The Scripting Games

Server Inventory

By Richard Siddaway
Winter Scripting Games 2014

Dr Scripto is dancing a jig and looking very unhappy. He's dancing because he's received a big increase in his budget. He's unhappy because he's been told he has to spend at least some of it modernising the server estate of the company his organization has just acquired.

That organization was so badly run that the IT staff don't really know what they are managing. If Mrs Scripto wasn't listening Dr. Scripto would be using rude words.

Dr Scripto has decided that you need an opportunity to learn more about using PowerShell and he has just the job for you. You need to create a mechanism to determine the servers that exist in this new part of the organization. Your first task to create a method of scanning an IP address range – it could be a full Class C address or a smaller subnet. The subnet will be provided to you in the form 10.10.10.0/24. Dr Scripto needs to know if there is a machine associated with the IP address and its operating system, and service pack. The results – positive and negative – need to be recorded for future use – preferably in a CSV file or other readable format. It is important that unused IP addresses are recorded

Once the initial scan is produced Dr Scripto want to be able to investigate specific machines or groups of machines. Those investigations should return, one or more, the following sets of data:

- Hardware information including manufacturer, model, CPU, RAM and disk sizes (only local disks are required).
- The date of the last hotfix applied to the machine and last reboot time.
- If any of the following are installed:
  - IIS
  - SQL Server
  - Exchange
  - SharePoint
- Installed Windows components

The set of data to be returned should be selectable. The data needs to be saved for future analysis.

All file names should include the date of production and be descriptive of their contents.

Assume that you have permissions to access all of the machines on the network and that all required firewall ports are open. If you can test your preferred access mechanism and retrieve the required data in another way Dr Scripto will be pleased.

As a final requirement DR Scripto would like to be able to take a file of data and produce a report – including graphical representation of the data where possible. If those graphs could be imported in PowerPoint Dr Scripto would be extremely grateful.

Your code should be production ready with:

- Ability to optionally report on progress
- Full error checking, reporting and handling
- Ability to accept pipeline input where appropriate
- Help is available
- Input is validated

The code should be portable so that it can be reused if similar situations occur in the future or the inventory needs to be repeated for auditing or other purposes. The investigative code should be expandable so that the ability to return additional sets of data can be built into the code as requirements change

In your entry submission, include a transcript that shows you running the command as described in this scenario.

CONCENTRATED TECH

Finally! Public PowerShell Classes from Don Jones and Jason Helmick!

Get the schedule from
http://events.concentratedtech.com

SAPIEN POWERSHELL TOOLS

SAPIEN PowerShell STUDIO™

PrimalScript®

iPowerShell™ Pro

TO LEARN MORE GO TO www.sapien.com

Key Criteria

These are some of the main items our judges will consider. You do not need to meet all key criteria, but you may earn extra points for doing so. This list is intended as a summary, and does not override the specifications of the scenario above.

- Consider the practices in *The Community Book of PowerShell Practices* (linked at http://powershell.org/wp/newsletter)
- Avoid aliases, except for –Object cmdlets; avoid positional parameters and truncated parameter names.
- Use appropriate error handling.
- Use appropriate means of displaying output, progress messages, errors, etc.
- When appropriate, manage pipeline input correctly
- When appropriate, validate input via parameter validation attributes
- Provide help for all scripts and functions, including examples
- Script filenames should include production date for versioning
- Use modular programming practices to maximize opportunities to share code
- Use appropriate remote connectivity protocol(s), including, where appropriate, failover to backup protocols
- Produce appropriate CSV file for initial scan; other formats including XML are permitted.
- Input is subnet, not a range
- Report includes IP addresses with no response as well as responses
- Service pack reported (or report no service pack)
- Investigative routines can accept input from pipeline, or single machines
- Investigative routines are modular, which may include them being separate scripts
- Graphical reports available
- PowerPoint file including graphs is produced

As is often the case in Windows PowerShell, there will be many ways to complete these objectives. In most cases, judges will prefer approaches that:

- Perform well under the load specified
- Leverage built-in functionality of Windows PowerShell rather than reinventing the wheel
- Are the most straightforward and easy to read and understand

PolicyPak
WHAT YOU SET IS WHAT THEY GET.
Got Java? Got Firefox? Got Flash?
Smackdown pop-ups and lock'em down.
MVP Jeremy Moskowitz demos secret toolkit
www.PolicyPak.com