

Acl, Cacl, Toil and Trouble

By Jeff Wouters  
Winter Scripting Games 2014

No – that’s the Scottish play that Dr Scripto is directing.

Dr Scripto has a little job for you to do and he knows you’re just going to enjoy it. A file server migration project is happening in the organization and a whole new folder structure is being created. Your job is to set the initial permissions on the folders. Dr Scripto wants to test this for an individual department.

Dr Scripto is keen to make this department a model for the rest of the organization so he wants you to automate the setting of permissions.

The folder structure will look like this:

Department\_name

Department\_Open folder

Department\_team1

Team Shared folder

Team Private folder

Team Lead folder

Department\_team2

Team Shared folder

Team Private folder

Team Lead folder

Etc. depending on the number of teams in the department. Each of the folders in the structure above can have subfolders created by the relevant team.

The permissions are to be applied to groups only. For testing purposes Dr Scripto wants you to concentrate on the Finance team which has these groups:

* Finance – contains all members of the finance department
* Receipts – members of receipts team
* Receipts\_lead – members of receipts team leadership
* Payments –members of payments team
* Payments\_lead –members of payments team leadership
* Accounting –members of accounting team
* Accounting\_lead –members of accounting team leadership
* Auditing –members of audit team
* Auditing\_lead –members of audit team leadership

All groups are nested as appropriate.

***For the purposes of this event the groups can be Active Directory-based or local.***

The following permissions must be applied:

* The department open folder must be readable by the whole organization but only members of the department can write, create or delete content. This is the only folder that non-members of the department can access – they should be denied access to all the other folders belonging to the department
* Each team shared folder can be read by everyone in the department but only members of the relevant team can write, create or delete content.
* Each team private folder can only be accessed by the relevant team – they have read, write, create and delete permissions
* The team lead folders are only accessible by the appropriate team leadership group
* Members of the audit can read the contents of any folder for any department

Each department will have management team that should their own folder that is only accessible by that team.

All permissions must be set so that subfolders inherit the settings where appropriate. The inheritance and propagation rules must be configurable for each folder

For testing purposes Dr Scripto would like to be able to create a folder structure and apply the permissions prior to full roll out. The full permission set applied to each folder must be recorded for future needs. Any files must include the date and time they were created in the name.

Dr. Scripto is aware that people like to interfere in the administration of his servers so he wants you to create a routine to analyse the folder structure of a department and compare the permissions that are currently set on those folders with those recorded earlier. The analysis must test every sub-folder in the department’s folder irrespective of the length of the path. A dated report in HTML format must be produced showing the correct and actual permissions. Other formats can be produced if your solution requires it but the final report must be in HTML format.

Ideally, Dr Scripto would like another routine that would correct permissions but only amend those folders that need it.

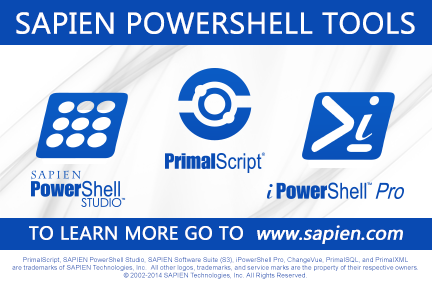
You have all required permissions to access any remote machines if required. All protocols required for accessing machines remotely are enabled across the server estate.

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.

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
* Hardcoding of values, as specified, is permitted
* Create a modular folder hierarchy existence/creation routine
* Paths are parameterized
* Permissions accepted from pipeline as specified
* Applied permissions are stored in a file
* Ability to report on permissions; report is HTML
* Long paths (>260 characters) supported
* Permissions stored in other format(s) as specified
* Permissions at individual folder level can be corrected

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

