ARRAY PERFORMANCE

Multiple Namespace Quick Start Guide

APRIL 20, 2023

MICHAEL WALKER

Version 2.0.0

QUICK START GUIDE FOR ARRAY PERFORMANCE INFORMATION

TABLE OF CONTENTS

Introduction

WHAT IS A MULTIPLE NAMESPACE SMI-S SERVER

CIM supports multiple namespaces (CIM_Namespace instances). Keys are unique within a namespace. Most SMI-S servers support multiple namespaces. One for the server infrastructure (the interop namespace) and one for the storage device supported. This allows separation of naming conventions between the server infrastructure and the device support. SMI-S implementations may actually have more than two namespaces. However, this guide only shows the two mentioned.

WHAT IS A QUICK START GUIDE?

SMI-S is 2516 pages of reading spread across 8 books, plus it references another 14 or so DMTF profiles which amount to another 660 pages of reading. So, the question is where do you start? We have come up with a series of Quick Start Guides that are designed to help you get started by illustrating how to find useful SMI-S information in mock servers (mock ups of SMI-S server implementations). The Quick Start Guides don't illustrate EVERYTHING in the 3176 pages, but they give you a head start at finding some important items in SMI-S.

We currently have quick start guides for:

1. The Interop Namespace - What is it and what does it tell us?
2. Performance Information - Where do I find performance information in an SMI-S Server?

3. Capacity Information - Where do I find storage capacity information in an SMI-S Server?
4. Hardware Information - Where do I find hardware information in an SMI-S Server?
5. Product Information - Where do I find product information in an SMI-S Server?
6. Software Information - Where do I find software information in an SMI-S Server?

TOOL USED FOR THE QUICK START GUIDE ILLUSTRATION

The tool used for illustrating how to find information in SMI-S is the pywbemcli (part of pywbemtools). It is a command line interface for accessing any WBEM Server. It uses pywbem, an interface for python program access to any WBEM Server. The pywbemtools (and the pywbemcli) and pywbem are python programs that use a set of python packages. Pywbem and the pywbemtools are actively being maintained and are available on Github.

We will be using the latest version of the pywbemcli in these guides. You can find documentation on the pywbemcli at the following website:

https://pywbemtools.readthedocs.io/en/latest/

THE MOCK IMPLEMENTATIONS

The mock implementations mock selected autonomous profiles and some of their component profiles in SMI-S 1.8.0.

We currently have mock ups for the following autonomous profiles:

1. The SNIA Server Profile
2. The DMTF WBEM Server Profile
3. The Array Profile
4. The NAS Head Profile

And we plan on doing a Fabric (and Switch) mock up.

We chose to do mock ups of the SMI-S 1.8.0 versions of these profiles to illustrate differences between 1.8.0 and 1.6.1. We don't mock everything that is new in 1.8.0, but we do highlight some key changes ... like the DMTF WBEM Server profile, new indications and Advance Metrics (performance) for Arrays.

HOW A GUIDE WORKS

The guide is a sequence of text explaining what we are looking for, followed by the command to obtain the information, followed by command output and then text that explains the output.

THE QUICK START GUIDE FOR ARRAY PERFORMANCE

In this guide we will be exploring performance information provided by an SMI-S Server for an Array. The mock for the Array supports SMI-S 1.8.0 and we will identify the differences between 1.6.1 and 1.8.0.

This 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 python 3.8 (or above), pywbem 1.6.1 and version 1.2.0 of pywbemtools (pywbemcli). It is important that you are running with pywbemtools 1.2.0. In developing

these scripts, the pywbem team was asked to make enhancements for multiple namespace access. The enhancements are only available in version 1.2.0 and beyond.

So, let's begin. First, we go to our virtual environment for mocks:

```
C:\Users\FarmerMike> workon mocks
(mocks) c:\Users\FarmerMike\devenv>
```

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 --default-namespace device
Enter 'help' for help, <CTRL-D> or ':q' to exit pywbemcli or <CTRL-r> to search history,
pywbemcli>
```

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. We also asked that the default namespace be ‘device’. This tells our load program that we want two namespaces. The default namespace for the Array instances and the interop namespace for the infrastructure instances.

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 -n
interop,device
Instances: CIM_BlockStatisticsCapabilities
+-----------+-----------------------------------+-----------------------------------------------------------+
| namespace | InstanceID                        | SupportedFeatures                                         |
|-----------+-----------------------------------+-----------------------------------------------------------|
| device    | "ACME+CF2A5091300089"             | 3 (Client Defined Sequence), 6 (Advanced Metrics)         |
+-----------+-----------------------------------+-----------------------------------------------------------+
pywbemcli>
pywbemcli> instance enumerate CIM_BlockStatisticsCapabilities --pl InstanceID,ElementTypesSupported -n
interop,device
Instances: CIM_BlockStatisticsCapabilities
+-----------+-----------------------------------+-----------------------------------------------------------------------------------------+
| namespace | InstanceID                        | ElementTypesSupported                                                                   |
|-----------+-----------------------------------+-----------------------------------------------------------------------------------------|
| device    | "ACME+CF2A5091300089"             | 2 (Computer System), 6 (Front-end Port), 8 (Volume), 10 (Disk Drive)                    |
+-----------+-----------------------------------+-----------------------------------------------------------------------------------------+
pywbemcli>
```

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. And the instance is in the device namespace.

Next, we will see what other capabilities are supported by runnlng the following commands:

```
pywbemcli> instance enumerate CIM_BlockStatisticsCapabilities --pl
InstanceID,SynchronousMethodsSupported -n interop,device
Instances: CIM_BlockStatisticsCapabilities
+-------------+-----------------------------+-----------------------------------------------------------------------------------------------------------------------------+
|namespace| InstanceID                  | SynchronousMethodsSupported                                                                                                 |
|-------------+-----------------------------+-----------------------------------------------------------------------------------------------------------------------------|
| device      | "ACME+CF2A5091300089" | 4 (GetStatisticsCollection), 5 (Manifest Creation), 6 (Manifest Modification), 7 (Manifest Removal) |
+-------------+-----------------------------+-----------------------------------------------------------------------------------------------------------------------------+
pywbemcli>
```

```
pywbemcli> instance enumerate CIM_BlockStatisticsCapabilities --pl InstanceID,ClockTickInterval -n
interop,device
Instances: CIM_BlockStatisticsCapabilities
+---------------+---------------------------------------+-----------------------------+
| namespace | InstanceID                        |  ClockTickInterval [us] |
|---------------+---------------------------------------+-----------------------------|
| device        | "ACME+CF2A5091300089" |                     100000 |
+---------------+---------------------------------------+-----------------------------+
pywbemcli>
```

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.? -n device
CIM_BlockStatisticsCapabilities.InstanceID="ACME+CF2A5091300089"
 +-- Capabilities(Role)
    +-- CIM_ElementCapabilities(AssocClass)
       +-- ManagedElement(ResultRole)
          +-- CIM_BlockStatisticsService(ResultClass)(1 insts)
             +-- /:CIM_BlockStatisticsService.~,Name="BlockStatisticsService",~,~
pywbemcli>
```

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. The -n device argument tells the shrub command to use the CIM_BlockStatisticsCapabilities in the device namespace. But the results will include elements in other namespaces (if there are any).

And we see the capabilities are related to an instance of CIM_BlockStatisticsService (via the CIM_ElementCapabilities association). And the service is also in the device namespace.

BLOCK STATISTICS SERVICE

So, let's get the related service.

```
pywbemcli> instance enumerate CIM_BlockStatisticsService --pl
Name,EnabledDefault,EnabledState,RequestedState,Started -n interop,device
Instances: CIM_BlockStatisticsService
+-----------+-----------------------------+----------------+--------------+------------------------+---------+
| namespace | Name                        | EnabledDefault | EnabledState | RequestedState         | Started |
|-----------+-----------------------------+----------------+--------------+------------------------+---------|
| device    | "BlockStatisticsService"    | 2 (Enabled)    | 2 (Enabled)  | 12 (Not Applicable)    | true    |
+-----------+-----------------------------+----------------+--------------+------------------------+---------+
pywbemcli>
```

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). And the service is, indeed, in the device namespace.

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 -n interop,device
#pragma namespace ("device")
instance of CIM_StatisticsCollection {
   SampleInterval = "00000000001500.000000:000";
   TimeLastSampled = "20131029094525.737000-240";
   InstanceID = "ACME+CF2A5091300089";
   ElementName = "Statistics Collection";
};

pywbemcli>
```

The statistics collection identifies the sample interval (15 minutes) and the last time they statistics were collected. The #pragma namespace ("device") is how the namespace is identified in the mof format.

Next, we need to inspect the shrub for the Statistics Collection.

```
pywbemcli> -o mof instance shrub CIM_StatisticsCollection.? -n device
CIM_StatisticsCollection.InstanceID="ACME+CF2A5091300089"
 +-- Statistics(Role)
```

```
|   +-- CIM_AssociatedBlockStatisticsManifestCollection(AssocClass)
|      +-- ManifestCollection(ResultRole)
|         +-- CIM_BlockStatisticsManifestCollection(ResultClass)(1 insts)
|            +-- /:CIM_BlockStatisticsManifestCollection.InstanceID="ACME+CF2A5091300089+MANIFEST_DEFAULT"
+-- Dependent(Role)
|   +-- CIM_HostedCollection(AssocClass)
|      +-- Antecedent(ResultRole)
|         +-- CIM_ComputerSystem(ResultClass)(1 insts)
|            +-- /:CIM_ComputerSystem.~,Name="ACME+CF2A5091300089"
+-- Collection(Role)
    +-- CIM_MemberOfCollection(AssocClass)
       +-- Member(ResultRole)
          +-- CIM_BlockStorageStatisticalData(ResultClass)(12 insts)
             +-- /:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Array"
             +-- /:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_0"
             +-- /:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_1"
             +-- /:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_2"
             +-- /:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_3"
             +-- /:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_4"
             +-- /:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_5"
             +-- /:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_6"
             +-- /:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_7"
             +-- /:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+FEPort+SP_A:0"
             +-- /:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+FEPort+SP_A:1"
             +-- /:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Volume+00005"
pywbemcli>
```

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 can find the instances of statistic data through the following enumerate command.

```
pywbemcli> instance enumerate CIM_BlockStorageStatisticalData --pl InstanceID,ElementType -n interop,device
Instances: CIM_BlockStorageStatisticalData
+-----------+-----------------------------------------------------+--------------------------+
| namespace | InstanceID                                          | ElementType              |
|-----------+-----------------------------------------------------+--------------------------|
| device    | "ACME+CF2A5091300089+Array"                         | 2 (Computer System)      |
| device    | "ACME+CF2A5091300089+Disk+0_0_0"                    | 10 (Disk Drive)          |
| device    | "ACME+CF2A5091300089+Disk+0_0_1"                    | 10 (Disk Drive)          |
| device    | "ACME+CF2A5091300089+Disk+0_0_2"                    | 10 (Disk Drive)          |
```

```
| device      | "ACME+CF2A5091300089+Disk+0_0_3"      | 10 (Disk Drive)         |
| device      | "ACME+CF2A5091300089+Disk+0_0_4"      | 10 (Disk Drive)         |
| device      | "ACME+CF2A5091300089+Disk+0_0_5"      | 10 (Disk Drive)         |
| device      | "ACME+CF2A5091300089+Disk+0_0_6"      | 10 (Disk Drive)         |
| device      | "ACME+CF2A5091300089+Disk+0_0_7"      | 10 (Disk Drive)         |
| device      | "ACME+CF2A5091300089+FEPort+SP_A:0" | 6 (Front-end Port)    |
| device      | "ACME+CF2A5091300089+FEPort+SP_A:1" | 6 (Front-end Port)    |
| device      | "ACME+CF2A5091300089+Volume+00005"| 8 (Volume)              |
+---------------+-----------------------------------------------------------+-------------------------+
pywbemcli>
```

We see that all the statistics are in the device namespace. They are statistics for a computer system, disk drives, front end ports and a volume.We will look at one instance of each element type, starting with the Array. To illustrate what the Advanced Metrics offers, we 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 -n device
Pick Instance name to process
0: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Array"
1: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_0"
2: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_1"
3: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_2"
4: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_3"
5: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_4"
6: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_5"
7: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_6"
8: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_7"
9: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+FEPort+SP_A:0"
10: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+FEPort+SP_A:1"
11: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Volume+00005"
Input integer between 0 and 11 or Ctrl-C to exit selection:
```

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

```
Input integer between 0 and 11 or Ctrl-C to exit selection: 0
Instances: CIM_BlockStorageStatisticalData
+-----------------------------------------+---------------------------+------------+-------------------------------+
| InstanceID                              | ElementType               | TotalIOs |  KBytesTransferred [kB] |
|-----------------------------------------+---------------------------+------------+-------------------------------|
| "ACME+CF2A5091300089+Array" | 2 (Computer System) |       132 |                            66 |
+-----------------------------------------+---------------------------+------------+-------------------------------+
pywbemcli>
```

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 -n device
Pick Instance name to process
0: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Array"
1: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_0"
2: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_1"
3: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_2"
4: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_3"
5: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_4"
6: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_5"
7: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_6"
8: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_7"
9: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+FEPort+SP_A:0"
10: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+FEPort+SP_A:1"
11: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Volume+00005"
Input integer between 0 and 11 or Ctrl-C to exit selection:
Input integer between 0 and 11 or Ctrl-C to exit selection: 0
Instances: CIM_BlockStorageStatisticalData
+--------------------------+----------+----------------+----------------------+----------+-----------------+---------------------+
| InstanceID               | ReadIOs  | ReadHitIOs     | KBytesRead [kB]      | WriteIOs | WriteHitIOs     | KBytesWritten [kB]  |
|--------------------------+----------+----------------+----------------------+----------+-----------------+---------------------|
| "ACME+CF2A50913"         |      132 |        1844272 |                   66 |        0 |         1845899 |                   0 |
| "00089+Array"            |          |                |                      |          |                 |                     |
+--------------------------+----------+----------------+----------------------+----------+-----------------+---------------------+pywbemcli>
```

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
Pick Instance name to process
0: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Array"
1: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_0"
2: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_1"
3: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_2"
4: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_3"
5: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_4"
6: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_5"
7: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_6"
8: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_7"
9: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+FEPort+SP_A:0"
10: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+FEPort+SP_A:1"
```

```
11: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Volume+00005"
Input integer between 0 and 11 or Ctrl-C to exit selection:
Input integer between 0 and 11 or Ctrl-C to exit selection: 9
Instances: CIM_BlockStorageStatisticalData
+-----------------------------------------------+-----------------------+-------------+--------------------------------+
| InstanceID                                    | ElementType           |  TotalIOs |  KBytesTransferred [kB] |
|-----------------------------------------------+-----------------------+-------------+--------------------------------|
| "ACME+CF2A5091300089+FEPort+S" | 6 (Front-end Port) |          66 |                             33 |
| "P_A:0"                                       |                       |             |                                |
+-----------------------------------------------+-----------------------+-------------+--------------------------------
+pywbemcli>
```

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

```
pywbemcli> instance get CIM_BlockStorageStatisticalData.? --pl InstanceID,ElementType,IOTimeCounter
Pick Instance name to process
0: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Array"
1: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_0"
2: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_1"
3: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_2"
4: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_3"
5: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_4"
6: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_5"
7: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_6"
8: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_7"
9: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+FEPort+SP_A:0"
10: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+FEPort+SP_A:1"
11: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Volume+00005"
Input integer between 0 and 11 or Ctrl-C to exit selection:
Input integer between 0 and 11 or Ctrl-C to exit selection: 9
Instances: CIM_BlockStorageStatisticalData
+-----------------------------------------------+-----------------------+----------------+
| InstanceID                                    | ElementType           |  IOTimeCounter |
|-----------------------------------------------+-----------------------+----------------|
| "ACME+CF2A5091300089+FEPort+SP_A:0" | 6 (Front-end Port) |         935579 |
+-----------------------------------------------+-----------------------+----------------+
pywbemcli>
```

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 -n device
Pick Instance name to process
0: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Array"
```

```
1: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_0"
2: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_1"
3: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_2"
4: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_3"
5: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_4"
6: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_5"
7: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_6"
8: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_7"
9: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+FEPort+SP_A:0"
10: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+FEPort+SP_A:1"
11: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Volume+00005"
Input integer between 0 and 11 or Ctrl-C to exit selection:
Input integer between 0 and 11 or Ctrl-C to exit selection: 11
Instances: CIM_BlockStorageStatisticalData
+--------------------------------+-------------------+-------------+--------------------------+-------------+--------------+
| InstanceID                     | ElementType   |  TotalIOs |  KBytesTransferred |  ReadIOs |  WriteIOs |
|                                |                   |             |                    [kB] |             |             |
|--------------------------------+-------------------+-------------+--------------------------+-------------+-------------|
| "ACME+CF2A50913000" | 8 (Volume)     |       132 |                   66 |       132 |           0 |
| "89+Volume+00005"              |                   |             |                      |             |             |
+--------------------------------+-------------------+-------------+--------------------------+-------------+--------------+pywbemcli>
```

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

```
pywbemcli> instance get CIM_BlockStorageStatisticalData.? --pl
InstanceID,ElementType,IOTimeCounter,KBytesRead,KBytesWritten -n device
Pick Instance name to process
0: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Array"
1: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_0"
2: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_1"
3: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_2"
4: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_3"
5: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_4"
6: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_5"
7: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_6"
8: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_7"
9: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+FEPort+SP_A:0"
10: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+FEPort+SP_A:1"
11: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Volume+00005"
Input integer between 0 and 11 or Ctrl-C to exit selection:
Input integer between 0 and 11 or Ctrl-C to exit selection: 11
Instances: CIM_BlockStorageStatisticalData
+----------------------------------------+-------------------+----------------------+------------------------+---------------------------+
| InstanceID                             | ElementType   |  IOTimeCounter |  KBytesRead [kB] |  KBytesWritten [kB] |
|----------------------------------------+-------------------+----------------------+------------------------+---------------------------|
| "ACME+CF2A5091300089+Vo" | 8 (Volume)     |             152996 |                   66 |                         0 |
| "lume+00005"                           |                   |                      |                      |                           |
+----------------------------------------+-------------------+----------------------+------------------------+---------------------------+
```

```
pywbemcli>
```

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 -n device
Pick Instance name to process
0: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Array"
1: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_0"
2: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_1"
3: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_2"
4: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_3"
5: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_4"
6: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_5"
7: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_6"
8: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_7"
9: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+FEPort+SP_A:0"
10: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+FEPort+SP_A:1"
11: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Volume+00005"

Input integer between 0 and 11 or Ctrl-C to exit selection:
Input integer between 0 and 11 or Ctrl-C to exit selection: 1
Instances: CIM_BlockStorageStatisticalData
+-----------------------------------------+---------------------+--------------+--------------------------------+----------------+
| InstanceID                              | ElementType         |    TotalIOs |   KBytesTransferred [kB] |   ReadIOs   |
|-----------------------------------------+---------------------+--------------+--------------------------------+----------------|
| "ACME+CF2A5091300089+Di" | 10 (Disk Drive)  |   39726146 |                 453576349 |  38658792 |
| "sk+0_0_0"                              |                     |              |                                |              |
+-----------------------------------------+---------------------+--------------+--------------------------------+----------------+pywbemcli>
```

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

```
pywbemcli> instance get CIM_BlockStorageStatisticalData.? --pl
InstanceID,IOTimeCounter,KBytesRead,WriteIOs,KBytesWritten,IdleTimeCounter -n device
Pick Instance name to process
0: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Array"
1: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_0"
2: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_1"
3: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_2"
4: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_3"
5: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_4"
6: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_5"
7: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_6"
8: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Disk+0_0_7"
9: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+FEPort+SP_A:0"
10: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+FEPort+SP_A:1"
```

```
11: device:CIM_BlockStorageStatisticalData.InstanceID="ACME+CF2A5091300089+Volume+00005"
Input integer between 0 and 11 or Ctrl-C to exit selection:
Input integer between 0 and 11 or Ctrl-C to exit selection: 1
Instances: CIM_BlockStorageStatisticalData
+--------------------------------+---------------------+-----------------------+--------------+--------------------------+-----------------------+
| InstanceID                     |  IOTimeCounter |  KBytesRead [kB] |  WriteIOs  |  KBytesWritten [kB] |  IdleTimeCounter |
|--------------------------------+---------------------+-----------------------+--------------+--------------------------+-----------------------|
| "ACME+CF2A50913000" |         1941691 |         428873067 |  1067354 |            24703282 |         460929424 |
| "89+Disk+0_0_0"          |                     |                       |              |                          |                       |
+--------------------------------+---------------------+-----------------------+--------------+--------------------------+-----------------------+
pywbemcli>
```

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 -n device
CIM_BlockStatisticsManifestCollection.InstanceID="ACME+CF2A5091300089+MANIFEST_DEFAULT"
 +-- Collection(Role)
    +-- CIM_MemberOfCollection(AssocClass)
       +-- Member(ResultRole)
          +-- CIM_BlockStatisticsManifest(ResultClass)(4 insts)
             +-- /:CIM_BlockStatisticsManifest.InstanceID="ACME+CF2A5091300089+ACMEMANIFEST_DEFAULT+Array"
             +-- /:CIM_BlockStatisticsManifest.InstanceID="ACME+CF2A5091300089+ACMEMANIFEST_DEFAULT+Disk"
             +-- /:CIM_BlockStatisticsManifest.InstanceID="ACME+CF2A5091300089+ACMEMANIFEST_DEFAULT+FEPort"
             +-- /:CIM_BlockStatisticsManifest.InstanceID="ACME+CF2A5091300089+ACMEMANIFEST_DEFAULT+Volume"
pywbemcli>
```

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> -o mof instance get CIM_BlockStatisticsManifest.? -n device
Pick Instance name to process
0: device:CIM_BlockStatisticsManifest.InstanceID="ACME+CF2A5091300089+ACMEMANIFEST_DEFAULT+Array"
1: device:CIM_BlockStatisticsManifest.InstanceID="ACME+CF2A5091300089+ACMEMANIFEST_DEFAULT+Disk"
2: device:CIM_BlockStatisticsManifest.InstanceID="ACME+CF2A5091300089+ACMEMANIFEST_DEFAULT+FEPort"
3: device:CIM_BlockStatisticsManifest.InstanceID="ACME+CF2A5091300089+ACMEMANIFEST_DEFAULT+Volume"
Input integer between 0 and 3 or Ctrl-C to exit selection:
```

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

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

```
instance of CIM_BlockStatisticsManifest {
   InstanceID = "ACME+CF2A5091300089+ACMEMANIFEST_DEFAULT+Array";
   ElementType = 2;
   RateElementType = 0;
   IncludeStartStatisticTime = false;
   IncludeStatisticTime = true;
   IncludeTotalIOs = true;
   IncludeKBytesTransferred = true;
   IncludeIOTimeCounter = true;
   IncludeReadIOs = true;
   IncludeReadHitIOs = true;
   IncludeReadIOTimeCounter = false;
   IncludeReadHitIOTimeCounter = false;
   IncludeKBytesRead = true;
   IncludeWriteIOs = true;
   IncludeWriteHitIOs = true;
   IncludeWriteIOTimeCounter = false;
   IncludeWriteHitIOTimeCounter = false;
   IncludeKBytesWritten = true;
   IncludeIdleTimeCounter = false;
   IncludeMaintOp = false;
   IncludeMaintTimeCounter = false;
   CSVSequence = { "InstanceID", "ElementType", "StatisticTime", "TotalIOs",
     "KBytesTransferred", "IOTimeCounter", "ReadIOs", "ReadHitIOs",
     "KBytesRead", "WriteIOs", "WriteHitIOs", "KBytesWritten" };
   CSVRateSequence = { };
   IncludeRateIntervalStartTime = false;
   IncludeRateIntervalEndTime = false;
   IncludeTotalIOsRate = false;
   IncludeKBytesTransferredRate = false;
   IncludeReadIOsRate = false;
   IncludeReadHitIOsRate = false;
   IncludeKBytesReadRate = false;
   IncludeWriteIOsRate = false;
   IncludeWriteHitIOsRate = false;
   IncludeKBytesWrittenRate = false;
   IncludeMaintOpRate = false;
   ElementName = "ACME+CF2A5091300089:DEFAULT";
};

pywbemcli>
```

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.

CHANGE LOG

1.0.1	10/21/2020	The initial version based on pywbem defaults
1.1.0	11/07/2021	Changed the default namespace to ‘interop’
2.0.0	4/20/2023	Updated for a multiple namespace mock