



Global Knowledge™

Expert Reference Series of White Papers

Windows PowerShell: Why Developers Should Care

Windows PowerShell: Why Developers Should Care

Thomas Lee, Chief Architect for Microsoft, Global Knowledge EMEA, MVP, MCSE, MCT, MOF/MSF Trainer



Introduction

Windows PowerShell is Microsoft's next generation scripting and shell tool. Recently released, PowerShell is also part of Exchange 2007 and a host of other Microsoft Management products. While IT Pros are going to love PowerShell – why should developers? As an MVP on PowerShell, I'd like to spend some time explaining what this product is and why it's relevant.

This paper is written to accompany a talk I gave to Developers recently and is based on earlier articles I've written as well as on the presentations made by the PowerShell Team over the past three years.

Background

Microsoft is solving some key management problems with PowerShell. To manage a network today, Administrators face a range of challenges with respect to tools:

- **Range:** Admins are required to be experts in a huge range of tools. That's because, thus far, no one tool does everything.
- **The Command Line:** UNIX admins advocate using command line tools, but Windows tools for the command line have lagged behind the GUI ever since the earliest versions of Windows in the 1980s!
- **GUI Scalability:** a GUI is a great tool for performing a single operation on just one computer, but it can be slow and tedious if you need to perform that same operation on 1,000 systems. When using GUI tools such as MMC to administer any large object, such as using Active Directory Users and Computers MMC Snap-in to manage Active Directory, you have to retrieve information from the store in order to populate the GUI. This can mean a performance hit to manage a system.
- **Power:** Admins can often find themselves using one tool (e.g., CMD.Exe and batch scripts), only to find they can't quite do what they want and have to switch to another tool (e.g., VBscript or C++/C#).

Admins want a single tool that is consistent in usage, and with both broad reach and great depth. Such a tool should enable access to any object simply and straightforwardly. Finally, the tool should focus on the Administrator – and not on the developer.

PowerShell is Microsoft's answer to these problems. The goal is to provide:

- A rich command shell environment (such as provided by the Korn, Bourne, or other shells on UNIX/Linux)
- A rich programming language (on a par with Perl or Ruby)

- A tool that has wide scope everything
- Richness and the rigorousness of the .NET Framework

As Jeffrey Snover, PowerShell's architect, points out "PowerShell does the heavy lifting so the admin doesn't have to." PowerShell is a vast subject, and I can only scratch the surface in this white paper. I provide some links to more information later

What Is PowerShell?

PowerShell is a new shell, aka PowerShell.Exe, that incorporates a new language. As released, PowerShell is a Windows component issued effectively as a hot fix that you can download from the web at <http://tinyurl.com/yk5h8l>.

In operation, PowerShell makes use of small commands, known as cmdlets (pronounced command-lets) that you can combine in rich and powerful ways to create a first-class administration environment. You can use these commands individually or combine them with a UNIX-like pipeline.

The Basics of PowerShell

To launch the new shell, just run PowerShell.EXE. At first glance, it appears similar to CMD.EXE but don't let that fool you. Once running, you can use PowerShell like you would Cmd.exe or, for that matter, Bash on UNIX/Linux as an interactive shell. You can also create and run scripts to provide rich administration.

Some of the key features of the POWERSHELL include:

- **Cmdlets:** small executables, written in a .NET language and that implements a standard interface. Cmdlet are named using a 'verb-noun' format (e.g., Get-Help or Export-Csv). Cmdlets can take parameters or take input from the pipeline. Cmdlets are task-oriented and should be self-documenting.
- **Variables:** you can create variables, assign values to them, and use them in calculations, method calls, etc. Variables can be scalars or arrays.
- **Providers:** most applications have some sort of data store - a shared store such as Active Directory or your own unique data store held in SQLServer. Providers in PowerShell allow data to be presented from different data stores in a consistent fashion to existing cmdlets. This enables you to surface data that existing cmdlets can leverage. There are 7 providers built into PowerShell (Alias, Environment, File System, Function, Registry, Variable, and Certificate).
- **Pipeline:** UNIX scripting shells use the concept of a pipeline. This is extremely powerful, but suffers from the disadvantage that, in most cases, the pipelined data is just raw text. This means prayer-based parsing. Rather than passing text, PowerShell passes .NET objects. That means a cmdlet can use .NET reflection to look inside what is getting passed, and know what's being passed (aka the end to prayer-based parsing!).

More, More, More

PowerShell is one of those products where I find myself constantly saying "Oh, and then there's this other really cool feature" over and over and over again! In these few pages, I've just begun to explore what PowerShell can do. And I've not explained what a super .NET toybox it is – useful to help developers get a better understanding of .NET.

But What About Developers?

PowerShell is clearly aimed at, and is already a great product for, Admins and IT Professionals. But why would a developer care – indeed should a developer care at all? Let's look at why Developers should be thinking about PowerShell.

One important aspect for any C# developer is what the PowerShell calls the glide path. PowerShell is built directly on top of, and leverages extensively, the .NET Framework. You can use PowerShell to explore the objects that your C# code is planning to manipulate as a sort of rapid application development environment for your code.

PowerShell provides direct access to the .NET reflection capability, which can simplify and assist in the design and debugging. You can try out your code in PowerShell, and use reflection easily to find problems with your code. Converting the PowerShell back to C# is relatively simple (if you know C#). The developers on the PowerShell team make use of this – as should you.

The second aspect is that your application needs to be managed. No matter how good your app, if the IT Pros in your organization can't manage it, your app is most likely toast. One of the key aspects of most development methodologies is ensuring you get operational buy-in. Developing PowerShell Cmdlets or Providers can enable admins to see inside and to manage your application. By delivering these features as part of your application, you make your applications much easier to write and more Admin-friendly. Besides, they are not all that hard to write, since most of the heavy lifting is done for you by PowerShell.

Finally, a simple point: Writing PowerShell Cmdlets and providers, if you consider it early enough in your design, should make for less work overall. Writing a PowerShell cmdlet is significantly easier than designing an MMC snap-in, for example.

Call to Action

Your call to action is:

- Download PowerShell RTM and then install and use it.
- Remove any references to cmd.exe on your computer and use PowerShell instead.
- Work through the getting started document.
- Start reading [microsoft.public.windows.powershell](http://microsoft.public.windows.powershell@msnews.microsoft.com) at msnews.microsoft.com.
- Develop basic cmdlets, providers, and scripts – and share them.
- Join us on this long strange trip that is PowerShell!

Resources

There are several PowerShell resources you can use today:

- The newsgroup microsoft.public.windows.server.scripting at msnews.microsoft.com
- <http://www.reskit.net/PowerShell> pointers to more information about PowerShell
- <http://www.reskit.net/Monad/samplescripts.htm> – this is a sample scripts page
- <http://del.icio.us/tfl/blog+PowerShell> – a list of PowerShell or PowerShell-focused blogs
- <http://www.technorati.com/search/PowerShell> – this is a feed of PowerShell blog posts

- Author's personal technical blog – Under the Stairs
<http://tfl09.blogspot.com>
- Author's corporate blog – The Chief Architect's corner
<http://cacorner.blogspot.com>

Learn More

Learn more about how you can improve productivity, enhance efficiency, and sharpen your competitive edge. Check out our comprehensive list of Microsoft courses at www.globalknowledge.com or call 1-800-COURSES to speak with a sales representative.

Our courses and enhanced, hands-on labs offer practical skills and tips that you can immediately put to use. Our expert instructors draw upon their experiences to help you understand key concepts and how to apply them to your specific work situation. Choose from our more than 700 courses, delivered through Classrooms, e-Learning, and On-site sessions, to meet your IT and management training needs.

Summary

PowerShell is the future of scripting and administrative management on the Windows platform. As an admin, you need to understand it and start using it. As a developer, you should embrace it!

About the Author

Thomas F. Lee is currently the Chief Architect for Microsoft at Global Knowledge EMEA (Europe, Middle East, and Africa). As Chief Architect, Thomas is responsible for ensuring consistent approaches to training across EMEA. He holds a BS in Computer Problem Solving from Carnegie Mellon University, and did a year of post-graduate research there. He is a Microsoft Certified Trainer, with experience in MSF, MOF, Security, .NET for IT Pros, and Windows (from Windows 2 through Vista and Longhorn). Certifications include MVP, MCSE, MCT MSF Trainer, and certified Desktop Support Technician. Thomas was recently named as a Microsoft Most Valued Professional (MVP) for the twelfth year in succession. The latest MVP Award makes Lee one of longest-serving MVPs in the world.

Thomas continues to do some teaching, but also writes. He is a co-author of several books on Windows-related topics, TCP/IP, and query languages. He is also a regular speaker at Microsoft events and a regular columnist in Server Management magazine and PC Pro. He is a Fellow of the British Computer Society, and a member of the Big-8 Management Board.