Mike Nelson – SVD Article – due September 24, 2010

Creating an I.T. Admin's Powershell Toolbox

Remember a time when doing repetitive tasks as an I.T. Admin on Windows servers meant breaking out your DOS batch scripting skills? How about not too long ago when you had to get the syntax just right for a VB Script to do just what you wanted it to in a Scheduled Task? I for one am thankful that the ways of old have been replaced with the means of new and Powershell is now a staple in my tool chest.

Powershell has been embraced by several hardware and software manufacturers as the tool of choice for installation, configuration, and reporting in their products, and is in fact, sometimes the only means of administration for some. Powershell has been around for awhile in version 1, but with the release of Version 2 being a standard fixture in Server 2008 R2 and Windows 7, it is gaining a lot of momentum as the scripting tool of choice. There has even been a project in the works to create an Open Source Implementation of Powershell, adding the capability of scripting in other Operating Systems including Linux, which could open up a whole new set of possibilities. While I embrace the theory behind this initiative, I was also born and raised on Perl and Bash shell scripting, and I find it difficult to think they could be subsidized or replaced.

On the Windows and API front, I have become a believer of Powershell and the flexibility and power it can harness for the I.T. Admin. Using the architecture of cmdlets and aliases, and many other features that are in other popular scripting languages, many if not all of my administrative and reporting duties can really be done inside the proverbial "single pane of glass". Contrary to common perception though, the pane of glass is a command line one, not a GUI one, as many have become accustom to. But, I for one am grateful for the nostalgia of the CLI.

Now, given that I prefer to use the CLI (or ISE as it's been named), I do admit that on occasion I do appreciate the convenience of a GUI for more elaborate editing and function creation. A good Powershell Editor should be your core pick for your Essential Powershell Toolbox. The one that I use is PowerGUI from Quest Software. I have also looked at other GUI editors, such as Primalscript from Sapien and PowerShell Plus from Idera, but I do like the community feel of PowerGUI and have had very good luck with it (Quest does also sell PowerGUI Pro, which is commercial and adds some features, but not enough for me to buy). I am also a huge fan of the vEcoShell Initiative from Vizioncore. Scott Herold has created a fantastic .NET interface that echoes the one of PowerGUI, including the addition of PowerPacks and Libraries. For those cases where I have to get an editor up and running quickly without my laptop or performing an install, I prefer to use Notepad++, which I have handy on my Toolbox USB stick. This editor is great for any type of language scripting, including Bash, Perl, and VB Script.

Another set of tools that should be included are the SDK and API's provided by vendors to allow you to add their cmdlets to your environment. The most used ones for me are the PowerCLI from VMware, the Active Directory Management from Quest Software, and the Citrix XenApp and XenServer SDK's. Many other enterprise application vendor have created cmdlets or support Powershell calls via their API's and you should check to see if any apply to you and your daily administration and management tasks.

What will be, or should be, taking up the most space in your toolbox is the code itself. Creating and maintaining collections of code snippets and functions is quick and easy, but organizing them in such a way that you can easily refer to them and know what they do is another. Don't get sidetracked by collecting as much code as possible. Instead, take the time to collect useful code and put them in your own repository, creating an index of sorts with comments and descriptions of the code and functions. Personally, while it may not be the best way to do it, I keep an index of my created and collected code in

an OneNote Notebook on my laptop. You should also not forget that there are several repositories on the Internet that are there at a click of the mouse to hopefully find what your looking for and grab it for immediate use. The largest and best of these that I have found are Microsoft's Script Library (of course) and the POSHCode Repository.

One last important thing for any Admin's Toolbox is to keep the tools in that toolbox dynamic. Software has gotten to the point that it is changing at light speed, with updates and patches making more headlines than the new releases. It's pretty much a fact that in our daily work lives, the lives of an I.T. Admin, we really need 26 hours in a day to get most everything done that we need to get done (notice I didn't say "want" to get done). I think that in most cases for me, Powershell has knocked off those extra hours of work time, allowing me to use it for what I call more productive personal "barley beverage" time ;-)