Iron Scripter 2018: Prequel 10

The puzzle

Greetings Iron Scripters. This is your last challenge before Iron Scripter.

In this weeks challenge you're required to:

- List all event logs that contain 1 or more records
- Determine the percentage of the log size that has been used
- If the log is over 80% full:
    - Ensure that the log is set to over write old records as required.
    - Increase the maximum log size by 10%
- Create a log file of your changes
- Your code should not display any errors when run.
- Your code should display all logs that have entries in them.
- Your code should display any hidden logs that contain entries in them.
- You should display the complete log name, and the number of entries in the log.
- The number of entries in the logs should be displayed in descending order (the log with the most entries in it should appear on the first line of the output).

Use best practice if it doesn't conflict with your faction's aims. The solution must be acceptable to your faction:

- Daybreak Faction - beautiful code
- Flawless Faction - flawless code
- Battle Faction - good enough to get the job done

Good luck and good coding.

The Commentary

This puzzle revolves around the event logs. To display the event log information, you may be tempted to do this:

```
Get-EventLog -List
```

While you'll get a display similar to this:

```
PS> Get-EventLog -List

  Max(K) Retain OverflowAction       Entries Log
  ------ ------ --------------       ------- ---
  20,480      0 OverwriteAsNeeded     38,781 Application
  20,480      0 OverwriteAsNeeded          0 HardwareEvents
     512      7 OverwriteOlder             0 Internet Explorer
  20,480      0 OverwriteAsNeeded          0 Key Management Service
     128      0 OverwriteAsNeeded        298 OAlerts
     512      7 OverwriteOlder             0 PiALog
  20,480      0 OverwriteAsNeeded     25,419 Security
  20,480      0 OverwriteAsNeeded     15,449 System
  16,384      0 OverwriteAsNeeded         38 TechSmith
  15,360      0 OverwriteAsNeeded      8,086 Windows PowerShell
```

It's only part of the story. The event logs above are the "traditional" event logs. Another type of event log was introduced with Windows Vista. For those you need to use:

```
Get-WinEvent -ListLog *
```

In which case you'll get this (truncated) view:

```
PS> Get-WinEvent -ListLog *

LogMode   MaximumSizeInBytes RecordCount LogName
-------   ------------------ ----------- -------
Circular           20971520       38781 Application
Circular           20971520           0 HardwareEvents
Circular            1052672           0 Internet Explorer
Circular           20971520           0 Key Management Service
Circular            1052672         298 OAlerts
Circular            1052672           0 PiALog
Circular           20971520       25419 Security
Circular           20971520       15449 System
Circular           16777216          38 TechSmith
Circular           15728640        8086 Windows PowerShell
Circular            1052672           0 AMSI/Operational
Circular           20971520             ForwardedEvents
Circular           10485760           0 Microsoft-AppV-Client/Admin
Circular           10485760           0 Microsoft-AppV-Client/Operational
Circular           10485760           0 Microsoft-AppV-Client/Virtual Applications
Circular            1052672        2397 Microsoft-Client-Licensing-Platform/Admin
Circular            1052672           0 Microsoft-User Experience Virtualization-Ag...
Circular            1052672           0 Microsoft-User Experience Virtualization-Ap...
Circular            1052672           0 Microsoft-User Experience Virtualization-IP...
Circular            1052672           0 Microsoft-User Experience Virtualization-SQ...
Circular            1052672           1 Microsoft-Windows-AAD/Operational
Circular            1052672           0 Microsoft-Windows-All-User-Install-Agent/Admin
Circular            1052672           0 Microsoft-Windows-AllJoyn/Operational
Circular            1052672           0 Microsoft-Windows-AppHost/Admin
```

You get the traditional event logs and much more. A total of 440 logs on my Windows 10 system. The logs you see depend on the features you've enabled.

We only want event logs with at least 1 record:

```
Get-WinEvent -ListLog * |
where RecordCount -ge 1
```

The next step is to determine the percentage of the log that has been used. If you examine an individual log:

```
Get-WinEvent -ListLog  'Microsoft-Windows-PowerShell/Operational'  |
where RecordCount -ge 1 |
Format-List *
```

```
PS> Get-WinEvent -ListLog  'Microsoft-Windows-PowerShell/Operational'  |
where RecordCount -ge 1 |
Format-List *



FileSize                         : 15732736
IsLogFull                        : False
LastAccessTime                   : 26/10/2017 11:48:39
LastWriteTime                    : 15/03/2018 11:00:08
OldestRecordNumber               : 15656
RecordCount                      : 2828
LogName                          : Microsoft-Windows-PowerShell/Operational
LogType                          : Operational
LogIsolation                     : Application
IsEnabled                        : True
IsClassicLog                     : False
SecurityDescriptor               : O:BAG:SYD:(A;;0x2;;;S-1-15-2-1)(A;;0x2;;;S-1-15-3-1024
                                   -3153509613-960666767-3724611135-2725662640-12138253-5
                                   43910227-1950414635-4190290187)(A;;0xf0007;;;SY)(A;;0x
                                   7;;;BA)(A;;0x7;;;SO)(A;;0x3;;;IU)(A;;0x3;;;SU)(A;;0x3;
                                   ;;S-1-5-3)(A;;0x3;;;S-1-5-33)(A;;0x1;;;S-1-5-32-573)
LogFilePath                      : %SystemRoot%\System32\Winevt\Logs\Microsoft-Windows-Po
                                   werShell%4Operational.evtx
MaximumSizeInBytes               : 15728640
LogMode                          : Circular
OwningProviderName               : Microsoft-Windows-PowerShell
ProviderNames                    : {Microsoft-Windows-PowerShell}
ProviderLevel                    :
ProviderKeywords                 :
ProviderBufferSize               : 64
ProviderMinimumNumberOfBuffers   : 0
ProviderMaximumNumberOfBuffers   : 64
ProviderLatency                  : 1000
ProviderControlGuid              :
```

You'll see a FileSize and MaximumSizeInBytes. If you notice the file size is slightly bigger than the maximum size in this instance suggesting the log is full! The extra must be an overhead in the log.

To calculate the percentage of the log that is used:

```
Get-WinEvent -ListLog  * |
where RecordCount -ge 1 |
select LogName, LogMode, RecordCount,
@{N='PercentFull'; E={[math]::Floor(($_.FileSize / $_.MaximumSizeInBytes)*100)}}
```

Which gives results like these:

```
LogName                                                                 LogMode RecordCount PercentFull
-------                                                                 ------- ----------- -----------
Application                                                           Circular       38788         100
OAlerts                                                               Circular         298         100
Security                                                              Circular       25421         100
System                                                                Circular       15449          45
TechSmith                                                             Circular          38           0
Windows PowerShell                                                    Circular        8088         100
Microsoft-Client-Licensing-Platform/Admin                             Circular        2400         100
Microsoft-Windows-AAD/Operational                                     Circular           1           6
Microsoft-Windows-Application-Experience/Program-Compatibility-Assistant Circular        4           6
Microsoft-Windows-Application-Experience/Program-Telemetry            Circular        1441         100
Microsoft-Windows-ApplicationResourceManagementSystem/Operational     Circular        1388         100
Microsoft-Windows-AppModel-Runtime/Admin                              Circular        2004         100
Microsoft-Windows-AppReadiness/Admin                                  Circular         571          21
Microsoft-Windows-AppReadiness/Operational                            Circular         257          21
Microsoft-Windows-AppXDeployment/Operational                          Circular        2090         100
Microsoft-Windows-AppXDeploymentServer/Operational                    Circular        5946         100
Microsoft-Windows-AppxPackaging/Operational                           Circular        2007         100
Microsoft-Windows-A
```

I've used the [math]::Floor() method to round the percentage down to the integer portion.

You need to use the -Force parameter to ensure that you get the debug and analytic logs otherwise they're hidden:

```
Get-WinEvent -ListLog  * -Force |
where RecordCount -ge 1 |
select LogName, LogMode, RecordCount,
@{N='PercentFull'; E={[math]::Floor(($_.FileSize / $_.MaximumSizeInBytes)*100)}}
```

There's a requirement to display the logs in descending record number order:

```
Get-WinEvent -ListLog  * -Force |
where RecordCount -ge 1 |
sort RecordCount -Descending |
select LogName, LogMode, RecordCount,
@{N='PercentFull'; E={[math]::Floor(($_.FileSize / $_.MaximumSizeInBytes)*100)}}
```

If the log is more than 80 percent full you were tasked with ensuring the log was set to overwrite old records and to increase the size by 10 percent. Any changes had to be logged.

Let's see what we have to work with in the output of Get-WinEvent -ListLog

```
PS> $l = Get-WinEvent -ListLog 'Microsoft-Windows-HomeGroup Provider Service/Operational'

PS> $l | gm


   TypeName: System.Diagnostics.Eventing.Reader.EventLogConfiguration

Name                            MemberType   Definition
----                            ----------   ----------
Dispose                         Method       void Dispose(), void IDisposable.Dispose()
Equals                          Method       bool Equals(System.Object obj)
GetHashCode                     Method       int GetHashCode()
GetType                         Method       type GetType()
SaveChanges                     Method       void SaveChanges()
ToString                        Method       string ToString()
FileSize                        NoteProperty long FileSize=1052672
IsLogFull                       NoteProperty bool IsLogFull=False
LastAccessTime                  NoteProperty datetime LastAccessTime=26/10/2017 11:35:34
LastWriteTime                   NoteProperty datetime LastWriteTime=15/03/2018 10:20:42
OldestRecordNumber              NoteProperty long OldestRecordNumber=1
RecordCount                     NoteProperty long RecordCount=947
IsClassicLog                    Property     bool IsClassicLog {get;}
IsEnabled                       Property     bool IsEnabled {get;set;}
LogFilePath                     Property     string LogFilePath {get;set;}
LogIsolation                    Property     System.Diagnostics.Eventing.Reader.EventLogIsolation
LogIsolation {get;}
LogMode                         Property     System.Diagnostics.Eventing.Reader.EventLogMode
LogMode {get;set;}
LogName                         Property     string LogName {get;}
LogType                         Property     System.Diagnostics.Eventing.Reader.EventLogType
LogType {get;}
MaximumSizeInBytes              Property     long MaximumSizeInBytes {get;set;}
OwningProviderName              Property     string OwningProviderName {get;}
ProviderBufferSize              Property     System.Nullable[int] ProviderBufferSize {get;}
ProviderControlGuid             Property     System.Nullable[guid] ProviderControlGuid {get;}
ProviderKeywords                Property     System.Nullable[long] ProviderKeywords {get;set;}
ProviderLatency                 Property     System.Nullable[int] ProviderLatency {get;}
ProviderLevel                   Property     System.Nullable[int] ProviderLevel {get;set;}
ProviderMaximumNumberOfBuffers  Property     System.Nullable[int] ProviderMaximumNumberOfBuffers
{get;}
ProviderMinimumNumberOfBuffers  Property     System.Nullable[int] ProviderMinimumNumberOfBuffers
{get;}
ProviderNames                   Property     System.Collections.Generic.IEnumerable[string]
ProviderNames {get;}
SecurityDescriptor              Property     string SecurityDescriptor {get;set;}
```

Changing the logmode involves using the System.Diagnostics.Eventing.Reader.EventLogMode enum:

```
PS> [enum]::GetValues('System.Diagnostics.Eventing.Reader.EventLogMode')
Circular
AutoBackup
Retain
```

The MaximumSizeInBytes is a long so you can multiply by 1.1 to increase the value by 10%. You have to use the SaveChages() method to apply the changes.

NOTE: Adding 10% to the maximum size may not be enough to get the log to below 80% full. You may need to run this a couple of times.

Adding the changes and logging gives this:

```
Get-WinEvent -ListLog  * -Force |
where RecordCount -ge 1 |
sort RecordCount -Descending |
```

```powershell
foreach {
  $percfull = [math]::Floor(($_.FileSize / $_.MaximumSizeInBytes)*100)
  $psitem | select LogName, LogMode, RecordCount, @{N='PercentFull'; E={$percfull}}
  if ($percfull -gt 80){
    if ($_.LogMode -ne 'Circular') {
       $_.LogMode =  [System.Diagnostics.Eventing.Reader.EventLogMode]::Circular
       $_.SaveChanges()
       $outstring = "$(Get-Date) $($_.LogName) set to circular logging"
       Add-Content -Path C:\TestScripts\EventLogChanges.txt -Value $outstring
    }
    $newsize = ($_.MaximumSizeInBytes * 1.1) -as [long]
    $_.MaximumSizeInBytes = $newsize
    $_.SaveChanges()
    $outstring = "$(Get-Date) Increased file size of $($_.LogName) to $newsize"
    Add-Content -Path C:\TestScripts\EventLogChanges.txt -Value $outstring
  }
}
```

I had to move the addition of the Percentfull property inside the foreach because using select before the foreach stripped off the SaveChanges() method!

All that's left is to add the bells, whistles and error handling to ensure that the script won't fail. That sounds like a faction orientated exercise so I'll leave to you as homework before Summit.