

# Processing Windows performance and capacity data with Captell

Automate reporting, innovate decisions

Captell Developments have recently developed some report templates to cover basic areas of Windows capacity and performance such as CPU utilisation and queue length, disk space utilisation, memory utilisation and network throughput. This data is made available by the Windows operating system (for this article I am using Vista; however, the same applies to Windows Server 2000, 2003 and Windows XP etc.) and is collected and stored by any number of proprietary monitor packages such as Microsoft [SCOM](#) or [MOM](#), [CA Unicenter](#), [Mercury Sitescope](#), [BMC Patrol](#), etc.

For large windows based networks a suitable monitor package is vital for the smooth operation of the environment. Most large sites have some form of alerting in place that brings impending problems to the attention of someone responsible for their resolution. Many also have data collection and storage which can then be fed to a suitable reporting package (hopefully Captell). Our templates have a generic interface that allows the same template package to accept data from many different monitors. This is achieved by creating a Captell table definition that has either pass through SQL or input SQL which manipulates the source data to a consistent form. As the most common form of storing the Windows performance data is as indicated in the following figure we have adopted this as our initial "raw" storage form.

MachineName	ObjectName	CounterName	InstanceName	DateTime	CounterValue
\\CDPL-DEVL-1	LogicalDisk	Avg. Disk Read Queue Length	D:	06-Jul-2007 16:48:26.9100	0
\\CDPL-DEVL-1	LogicalDisk	Avg. Disk sec/Read	D:	06-Jul-2007 16:48:26.9100	0
\\CDPL-DEVL-1	LogicalDisk	% Idle Time	C:	06-Jul-2007 16:48:26.9100	0
\\CDPL-DEVL-1	LogicalDisk	Avg. Disk Bytes/Read	C:	06-Jul-2007 16:48:26.9100	0
\\CDPL-DEVL-1	LogicalDisk	Avg. Disk sec/Read	C:	06-Jul-2007 16:48:26.9100	0
\\CDPL-DEVL-1	LogicalDisk	Avg. Disk sec/Transfer	C:	06-Jul-2007 16:48:26.9100	0
\\CDPL-DEVL-1	Processor	% Processor Time	_Total	06-Jul-2007 16:48:26.9100	0
\\CDPL-DEVL-1	Processor	% Privileged Time	0	06-Jul-2007 16:48:26.9100	0
\\CDPL-DEVL-1	LogicalDisk	Disk Transfers/sec	C:	06-Jul-2007 16:48:26.9100	0
\\CDPL-DEVL-1	Memory	Demand Zero Faults/sec	na	06-Jul-2007 16:48:26.9100	185.821490260194

Figure 1 Generic raw windows data format

In most cases this is the only table that is required to access external data. From here we have a separate Captell query for each Windows performance object; these queries pivot the data into a more useful format and provide the input to object level table definitions. Figure 2 shows the partial SYSTEM object table.

MachineName	InstanceName	DateTime	Context Switches/sec	ProcessorQueueLength	System Up Time	% Registry Quota In Use	AlignmentFixups/sec	Exception
\\CDPL-DEVL-1	na	06-Jul-2007 16:45:00.0000	450.97930824127911	6	279159.426295	(null)	(null)	(null)
\\CDPL-DEVL-1	na	06-Jul-2007 16:50:00.0000	275.17220563022658	4	279460.159431	(null)	(null)	(null)
\\CDPL-DEVL-1	na	06-Jul-2007 16:55:00.0000	258.46974631357159	3	279760.908231	(null)	(null)	(null)
\\CDPL-DEVL-1	na	06-Jul-2007 17:00:00.0000	260.50641347334357	5	280061.657031	(null)	(null)	(null)
\\CDPL-DEVL-1	na	06-Jul-2007 17:05:00.0000	288.19346815418248	4	280362.405831	(null)	(null)	(null)
\\CDPL-DEVL-1	na	06-Jul-2007 17:10:00.0000	252.57634840574264	7	280663.154631	(null)	(null)	(null)
\\CDPL-DEVL-1	na	06-Jul-2007 17:15:00.0000	254.53308488751114	4	280963.903431	(null)	(null)	(null)
\\CDPL-DEVL-1	na	06-Jul-2007 17:20:00.0000	254.35640778941934	3	281264.652731	(null)	(null)	(null)
\\CDPL-DEVL-1	na	06-Jul-2007 17:25:00.0000	259.31648585385653	4	281565.401031	(null)	(null)	(null)
\\CDPL-DEVL-1	na	06-Jul-2007 17:30:00.0000	253.70309207461568	3	281866.149831	(null)	(null)	(null)
\\CDPL-DEVL-1	na	06-Jul-2007 17:35:00.0000	275.25003698583328	3	282166.898531	(null)	(null)	(null)

Figure 2 Pivoted SYSTEM object ready for reporting

Figure 3 shows the PIVOT query that is used to process the raw data table, Figure 1, to produce the useful table at Figure 2.

```
select      *
From        (Select    [MachineName],
                      [InstanceName],
                      [CounterName],
                      dbo.cap_round_time([DateTime],5) as [DateTime] ,
                      avg([CounterValue]) as [CounterValue]
            From      [Data\Raw\Tables\RetainedRawData]
            where     ObjectName = 'System'
            group by  [MachineName],
                      [InstanceName],
                      [CounterName],
                      dbo.cap_round_time([DateTime],5)) p
Pivot      (AVG([CounterValue])
FOR CounterName IN      ([% Registry Quota In Use],
                        [Alignment Fixups/sec],
                        [Context Switches/sec],
                        [Exception Dispatches/sec],
                        [File Control Bytes/sec],
                        [File Control Operations/sec],
                        [File Data Operations/sec],
                        [File Read Bytes/sec],
                        [File Read Operations/sec],
                        [File Write Bytes/sec],
                        [File Write Operations/sec],
                        [Floating Emulations/sec],
                        [Processes],
                        [Processor Queue Length],
                        [System Calls/sec],
                        [System Up Time],
                        [Threads])) as pvt
```

**Figure 3 Pivot query to restructure "raw" data**

The structure of this query is such that if you are not collecting all the listed counters it will still function correctly by creating a column named appropriately but populated with nulls. The query is used as input to a table which is the permanent repository for the object level data.

We have currently adopted this technique for data generated from Microsoft SCOM, MOM, CA Unicentre, BMC Patrol and Mercury Sitescope.

Once this basic structure is in place for one performance object it's a simple matter to replicate for any other objects of interest.

As mentioned above this type of processing is relatively easy to establish when you have suitable monitoring software in place; however, what can we do where we do not have a suitable monitor.

### ***Using Windows Perfmon to collect data***

As most will be aware Windows has a built-in performance monitor names Perfmon, this can be accessed by typing PERFMON at the Run... command. What most will be unaware is that Perfmon can also log it's data to SQL Server providing a free powerful collection tool for capacity and performance reporting. I would not recommend using this technique for very large sites without more detailed testing but it is quite adequate for smaller sites or when you need a

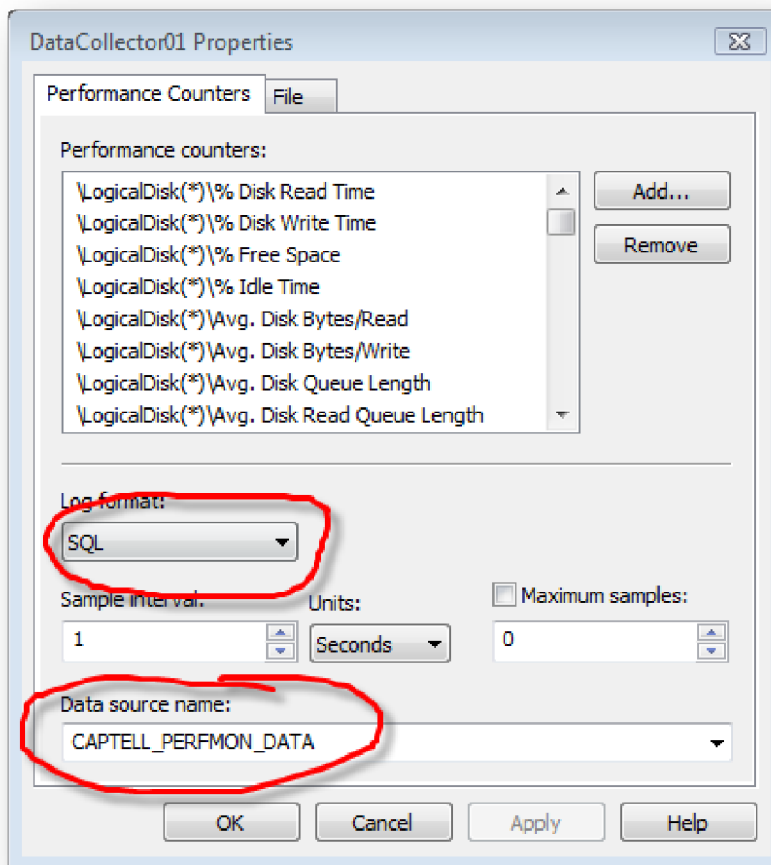
---

higher level or frequency of data collection than your current system wide monitors are collecting.

To implement Perfmon collection firstly install an instance of SQL Server, I would recommend installing SQL Server 2005 Express (downloadable from <http://msdn2.microsoft.com/en-au/express/bb410792.aspx>), once installed you can create a new empty database in your instance.

Secondly you will need to establish a Data Source Name (DSN) pointing at your database, this is done using Control panel → Administrative tools → Data Sources ODBC. Create a new SYSTEM DSN pointing at your SQL Server instance and new database.

Thirdly setup Perfmon to collect the required counters and store them in your database by setting the Data Collector Set to store its data in SQL Server format, you then need to specify your DSN.



Once started Perfmon will create the necessary tables in your database which can then be processed using the Captell templates described above. For more information on these templates contact Captell Developments [www.captelldevelopments.com](http://www.captelldevelopments.com).

---