Microsoft | TechNet

The Windows PowerShell Profile

This is your guide to getting started with Windows PowerShell. Read through these pages to get familiar with Windows PowerShell, and soon you'll be driving around like a pro.

On This Page

Windows PowerShell Profile

You must admit, Windows PowerShell has a great profile. You hadn't noticed? Start Windows PowerShell and stand to the side of your monitor. See it now? True, you can't see much of anything on the monitor anymore, but you must be able to see that Windows PowerShell looks great from the side. Title bar not too big, coloring about right, smooth face … yep, Windows PowerShell has a great profile. You don't see any of the slight imperfections you might see while looking at it head-on.

So why would we want to mess with something like that? Well, we wouldn't. Instead we're going to explain how to set up and change a different type of profile, a profile that gives you a tremendous amount of control over your Windows PowerShell experience. This profile will help you smooth out some of those head-on imperfections - sorry, *beauty marks* - you might have noticed.

Before we get started, stop staring at the side of your monitor and sit back down in front of it. Sorry, but it really will be easier to follow along this way.

Top of page

The Profile

So what is this profile we're talking about? In the simplest sense, it's a text file. Somewhat less simple, it's a Windows PowerShell script file (a file with a .ps1 file extension). So what makes this a profile rather than just a script file? Location, location, location. Oh, and name. And - well, we'll get to all that in a moment.

The Windows PowerShell profile is simply a script file that runs when Windows PowerShell starts up. You can put cmdlets, scripts, functions - any valid Windows PowerShell commands - into this script file. Each time you start Windows PowerShell, this script file will run. That means you can use the profile to set up your Windows PowerShell environment. Typically that would be custom console settings and aliases, but use your imagination and you can come up with other things you'd like to customize in PowerShell before you start working with it.

Top of page

Finding the Profile

We mentioned location (three times, which should give you some idea how important that is). What makes the profile a profile and not a regular script file is the name and location of the file. Type this at your PowerShell command prompt:

```
$profile
```

That built-in variable will return something like this for Windows XP:

```
C:\Documents and Settings\keymyer\my documents\WindowsPowerShell\Microsoft.PowerShell_profile.ps1
```

And this for Windows Vista and Windows Server 2008:

```
C:\Users\kenmyer\Documents\WindowsPowerShell\Microsoft.PowerShell_profile.ps1
```

This is the full path to the file that Windows PowerShell will try to run when it starts. Notice we said "try" to run. Here's an interesting fact: just because you were able to find the profile doesn't mean it actually exists. $profile is simply a built-in variable that contains the full path to where the profile will be if there is one; the file doesn't actually have to exist, and, by default. it doesn't. If you want a profile to run when you start Windows PowerShell, you need to create this file.

Top of page

Creating a Profile

Before we decide to create a profile, let's check to see whether we already have one:

```
Test-Path $profile
```

If the profile exists this command will return True; if it doesn't exist, the command will return False. If this command returns False, you need to create the profile.

Creating a profile in Windows XP is really easy. Simply type this at the command prompt:

```
notepad $profile
```

This command opens the profile in Notepad. If the profile doesn't exist, you'll be prompted to create it. If you choose **Yes**, the file will be created for you and will be opened in Notepad. Remember, though, we said this is for Windows XP. If you're running Windows Vista or Windows Server 2008 things get a little more complicated. (But just a little bit, not very much.) If the full path to this file doesn't exist, trying to open it in Notepad will return an error; you won't receive any prompts and the file won't be created for you. The way to create this file in Windows Vista and Windows Server 2008 (an approach which will also work in Windows XP) is to use the New-Item cmdlet:

```
New-Item -path $profile -type file -force
```

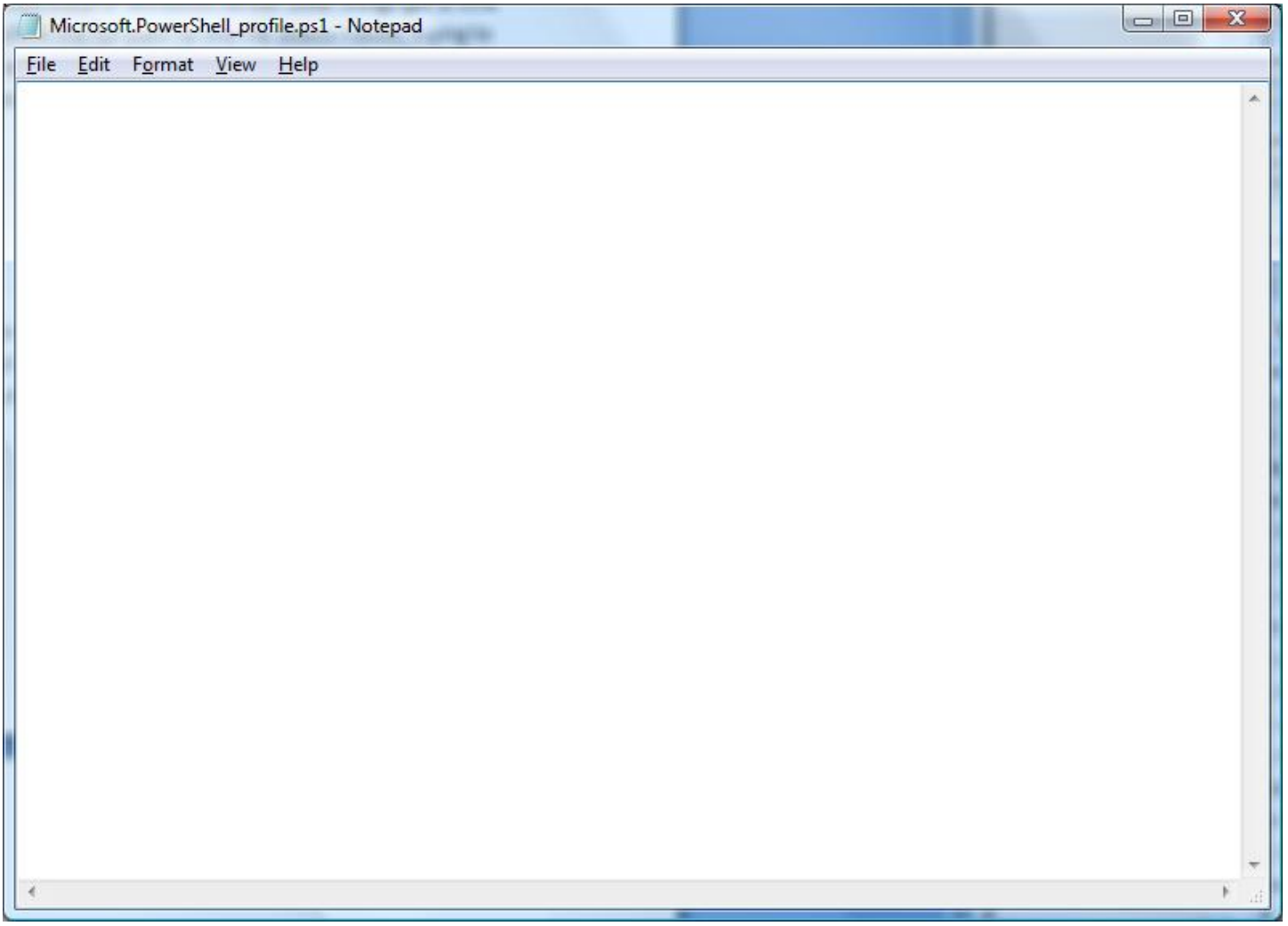
We pass three parameters to New-Item:

- **-path $profile** We're passing the full path, stored in the $profile variable, of the item we want to create.
- **-type file** This tells New-Item what type of item we're creating, in this case a file.
- **-force** This parameter tells New-Item to create the full path and file no matter what.

Now you can open the profile and take a look:

```
notepad $profile
```

And what will you see? Notepad:



Top of page

What Goes in the Profile?

There are a lot of different things you can put in your profile; we're going to do a couple of simple things to show you how it works. You might notice when you start Windows PowerShell that you always start out in the same folder. For example, on Windows Vista, by default, you start your PowerShell session in your user folder:

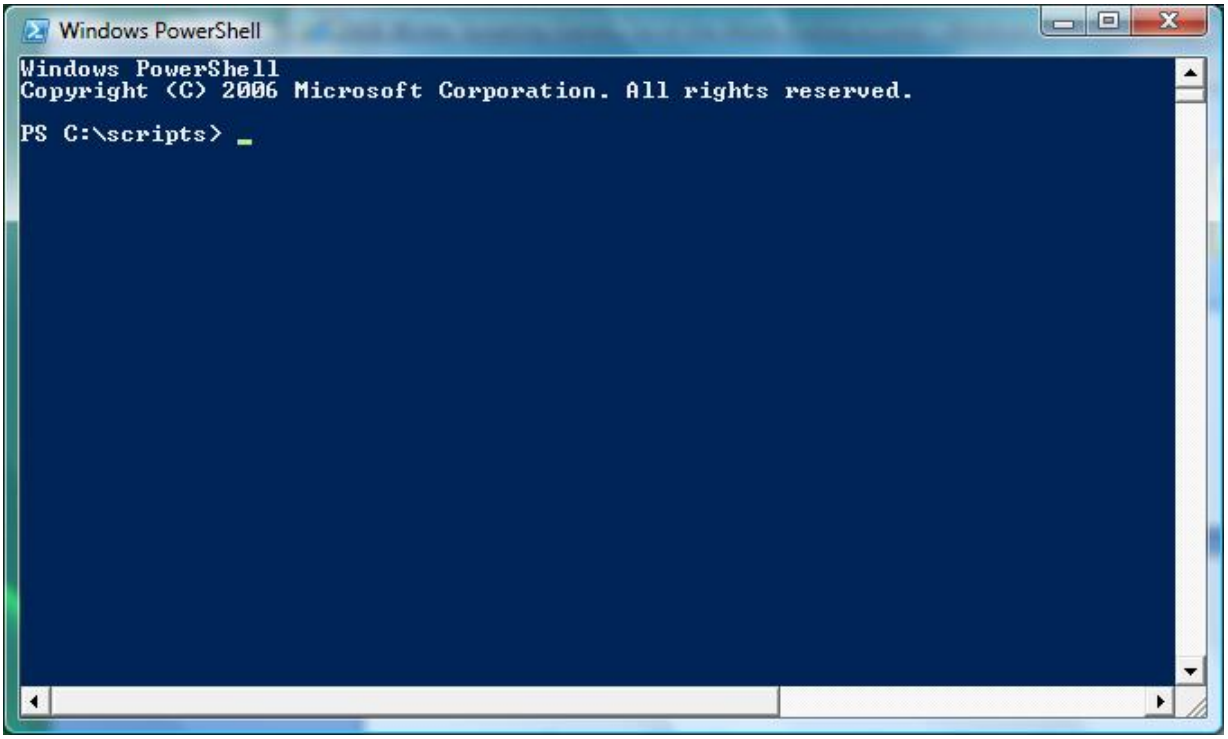
```
PS C:\Users\kenmyer>
```

That's fine, but suppose this isn't the folder you usually work in; maybe you usually work in your C:\Scripts folder. That means that, every single time you start Windows PowerShell, the first thing you have to do is change directories to the C:\Scripts folder. Not a big deal, but kind of a hassle. Well, then why not set up your profile to start you out in the C:\Scripts folder in the first place? Give the following a try:

If your profile isn't open, open it now. (Remember, we just showed you how to do that.) Inside your profile type this:

```
Set-Location C:\Scripts
```

Save the profile and close Windows PowerShell. Now open PowerShell again. Voila!



Now every time you open PowerShell, you'll go straight to your C:\Scripts folder. You can also do things such as set aliases and run functions within your profile. Not only that, but it's simple to set up all these things on other computers; just copy the commands from your profile to the profile on the other machine.

For an example of other things you can do with profiles, take a look at **Windows PowerShell Aliases**[1] or **Customizing the Windows PowerShell Console**[2] in this Owner's Manual.

If you'd like to go back to looking at Windows PowerShell in profile, go right ahead. But you'll probably find it much more useful to remain head-on and work with the Windows PowerShell profile file.

Top of page

Windows PowerShell Owner's Manual

- Getting Started with Windows PowerShell[3]
- Windows PowerShell Shortcut Keys[4]
- Piping and the Pipeline[5]
- Running Windows PowerShell Scripts[6]
- Windows PowerShell Aliases[1]

Top of page

Links Table

[1]*http://technet.microsoft.com/en-us/library/ee692685.aspx*

[2]*http://technet.microsoft.com/en-us/library/ee156814.aspx*

[3]*http://technet.microsoft.com/en-us/library/ee177003.aspx*

[4]*http://technet.microsoft.com/en-us/library/ee176868.aspx*

[5]*http://technet.microsoft.com/en-us/library/ee176927.aspx*

[6]*http://technet.microsoft.com/en-us/library/ee176949.aspx*

Community Content