SOP Reporting on printer usage

Printer usage information is useful for capacity planning, and possibly budgeting, for your shared printers. By default, printer usage details are unmonitored, but you can turn logging on (and off!). When you enable printer usage monitoring, the Windows Spooler service writes a record to the event log for each print job containing critical usage information. This recipe shows you how to turn on printer usage reporting and shows how to create a function to return printer usage information. This recipe creates a function that returns printer usage information as objects. Objects are easier as they enable you to process the output specifically for your environment-for example, counting the total number of pages printed, reporting on who is using the printer, and so on.

Getting ready

Run this recipe on PSRV where you have already set up a printer. This recipe assumes you have set up a printer. You also need to use the printer a bit to generate some event log entries on which you report, otherwise the recipe may return errors when you try to get event log entries.

How to do it:

1.Run webtutil to turn on printer monitoring:

$log = 'Microsoft-Windows-PrintService'   
 webtutil.exe sl $log /operational /Enabled:true

2.Define a Function:

1.Specify the Function header for an advanced function:

Function Get-PrinterUsage {   
 [CmdletBinding()]   
 Param()

2.Get the events from the PrintService event log:

$Dps = Get-WinEvent -LogName   
 Microsoft-Windows-PrintService/Operational |   
 Where-Object ID -eq 307

3.Create a hash table for each event log record:

Foreach ($Dp in $Dps) {   
 $Document = [Ordered] @{}

4.Populate the hash table with properties from the event log entry:

$Document.Id = $dp.Properties[0].value   
 $Document.Type = $dp.Properties[1].value   
 $Document.User = $dp.Properties[2].value   
 $Document.Computer = $dp.Properties[3].value   
 $Document.Printer = $dp.Properties[4].value   
 $Document.Port = $dp.Properties[5].value   
 $Document.Bytes = $dp.Properties[6].value   
 $Document.Pages = $dp.Properties[7].value

5.Create an object for this printer usage entry:

$UEntry = New-Object -Type PSObject   
 -Property $Document

6.Give it a better type name:

$UEntry.PsTypeNames.Clear()   
 $UEntry.PsTypeNames.Add("Packt.PrintUsage")

7.Output the entry:

$UEntry

} # End of foreach   
 } # End of function

3.Set and use an alias to get the printer usage:

Set-Alias -Name GPRU   
 -Value Get-PrinterUsage   
 GPRU | Format-Table

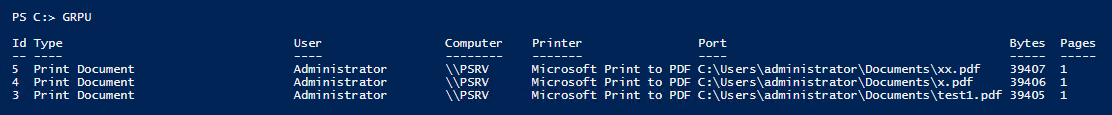
How it works:

In the first step of the recipe, you use the utility wevtutil.exe to tell the Spooler to start recording printer usage details to the event log. Printer usage event logging is not turned on by default, and at present, there is no PowerShell cmdlet to turn on event logging.

In the first sub-step in *step 2*, you create an advanced function by decorating the Param()   
block with the CmdletBinding() attribute. In the second sub-step, you get all the printer   
event log entries that relate to usage reporting (ObjectID 307). In the third sub-step in *step*

*2*, the function iterates through each entry in the log. In the fourth sub-step, for each entry,   
you create a hash table that holds the information returned from the event log. In *sub-step 5* and *sub-step 6*, you create a PSObject for the event log entry and change the object type   
name from PSObject to Packt.PrintUsage. Finally, in *sub-step 7*, you also close out the   
foreach loop and the advanced function.

Finally, in *step 3*, you define an alias for this new function. Then you use the function, via its alias, and pipe the output objects to Format-Table to produce a nice output like this:



There's more:

By creating a function that returns an object for each event log entry, you get significant   
flexibility in using the output of the function. The Get-PrinterUsage function changes the type name of the returned object. With a custom type name, you could create a customized display XML that creates an output that suits your requirements. You can also use the   
objects returned and filter out the usage of specific printers by user. You can also use   
Measure-Object to get the total number of pages printed, the average pages per job, and   
the maximum print job length.