our parameters we can click on the **Run** button to launch the full command. But before we do that, you may also notice that there is a **Copy** button. It is exactly what you think it is, which is a way to copy the full command to the clipboard for pasting elsewhere.

Once we click the **Copy** button we can now paste the output into your editor of choice to use for a script file. I'm a big fan of Notepad myself so this is where I will paste my result to see what the full command looks like.

What I recommend is that you use the **Show-Command** CmdLet to explore what parameters can be used for your various CmdLets and then eventually you will build your comfort and understanding of the capabilities and options of each one for more advanced scripting.

One note about the **Show-Command** is that because it is a GUI, you can only use it on the console and not in a remote PowerShell session (PSSession).

I recommend that you go through as many CmdLets with the **Show-Command** CmdLet now and get more and more experience with how to get the most of each and every CmdLet you use.

Happy scripting!

Get-Started – The DiscoPosse PowerShell 101 series

With so much focus on PowerShell for systems administrators today, one thing I wanted to do was to bring scripting to the non-scripters. What I have found quite often is that many of today's new

administrators have not had any exposure to shell commands and scripting because they have not had a need to during early training.

This is why I am introducing the DiscoPosse Get-Started series. These posts will be geared towards those who are new to using scripting and the PowerShell language.

Since I have come up from a command line world with DOS and Linux and moved into Microsoft Windows afterwards, the jump back to dabble in the command shell was an easy transition. My goal with this series is to bring that comfortability with the shell to those who are new to it.

 Nearly every task that can be performed in the GUI can be done with a PowerShell CmdLet or script. Even more importantly, some things can be done in the PowerShell console that cannot be done in the GUI.

We are going to start with simple walk throughs with the CmdLets and understanding aspects of the PowerShell language which will ramp up your ability and comfort with the language and its capabilities.

Thank you for coming here to learn, and I hope that you enjoy the Get-Started series!

Get-Started Series Article List (links will be added as new articles are online)

PowerShell 3.0 Series

- **Get-Started: PowerShell 3.0 – Mastering parameters with the Show-Command**

PowerShell 2.0 Series

- **Getting Help – Using Get-Command and Get-Help (COMING SOON!)**