Windows PowerShell is a shell and scripting language used by many IT professionals. Very often they get caught up in preconceptions and misinterpretations, usually based on prior experience with scripting or development. In this session we will explore the 10 most common mistakes and traps people fall into with PowerShell, and how to avoid them.

PowerShell Remoting

Martin Schwartzman
Senior Premier Field Engineer
maschvar@microsoft.com
Why remote commands?
In the old days...

- PsExec and WMI were almost the only tools allowing remote execution
  - With the downside: The new process on the remote machine cannot be controlled from the machine where the process was spawned
What isn’t PowerShell Remoting?

- The `-ComputerName` parameter does not rely on PowerShell Remoting.
- They use distributed COM (DCOM) or remote procedure call (RPC) to connect to the remote systems.
- For example:
  - `Get-WmiObject`
  - `Get-HotFix`
  - `Get-Process`
  - `Get-EventLog`
What is PowerShell Remoting?

- A PowerShell feature that allows remote management from a central location
- Based on WinRM 2 (an implementation of WS-Man)
- Adapts the Universal Code Execution Model (whatever runs locally should run anywhere)
- There are many different styles of remoting (interactive, fan-out, fan-in, implicit)
What do I need?

- PowerShell v2.0 (v3.0 or above for some features)
- The user needs to be in the local administrators group (for the default session configurations)
- The network location must be private or domain
- Remoting needs to be enabled (It is disabled by default)
- As of Windows Server 2012, PowerShell Remoting is enabled by default and is mandatory for server management
Under the hood

- WinRM / WS-Man is based on HTTP and a single port (5985)
  - 5986 for HTTPS
- Objects are serialized into XML streams
CIM vs. WMI

- WMI requires DCOM connectivity
  - TCP 135 & TCP 1024+
- CIM requires less complex network connectivity
  - TCP 5985
- Open platform WSMan standard (OS Agnostic)

- Get-WmiObject Win32_BIOS -ComputerName DC01
- Get-CimInstance Win32_BIOS -ComputerName DC01
Getting started

- Enable-PSRemoting
- winrm quickconfig
- Computer Configuration -> Policies -> Administrative Templates -> Windows Components
  - Windows Remote Management
  - Windows Remote Shell
I don’t want to be me

- Cmdlets in PowerShell do not accept credentials as strings (username and password). They expect a PSCredential object.
- This object can either be obtained using the cmdlet `Get-Credential`, or using `New-Object System.Management.Automation.PSCredential`.
- Credentials can also be saved to disk or a database using the data protection API `Export-CliXml / Import-CliXml`. 
When do I use it?

- **1:Many (fan-out): Large Scale Automation**
  - Send the script to remote machines
  - Throttling – limits the number of concurrent operations

- **1:1 (interactive): Secure Telnet Replacement**
  - Cmdline equivalent of Remote Desktop
  - Interact with a remote machine as if it were local

- **Many:1 (fan-in): Delegated Administration & Hosting**
  - No tools installation required on client
  - Constrained session environment (cmdlets, parameters, language)
One at a time

- **Invoke-Command**
- Can be used against one or many remote computers
- Script blocks can be executed remotely as well as scripts. The result can be treated as coming from the local machine
- Scripts do not need to be on the remote machine or on a share. WinRM copies the script in the background
Keep it running

- New-PSSession
- Create a persistent session object
- Persistent sessions can be used to prevent the creation of a new runspace every time
- Pass the session object using the -Session parameter
I want to be there

- Enter-PSSession
- Enter the previously created session, or create one on the fly
- For some operations determine the lifetime of a session is desired
  - v2.0: If the job controller loses the connection the session is destroyed and running scripts are stopped
  - v3.0 and above: Sessions can be disconnected and reconnected even from another computer
Be there, here

- **Import-PSSession**
- Brings the remote commands to the local session
- Can import cmdlets that do not exist on the local computer
- Managing different technologies (SharePoint, Exchange, Active Directory) from a single computer without the need of installing the management tools on various machines
Where are the tweaks?

- Use the WSMan: PSDrive to navigate through the configuration settings
  - dir WSMan:\localhost\Service -Recurse
  - Set-Item WSMan:\localhost\Shell\MaxShellsPerUser 25
  - Set-Item WSMan:\localhost\listener\*\Port 8888
What about non-admins?

- By default, only local administrators can use remote PowerShell
- `Set-PSSessionConfiguration` can be used to change the permission using the UI or SDDL
- The information can be also changed directly in the registry (XML)
- To transform SDDL into something readable and vice versa, the `CommonSecurityDescriptor` class can be used
PowerShell Web Access

- Acts as a Windows PowerShell gateway, providing a web-based PowerShell console
- Increases the value of your investment in PowerShell
- Built for phones, tablets
- Cross-platform support
Q & A
Resources

- Layman’s Guide to PowerShell 2.0 remoting:
  http://www.ravichaganti.com/blog/?page_id=1301

- Administrator’s Guide to Windows PowerShell Remoting:
  http://powershell.com/cs/media/p/4908.aspx

- Secrets of PowerShell Remoting: