**PROfit Sunday Restart script - description**

1. **Script usage**
   1. **Running the script** - When the PROfitSundayRestarts.ps1 file it expects an environment parameter to be passed to it e.g. PROfitSundayRestarts.ps1 -env {EnvParameter}. In case it is run manually without the parameter it will prompt for it. There must be a configuration for that environment name in the config file .\DATA\Environments.json (see next section for details) or the script will abort.
   2. **Script Configuration**
      1. **.\DATA\Environments.json** – file containing the configuration for all environments the script can be used on – basically the ones we configure with the DSC. Each environment is an element in an array. The format of the elements is as follows:  
         {

"Name": "ST",

"DSCsrv": "sofentpublish.cloudad.local",

"MailSMTP": "sofcas.cloudad.local",

"Mailform": "TechSysAppAdministration@naxex-tech.com",

"Mailto": [

"TechSysAppAdministration@naxex-tech.com"

],

"Mailcc": [

"TechSysAppAdministration@naxex-tech.com"

]

}

* + - 1. **Name** – abbreviation of the environment that will passed to the script. It must be the same as the designation of the environment in the DSC as the script loads the application and server data from the DSC’s configuration files.
      2. **DSCSrv** – the server the DSC is located on. The following files must exist on it:
         1. D:\DSC\Config\_{EnvParameter}.psd1
         2. D:\DSC\{EnvParameter}\applications.csv
      3. **MailSMTP, Mailform, Mailto, Mailcc** – self-explanatory. Mailto and Mailcc are arrays and can take multiple addresses separated by “,” (standard json syntax)
    1. **.\DATA\RestartedServices.csv** – a .csv file describing the servicese to be restarted. The columns are used as follows:
       1. **Component** – Name of the component to be restarted. Must be the same as in the name used in the DSC configuration files.
       2. **Priority** – Defines the order in which the services are restarted. All services with 0 priority are restarted in parallel after the restarts of the ones with higher values are finished.
       3. **HealthCheck** – needed only for web services. This is the Uri of the web request to be made to the service to determine if it is healthy. Check is passed if the response status is 200. In case this column is empty no health-check will be made and the HMStatus in the final report file will be ‘notConfigured’
       4. **HMProtocol** –the protocol used for the health-check request – http, https etc. In case it is not set http will be used by default.
  1. **Script Logging** – the script writes a detailed log file and a final status file which are sent by e-mail when the restarts are complete. They are located in .\Logs.
     1. **Log\_yyyyMMdd.log** – Contains the detail log of the script executions. Since the script uses multiple processes to restart some of the services in parallel the name of the thread the log entry comes from is written after the date for each logging message. When reading the log its best to find the thread you need to check and then use find next in the text editor you are using to follow it.
     2. **Status\_yyyyMMdd.txt** – Contains the status of each of the restarted services after all restarts are finished.
     3. **E-mail Notifications** – Each time the script is run a [START] e-mail is sent to the recipients in the environment config. When the restarts finish an [END] e-mail is sent containing the Log and Status files.

1. **Script logic**
   1. **Building a list of services to restart** – the script reads the RestartedServices.csv file and the DSC config files – Config\_{EnvParameter}.psd1 and applications.csv. Then it compiles a list of components to restart whose names are present in both RestartedServices.csv and the DSC (live and demo type for each component are considered separate entries).  
      Weather a component is a windows service or a web application/site is determined based on the node information in the DSC it’s installed on. Then the “service control name” of each component is created by adding “. AppPool” to the name for each one that is web (same as the DSC does) or reading the service name from the applications.csv for the windows services.
   2. **Restart of a web service algorithm** – each web component is restarted by recycling it on each of the servers in order with 90 sec. pause between each restart.
   3. **Script’s restart algorithm:**
      1. Restarts all services with priority in RestartedServices.csv higher than 0 in order (highest value goes first). In case this includes web services they are restarted according to 2.
      2. Start parallel thread for each web service with priority 0 to restart it according to the algorithm in point 2.
      3. Restart all the windows services with priority 0 consequentially in the main thread waiting for 90 secs between each. After this is done the main thread waits for all the threads in b) to finish in case they are still running.