Monitoring Support

By Richard Siddaway  
Winter Scripting Games 2014

Dr. Scripto needs your help. He is rolling out a monitoring solution and was promised some assistance but has been let down. He needs you to:

* Create a monitoring xml configuration file for each server from the data supplied in servers.csv. A sample file is part of the download package. The items that have to be monitored may be different between servers and not all items will be present for all servers
* Copy that xml file to the appropriate server – path c:\drsmonitoring
  + If the path doesn’t exist it should be created
  + Store the config file on the local disk in c:\monitoringfiles
* Set registry value. Key - HKLM:\SOFTWARE\DRSmonitoing. The registry value Monitoring under that key should be set to 1. The registry key may or may not already exist.

The code should be able to perform bulk actions or work for a single server. The file copy and registry setting should be generic so that functionality can be re-used for other projects.

Dr. Scripto needs the following information gathering so that he can report on progress:

* Servers where the registry key existed and was set correctly
* Servers where the registry key existed and was set incorrectly
* Servers where the registry key had to be created
* Servers that have had the monitoring config file installed

Dr.Scripto would like you to output the data as an object that can be optionally stored on disk. The object should have its own type and formatting (table and list). The data should be presented as a single report including server name and the data the test was performed. He would also like an html based report.

As a final option Dr. Scripto would like the ability to compare the configuration file on a remote server with the copy you created from the CSV file. This will test for unwanted changes. If a new file is rolled out it should overwrite the existing file in both the local store and the remote server.

Assume that you have permissions to the remote machines.

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

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

## 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
* Test file system and registry paths
* Report contains information as specified (server, if configuration file is installed, registry key exists and is correct, date and time, etc)
* Produce report as HTML
* Appropriate use of parameters and default values
* Registry keys are configured as specified
* Configuration file produced from CSV input, not hardcoded

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