C-MP 4A : Collecting CPU Information – Perfmon

Now that we have discovered our Spooler class, it's about time that we start to collect some useful information about all the instances. The first thing that I would like to collect in the amount of CPU time the service is currently using. This information can be retrieved one of two ways:

1. Performance Monitor (Perfmon)
2. Windows Management Instrumentation (WMI)

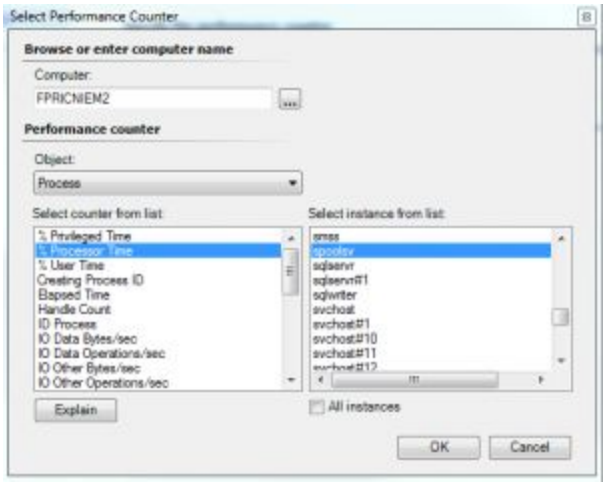
For now I am going to collect this using Perfmon (tomorrow I will use WMI). So the first thing we will need to do is fire up perfmon and have a look around for the counter that meets our needs. I know that the Process counter will contain information about the CPU usage of a process and after looking through the available instances I found that the spooldv represents our Spooler instance on my computer (this is the exe name of the service executable minus the exe portion). This is not a general rule of thumb for looking up processes in the Process counter, but for the Spooler Service it holds true.

Now that we know the counter name and instance name we can open up our authoring console and edit our management pack. Navigate to Health Model -> Rules and right click anywhere, select New -> Collection -> Performance Based -> Windows Performance Collection.

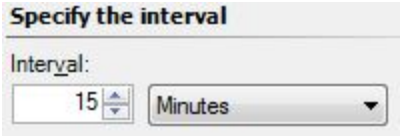
Complete the General screen with the following information:

Rule ID: Richard.CustomMP.Spooler.CPUUsage.Perf.13001
Rule Name: Richard.CustomMP: Spooler CPU Usage (Perf) (13001)
Description: Collects the CPU usage of the spooler service
Target: Richard.CustomMP.Spooler
Category: Performance Collection

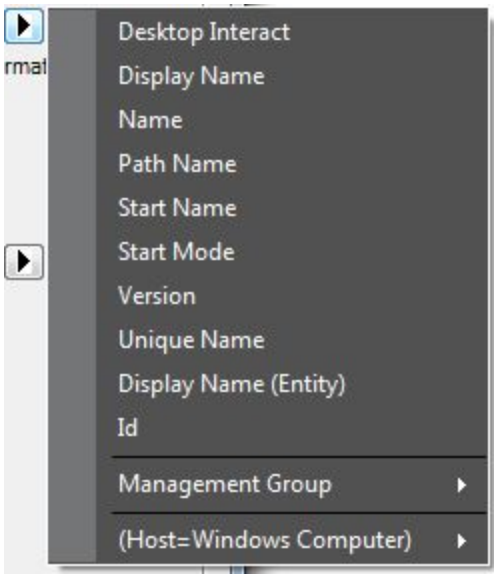
Click Next to continue. On the Performance Counter screen, click the select button and select the spooler service counter from the dialog box that appears.



For the interval, select 5 min.



Click Next and Finish to complete creating your collection rule. As you can see from this example the performance collection was a piece of cake and after 10 min you can start seeing the data brought back into SCOM. A final thing I would like to point out is the “fly outs” that are available when populating the object and instance. SCOM allows you to push properties from your class or the parent of your class (and so on) to these values allowing for a totally dynamic monitoring solution. These could be useful if you needed to watch multiple counters on the same computer.



Unfortunately SCOM does not show 0 100% correctly, but as you can clearly see our collection is working and is collecting against our Richard.CustomMP.Spooler class.

You can grab the latest build of the custom management pack here.