THE QUICK START GUIDE FOR ARRAY PERFORMANCE
Hello, I am Mike Walker.
Welcome to a demostration of how to find performance information in an SMI-S 1.8.0 Array.
This demo is shown as a Quick Start Guide for Array performance on my GitHub site for
SMI-S Mocks, identified in the video on installation and setup for running mock ups of SMI-S WBEM Servers.
The quick start guide is a 13-page document highlights information that can be found in
about 70 pages of SMI-S, the block server performance profile in the block book of SMI-S.
In this script we will be working with pywbem 1.1.1 and version 0.8.0 of pywbemtools

(pywbemcli).
So, let's begin. We will switch to my command prompt window.

First, we go to our virtual environment for mocks:

```
C:\Users\FarmerMike>workon mocks
```

We are now in our virtual environment for mocks.
We will be working with a mock server that supports an SMI-S 1.8.0 Array. We will point out items that were introduced
after 1.6.1.
So, we need to establish a connection to the Array mock up:

```
(mocks) c:\Users\FarmerMike\devenv> pywbemcli -o table -n ArrayMock
```

In our command, I requested the default format of output to be in “table” format (-o table) and name the mock that I want
(-n ArrayMock). The command worked and we get a pywbemcli prompt to start entering commands on the ArrayMock.

PERFORMANCE CAPABILITIES
In SMI-S 1.8.0 Advanced Metrics was defined for storage Arrays. We can determine if our Mock Array supports this by
inspecting the CIM_BlockStatisticsCapabilities. Specifically, the SupportedFeatures property will include the value "6".
So, let's get the performance capabilities. We will do this with two requests for readability.

```
pywbemcli> instance enumerate CIM_BlockStatisticsCapabilities --pl InstanceID,SupportedFeatures
pywbemcli> instance enumerate CIM_BlockStatisticsCapabilities --pl InstanceID,ElementTypesSupported
```

In the first request we see SupportedFeatures does, indeed, contain the value "6" (as well as the value "3"). In the second
request we also see that statistics are being kept for the Array itself, the front

end port, storage volume and disk drive
elements.
Next, we will see what other capabilities are supported by runnlng the following commands:

```
pywbemcli> instance enumerate CIM_BlockStatisticsCapabilities --pl
InstanceID,SynchronousMethodsSupported
pywbemcli> instance enumerate CIM_BlockStatisticsCapabilities --pl
InstanceID,ClockTickInterval
```

In the first request we see that the methods supported are GetStatisticsCollection, Manifest Creation, Manifest Modification
and Manifest Removal. And in the second request we see that the implementation has a clock tick interval of 100000
microseconds. The [us] indicates the units are microseconds. IO time counters are in terms of the number of clock tick
intervals.
Next, we will look for what the BlockStatisticsCapabilities are related to with the following command:

```
pywbemcli> -o mof instance shrub CIM_BlockStatisticsCapabilities.?
```

This command is a shrub command. It is looking for everything that is directly related to CIM_BlockStatisticsCapabilities.
The -o mof option says we want to switch to the mof format for the output for this command.
And we see the capabilities are related to an instance of CIM_BlockStatisticsService (via the CIM_ElementCapabilities
association).

BLOCK STATISTICS SERVICE
So, let's get the related service.

```
pywbemcli> instance enumerate CIM_BlockStatisticsService --pl
Name,EnabledDefault,EnabledState,RequestedState,Started
```

We see that both the EnabledState and the EnabledDefault is Enabled, and RequestedState is not applicable and Started is
true (meaning the service is has been started).

STATISTICS COLLECTION
In terms of finding the statistics there are a couple of approaches to take. You could start with an element for which
statistics are kept. But a more straightforward approach is to find the Statistics Collection. This collection collects
all the statistics for all the element types.
So, let’s find the statistics collection with the following command:

```
pywbemcli> -o mof instance enumerate CIM_StatisticsCollection
```

The statistics collection identifies the sample interval (15 minutes) and the last time they statistics were collected.
Next, we need to inspect the shrub for the Statistics Collection.

```
pywbemcli> -o mof instance shrub CIM_StatisticsCollection.?
```

We see that there are three associations to other elements. There is a CIM_AssociatedBlockStatisticsManifestCollection
to a CIM_BlockStatisticsManifestCollection. There is a CIM_HostedCollection to a CIM_ComputerSystem. And there is a
CIM_MemberOfCollection to a bunch of instances of CIM_BlockStorageStatisticalData. These instances contain the
statistics data for the element types identified in the CIM_BlockStatisticsCapabilities.
We will start by looking at the instances of CIM_BlockStorageStatisticalData.

BLOCK STATISTICAL DATA
We will look at one instance of each element type, starting with the Array. To illustrate what the Advanced Metrics offers,
I will run two requests for each element type. The first command will retrieve the basic statistics and the second will
retrieve the advanced statistics.
For the Array (type 2) the command to retrieve the basic statistics is:

```
pywbemcli> instance get CIM_BlockStorageStatisticalData.? --pl
InstanceID,ElementType,TotalIOs,KBytesTransferred
```

To get the statistics for the Array, we select item 0:

```
Input integer between 0 and 11 or Ctrl-C to exit selection: 0
```

The KBytesTransferred is in 1024 byte (kB) chunks. So this array has transferred 66 1024 chunks of data. The IOs
refers to the number of IOs processed by the array.
The command to get the advanced statistics or the Array is:

```
pywbemcli> instance get CIM_BlockStorageStatisticalData.? --pl
InstanceID,ReadIOs,ReadHitIOs,KBytesRead,WriteIOs,WriteHitIOs,KBytesWritten
```

I repeated InstanceID to verify the two queries are on the same element. We see the Advanced Metrics adds quite a few
property values over the basic support. ReadHitIOs are the cumulative count of reads that are satisfied from the cache.
The WriteHitIOs are the cumulative count of writes that went to the cache.
For the Front End Ports (type 6) the command to retrieve the basic statistics is:

```
pywbemcli> instance get CIM_BlockStorageStatisticalData.? --pl
InstanceID,ElementType,TotalIOs,KBytesTransferred
Input integer between 0 and 11 or Ctrl-C to exit selection: 9
```

This shows the basic metrics. Now we retrieve the advanced metrics:

```
pywbemcli> instance get CIM_BlockStorageStatisticalData.? --pl
InstanceID,ElementType,IOTimeCounter
Input integer between 0 and 11 or Ctrl-C to exit selection: 9
```

We see the Advanced Metrics adds the IOTimeCounter statistic. The IOTimeCounter is the cumulative elapsed I/O time (number of Clock Tick Intervals) for all I/Os as defined in “Total I/Os”.
For the Storage Volumes (type 8) the command to retrieve the basic statistics is:

```
pywbemcli> instance get CIM_BlockStorageStatisticalData.? --pl
InstanceID,ElementType,TotalIOs,KBytesTransferred,ReadIOs,WriteIOs
Input integer between 0 and 11 or Ctrl-C to exit selection: 11
```

This shows the basic metrics. Now we retrieve the advanced metrics:

```
pywbemcli> instance get CIM_BlockStorageStatisticalData.? --pl
InstanceID,ElementType,IOTimeCounter,KBytesRead,KBytesWritten
Input integer between 0 and 11 or Ctrl-C to exit selection: 11
```

We see the Advanced Metrics adds the IOTimeCounter, KBytesRead and KBytesWritten statistics.
For the Disk Drives (type 10) the command to retrieve the basic statistics is:

```
pywbemcli> instance get CIM_BlockStorageStatisticalData.? --pl
InstanceID,ElementType,TotalIOs,KBytesTransferred,ReadIOs
Input integer between 0 and 11 or Ctrl-C to exit selection: 1
```

This shows the basic metrics. Now we retrieve the advanced metrics.

```
pywbemcli> instance get CIM_BlockStorageStatisticalData.? --pl
InstanceID,IOTimeCounter,KBytesRead,WriteIOs,KBytesWritten,IdleTimeCounter
Input integer between 0 and 11 or Ctrl-C to exit selection: 1
```

The advance metrics add quite a few statistics. Actually, Advanced Metrics calls for EITHER the IOTimeCounter OR the
IdleTimeCounter. The IdleTimeCounter is the cumulative elapsed idle time using ClockTickInterval units (Cumulative Number
of Time Units for all idle time in the array).

BLOCK STATISTICS MANIFESTS AND COLLECTION
One last thing to point out on the block server performance profile. The CIM_BlockStatisticsManifestCollection associated
with the Statistics Collection (see the shrub for the CIM_StatisticsCollection) contains manifests for each of the
element types. There is some important information in these manifests. So, let's look at those manifests:

```
pywbemcli> -o mof instance shrub CIM_BlockStatisticsManifestCollection.? --ac
```

CIM_MemberOfCollection

Here we see there are default manifests for each element type. So, let's look at one to illustrate what the manifest tells us.

pywbemcli> 0-o mof instance get CIM_BlockStatisticsManifest.?

We will look at the manifest for the Array (selection 0).

Input integer between 0 and 3 or Ctrl-C to exit selection: 0

The first thing we see is that the manifest contains a lot of "Include ..." properties. This tells us what statistics
are kept for the element type. These are booleans. A value of true means the statistic is kept.
Next there is the CSVSequence property. It tells what sequence the data should be ordered in a data record
(this is unrelated to the sequence provided by CIM or pywbem or the pywbemcli). When the data is retrieved and written
to a file, the CSVSequence identifies the desired order of headers and values.