PowerShell: The Gift That Keeps On Giving to Microsoft Exchange Server

I realize that many administrators who work with Microsoft Exchange Server might not share my enthusiasm for PowerShell. After all, isn't revisiting the command line something that makes one think that you're coping with the piping and scripting loved by UNIX and Linux geeks? Well, Windows PowerShell's command-line nature is there for all to see, and PowerShell is awfully fond of piping (and gets a lot of its power from this ability). Also, there's no doubt that scripting is something that PowerShell devotees spend a lot of their time discussing.

But all of this is missing the point. The reason I think PowerShell has made such a contribution to Exchange is simple: PowerShell provides the ability to automate common administrative tasks quickly, simply, and accurately in a way that even the best-designed GUI-based management console will never be able to do.

The Exchange development group took the momentous decision from Exchange Server 2007 onward to encapsulate the business logic that drives the product around its very large set of 600+ PowerShell cmdlets. In passing, let me say that "cmdlet" is one of my least favorite technical terms. I would much prefer the simplicity of the term"command" instead. However, cmdlet is the term coined by PowerShell developers—and who am I to debate the wisdom of their choice?

Moving away from my bias against cmdlet (only the term, not the implementation), the decision taken for Exchange 2007 let developers build Exchange administrative interfaces on a common foundation, meaning that the Exchange Management Console (EMC), the setup program, and the Exchange Management Shell (EMS) all execute the same code. This eliminated the inconsistency seen in previous versions of Exchange and let Microsoft remove redundant and overlapping code.

The central role of PowerShell was further expanded in Exchange Server 2010 with the addition of the Exchange Control Panel (ECP), which leverages the same platform as EMC and EMS. Exchange 2010 also ties its Role-Based Access Control (RBAC) mechanism to its PowerShell cmdlets, defining roles in terms of the cmdlets that a holder of a role can execute. Indeed, the granularity is such that RBAC lets roles define the level of parameters to a cmdlet that a user can execute, meaning that users can retrieve details of their mailboxes (running the Get-Mailbox cmdlet behind the scenes) and set some properties (with Set-Mailbox), whereas administrators can do a lot more.

Exchange 2010 also includes remote PowerShell that lets administrators run PowerShell to manage remote Exchange servers from workstations and other computers. Remote PowerShell forces users to connect via IIS and be authenticated to build a session that connects to Exchange. The session includes details of the cmdlets and parameters that the user is entitled to use as defined by the RBAC roles that they hold. Collectively, these components provide the ability to connect to Exchange Online running in Office 365 to perform management using the EMC, ECP, or EMS. Indeed, as explained in "How to Manage Your Exchange 2010 Organization with PowerShell Implicit Reporting over the Internet" (http://bit.ly/mRz6x4), it's even possible to emulate the connections used to connect with PowerShell to manage Office 365 by establishing external connections from the Internet to manage on-premises Exchange 2010 servers (protected of course by a reverse proxy).

There's no doubt that the move to comprehensively embrace PowerShell was a stunning and far-reaching decision that no other major Microsoft server product emulated for several years. Indeed, it's only recently that elements of Windows Server have supported similar access via PowerShell. Perhaps the biggest breakthrough will occur in Windows 8, because Windows Server 8's GUI-based management console is built on top of PowerShell. The most important points are that Windows Server 8 contains hundreds of new cmdlets to enable management of components from the shell and to eliminate the need for administrators to use GUI consoles. In short, Windows administrators need to get rid of the idea of logging on to a server to do work, something that Exchange administrators have become used to since the advent of remote PowerShell support in Exchange 2010.

In addition, the developer preview of Windows Server 8 includes PowerShell Web Access, another component of PowerShell 3.0. To me, PowerShell Web Access looks very much like a natural development from the ideas proven by Exchange 2010's implementation of remote PowerShell where servers are managed from workstations through PowerShell without the need to log on directly to the target server. The interaction between workstation and target server is managed by a combination of IIS, Active Directory (AD), and Exchange's RBAC security model.

All in all, PowerShell provides Exchange administrators with huge potential for managing their servers in a way that the EMC or ECP just can't deliver. In passing, let me also acknowledge the wisdom of whoever decided to include the function into the EMC that outputs the PowerShell code for the different operations performed through the console; this is a fantastic learning device for anyone who wants to understand the basic syntax and construct for using PowerShell to manage Exchange. If you haven't yet taken the plunge to get down and dirty with PowerShell, you're really missing out on something that can save you scads of time.

—Tony Redmond

InstantDoc ID 140470