SQL Server PRO

EXECUTIVE SUMMARY

Remote Control in Detail: PowerShell Remoting

- "Remoting" with Microsoft PowerShell consolidates all server management into a single port and protocol
- PowerShell is highly configurable
- PowerShell provides multiple security options
- JAMS Scheduler leverages PowerShell for automated enterprise Windows systems management

MARCH 27, 2012

Don Jones, Senior Partner and Principal Technologist, Concentrated Technology LLC
Daniel St. Jean, Senior Support Engineer, MVP Systems Software, Inc.

in partnership with

JAMS
JOB ACCESS & MANAGEMENT SYSTEM

Remote Control in Detail: PowerShell Remoting

Overview

The days of separate applications and multiple protocols to control multiple remote servers are coming to an end. With the introduction of Windows PowerShell v2.0 and its remoting feature, system administrators finally have a flexible, extensible, and secure tool for managing any number of remote machines. In addition, PowerShell works with all versions of Windows back to Windows XP. Because this is such a key technology going forward, system administrators are advised to learn more about its installation, capabilities, and security options.

With a large set of plug-in tools, JAMS Scheduler builds on native PowerShell capabilities to automate job management across the Windows enterprise.

Context

During the webinar, Mr. Jones explained and demonstrated a number of key PowerShell functions. Mr. St. Jean described how the JAMS architecture leverages and amplifies PowerShell.

Key Takeaways

"Remoting" with Microsoft PowerShell consolidates all server management into a single port and protocol. Fully integrated with the .NET Framework, PowerShell is an application that runs on top of the open-standard WS-MAN (Web Services for Management) protocol in conjunction with the WinRM (Windows Remote Management) service, which functions as a "listener" (service) and "talker" (client) to remote machines. Because PowerShell uses standard Web HTTP (and optionally HTTPS) encryption, it is easy to create tunnels to reach firewalled or otherwise restricted or perimeter servers. Previews of the Windows 8 operating system show that Microsoft is systematically extending PowerShell to support all of its management products. Rather than using a separate Server Manager protocol to communicate in Windows 8 Server, for example, system administrators are migrating to PowerShell.

> *Remoting is definitely Microsoft's way forward for management. It's worth your time to start playing with it now.*
>
> Don Jones

The most straightforward way to establish PowerShell remoting on a machine that will be receiving incoming connections is to run the command "Enable-PSRemoting." In a single-domain, HTTP environment, this is generally all that an administrator needs to do. The command performs a series of setup tasks:

- Enables and starts the WinRM service
- Registers PowerShell as an endpoint
- Runs the "SetWSManQuickConfig" command
- Creates an HTTP listener

- Enables an exception in the Windows Firewall

In other circumstances, when an administrator needs to cross domains, configure with GPO (Group Policy Objects), or connect to a proxy server, a Help file (command "help about_remote_troubleshooting) provides comprehensive instructions.

To connect with a particular computer, administrators must use the computer name as it appears in Active Directory; the IP addresses or DNS alias won't work. Again, the troubleshooting Help file provides WinRM workarounds.

Once PowerShell is installed, the "Enter-PSSession" command starts a one-to-one interactive session with a remote computer and opens a command line on that machine. Various command arguments enable the administrator to specify a computer name, provide an alternate credential, specify a different configuration name or authentication protocol, or force the use of SSL (if set up as an HTTPS listener). Once the command is executed, all processes are implemented directly on the remote machine.

However, PowerShell is more useful in one-to-many remoting. This mode uses the "Invoke" cmdlet along with one or more computer names to cause those machines to (1) start their copies of PowerShell, (2) run a command, (3) encode the results into XML, and (4) transmit them back to the administrator's machine, whose copy of PowerShell then (5) decodes the results back into objects and (6) puts them in the pipeline. The "Invoke" command's syntax allows the administrator, among other things, to run a group of commands contained in a script in a local file path, set the authentication method, and specify an alternate port. By default, the command limits connections to 32 machines because each connection requires its own PowerShell session; however, the "ThrottleLimit" argument permits this to be overridden.

Although this generally works without problems, PowerShell has a PSDiagnostics module that initiates a "PSWSManCombinedTrace." The administrator can compare the trace results (contained in an operational log) to those of a properly functioning machine to troubleshoot any remoting issues.

PowerShell is highly configurable.

Unlike more limited traditional remote tools, PowerShell comes with an extensive library of built-in commands ("cmdlets") and scripts that can be combined for sophisticated, automated, programmatic operation. These commands use a consistent interface and parser, so the learning curve is faster.

Each individual application, including PowerShell, registers with WinRM as a network "endpoint," with specified capabilities, commands, and scripts. In addition, it is possible for PowerShell to register multiple endpoints, each with its own configuration. In Microsoft Office365, for example, customers can talk to their part of the cloud, but their commands are limited to what Microsoft has provisioned at its WinRM endpoint.

Even from a fairly high level—the WSMan local drive—a system administrator can configure options such as "MaxTimeoutsms" and "MaxBatchRequests." A WinRM listener

can also be configured to listen to specific IP addresses on specified ports.

The administrator can also allow machines to register themselves as endpoints instead of having to do it manually. This slightly reduces security in exchange for convenience and more automated operation.

In the Group Policy Management Editor, the system manager can specify, for example, how many administrators can get in at once and how much memory a remote user can use. Furthermore, developers can create custom versions of PowerShell with defined capabilities: who is allowed access, how many machines can be connected simultaneously, and many other options.

PowerShell provides multiple security options.

When a system administrator makes an outgoing connection over WinRM, his or her login credentials are delegated to the remote computer, which is a native function of the default Kerberos Active Directory authentication protocol. From the network's standpoint, it's as if the administrator walked up to the computer, logged in, and performed particular tasks as him or herself, not as a generic SuperUser. In this "security-transparent" approach, nothing is added or removed from the native Windows security layers, and all organizational audit trails are kept intact.

However, this authentication is only good for one "hop" (unless the administrator has remoted into an Active Directory domain controller, which has its own authentication routines). When the administrator needs to jump from the client machine to a staging server to a production server, for example, he or she must enable the CredSSP protocol, which is turned off by default. He or she then has multiple-hop capability.

WinRM sets itself up on HTTP (port 5985 by default); to enable HTTPS-level encryption, the administrator must configure WinRM to listen on the default port 5986. (These port assignments can be changed to any free port.) Alternatively, PowerShell and other applications can provide their own encryption, typically using a static pre-shared key applied to the traffic before handing it off to WinRM.

JAMS Scheduler leverages PowerShell for automated enterprise Windows systems management.

JAMS (Job Access and Management System) Enterprise Job Scheduler, from MVP Systems Software, automates and adds to PowerShell's native capabilities. JAMS Scheduler includes a PowerShell snap-in that contains more than 50 PowerShell cmdlets that simplify the manipulation and control of JAMS via PowerShell. The snap-in's PowerShell Provider also exposes the JAMS object hierarchy for easily moving and managing JAMS objects. Compared with PowerShell alone, JAMS' custom PowerShell host offers superior error control, parameter passing,

JAMS automates PowerShell and PowerShell makes JAMS completely automatable.

Daniel St. Jean

Remote Control in Detail: PowerShell Remoting

host-to-script communications, and detached user-interface capabilities.

The JAMS architecture comprises three components:

- **JAMS Scheduler**, the "brains" of the operation.
- **JAMS Agents**, which actually perform the tasks.
- **JAMS Clients**, which provide access to and control of the schedule.

JAMS displays a dashboard where administrators can view and control scheduled jobs running on the network. For example, an administrator can release a job from its dependencies or access specific job parameters. The schedule itself can be viewed either in an expandable tree format or graphically within a Gantt chart.

JAMS also ties into Active Directory groups for securing JAMS clients and provides enterprise-level controls for managing jobs, whether that involves job history, version control, JAMS Agent dispatch, time zone support, or cellphone notifications.

Resources

PowerShell.com features an ask-the-experts forum in which Don Jones participates, as well as a free "Administrators Guide to PowerShell Remoting."

Visit JAMSScheduler.com to learn more about the product and download an evaluation copy.

Don Jones

Don Jones is one of the world's leading experts on the Microsoft business technology platform. He is the author of more than 35 books, and is a top-rated speaker at technology conferences worldwide. He writes features and monthly columns for numerous print and online publications, is a multiple-year recipient of Microsoft's prestigious Most Valuable Professional (MVP) Award, and serves as Editor-in-Chief for Realtime Publishers.

Daniel St. Jean

Daniel St. Jean is a Senior Support Engineer at MVP Systems Software and brings more than 15 years of enterprise systems management expertise to the customers he supports. He has extensive knowledge of Windows, Linux, and UNIX, and has worked with a wide range of applications like SAP, JDE, Oracle, and MS Dynamics. His primary responsibilities include support for new customers as well as assisting onsite with enterprise engagements.