ARRAY CAPACITY

Multiple Namespace Quick Start Guide

APRIL 20, 2023

MICHAEL WALKER

Version 2.0.0

QUICK START GUIDE FOR ARRAY CAPACITY 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 CAPACITY

In this guide we will be exploring capacity information provided by an SMI-S Server for an Array. The mock for the Array supports SMI-S 1.8.0.

This 11-page document highlights information that can be found in about 104 pages of SMI-S, the block services and disk drive lite profiles 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. 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.

FIND THE TOP LEVEL SYSTEM FOR THE ARRAY

We will start by finding the registered profile for the Array Profile. We do this with the following command:

```
pywbemcli> instance enumerate CIM_RegisteredProfile  --pl
InstanceID,RegisteredOrganization,RegisteredName,RegisteredVersion -n interop,device
Instances: CIM_RegisteredProfile
+-----------+-----------------------------------------+----------------------------+-----------------------------------+-----------------------+
| namespace | InstanceID                              | RegisteredOrganization     | RegisteredName                    |RegisteredVersion|
|-----------+-----------------------------------------+----------------------------+-----------------------------------+-----------------------|
| interop   | "Array+1.8.0"                           | 11 (SNIA)                  | "Array"                           | "1.8.0"               |
| interop   | "Block Server Performance+1.8.0"        | 11 (SNIA)                  | "Block Server Performance"        | "1.8.0"               |
| interop   | "Block Services+1.8.0"                  | 11 (SNIA)                  | "Block Services"                  | "1.8.0"               |
| interop   | "DMTF Profile Registration+1.0"         | 2 (DMTF)                   | "Profile Registration"            | "1.0"                 |
| interop   | "Disk Drive Lite+1.8.0"                 | 11 (SNIA)                  | "Disk Drive Lite"                 | "1.8.0"               |
| interop   | "FC Target Ports+1.8.0"                 | 11 (SNIA)                  | "FC Target Ports"                 | "1.8.0"               |
| interop   | "IP Interface 1.1.1"                    | 2 (DMTF)                   | "IP Interface"                    | "1.1.1"               |
| interop   | "Indications+1.2.2"                     | 2 (DMTF)                   | "Indications"                     | "1.2.2"               |
| interop   | "Job Control 1.0.0"                     | 2 (DMTF)                   | "Job Control"                     | "1.0.0"               |
| interop   | "Multiple Computer System+1.2.0"        | 11 (SNIA)                  | "Multiple Computer System"        | "1.2.0"               |
| interop   | "Physical Package+1.8.0"                | 11 (SNIA)                  | "Physical Package"                | "1.8.0"               |
| interop   | "Profile Registration+1.7.0"            | 11 (SNIA)                  | "Profile Registration"            | "1.7.0"               |
| interop   | "Role Based Authorization 1.0.0"        | 2 (DMTF)                   | "Role Based Authorization"        | "1.0.0"               |
| interop   | "SMI-S+1.8.0"                           | 11 (SNIA)                  | "SMI-S"                           | "1.8.0"               |
| interop   | "Server+1.7.0"                          | 11 (SNIA)                  | "Server"                          | "1.7.0"               |
| interop   | "Simple Identity Management 1.1.0"      | 2 (DMTF)                   | "Simple Identity Management"      | "1.1.0"               |
| interop   | "Software+1.8.0"                        | 11 (SNIA)                  | "Software"                        | "1.8.0"               |
| interop   | "WBEM Server 1.0.1i"                    | 2 (DMTF)                   | "WBEM Server"                     | "1.0.1i"              |
+-----------+-----------------------------------------+----------------------------+-----------------------------------+-----------------------+
```

There are 18 registered profiles. And the instances are in the interop namespace.

We see the Array registered profile is the first entry (Array+1.8.0).

The next step is to get the central instance for the array. We will do that with the following command:

```
pywbemcli> -o mof instance shrub CIM_RegisteredProfile.?  --ac CIM_ElementConformsToProfile -n interop
Pick Instance name to process
0: interop:CIM_RegisteredProfile.InstanceID="Array+1.8.0"
1: interop:CIM_RegisteredProfile.InstanceID="Block Server Performance+1.8.0"
2: interop:CIM_RegisteredProfile.InstanceID="Block Services+1.8.0"
3: interop:CIM_RegisteredProfile.InstanceID="DMTF Profile Registration+1.0"
4: interop:CIM_RegisteredProfile.InstanceID="Disk Drive Lite+1.8.0"
5: interop:CIM_RegisteredProfile.InstanceID="FC Target Ports+1.8.0"
6: interop:CIM_RegisteredProfile.InstanceID="IP Interface 1.1.1"
7: interop:CIM_RegisteredProfile.InstanceID="Indications+1.2.2"
8: interop:CIM_RegisteredProfile.InstanceID="Job Control 1.0.0"
9: interop:CIM_RegisteredProfile.InstanceID="Multiple Computer System+1.2.0"
10: interop:CIM_RegisteredProfile.InstanceID="Physical Package+1.8.0"
11: interop:CIM_RegisteredProfile.InstanceID="Profile Registration+1.7.0"
12: interop:CIM_RegisteredProfile.InstanceID="Role Based Authorization 1.0.0"
13: interop:CIM_RegisteredProfile.InstanceID="SMI-S+1.8.0"
14: interop:CIM_RegisteredProfile.InstanceID="Server+1.7.0"
15: interop:CIM_RegisteredProfile.InstanceID="Simple Identity Management 1.1.0"
16: interop:CIM_RegisteredProfile.InstanceID="Software+1.8.0"
17: interop:CIM_RegisteredProfile.InstanceID="WBEM Server 1.0.1i"
Input integer between 0 and 17 or Ctrl-C to exit selection:
```

We select the Array profile (item number 0).

```
Input integer between 0 and 17 or Ctrl-C to exit selection: 0
CIM_RegisteredProfile.InstanceID="Array+1.8.0"
 +-- ConformantStandard(Role)
 |   +-- CIM_ElementConformsToProfile(AssocClass)
 |       +-- ManagedElement(ResultRole)
 |           +-- CIM_ComputerSystem(ResultClass)(1 insts)
 |               +-- /device:CIM_ComputerSystem.~,Name="ACME+CF2A5091300089"
+-- ManagedElement(Role)
    +-- CIM_ElementConformsToProfile(AssocClass)
        +-- ConformantStandard(ResultRole)
            +-- CIM_RegisteredProfile(ResultClass)(1 insts)
                +-- /:CIM_RegisteredProfile.InstanceID="SMI-S+1.8.0"
pywbemcli>
```

We want the first association (CIM_ElementConformsToProfile) where the Array profile is the conformant standard. We see the central instance of the array is an instance of CIM_ComputerSystem in the device namespace.

ELEMENTS THAT REPORT CAPACITY INFORMATION

In an Array, there are three elements (CIM_StoragePool, CIM_StorageVolume and CIM_StorageExtent) and one association CIM_AllocatedFromStoragePool that report capacity information. Primordial storage pools and primordial storage extents identify the raw storage available to the Array.

RAW STORAGE

The raw storage capacity available to the array is reported in the Primordial Storage Pool. So, we will begin there. We will look for a primordial storage pool that is hosted on our Array. We will do that with the following command:

```
pywbemcli> -o mof instance shrub CIM_ComputerSystem.? --ac CIM_HostedStoragePool -n device
Pick Instance name to process
0: device:CIM_ComputerSystem.CreationClassName="CIM_ComputerSystem",Name="ACME+CF2A5091300089"
1: device:CIM_ComputerSystem.CreationClassName="CIM_ComputerSystem",Name="ACME+CF2A5091300089+SP_A"
2: device:CIM_ComputerSystem.CreationClassName="CIM_ComputerSystem",Name="ACME+CF2A5091300089+SP_B"
Input integer between 0 and 2 or Ctrl-C to exit selection:
```

Our Array is item number 0 in this list, so we select it.

```
Input integer between 0 and 2 or Ctrl-C to exit selection: 0
CIM_ComputerSystem.CreationClassName="CIM_ComputerSystem",Name="ACME+CF2A5091300089"
 +-- GroupComponent(Role)
    +-- CIM_HostedStoragePool(AssocClass)
       +-- PartComponent(ResultRole)
          +-- CIM_StoragePool(ResultClass)(3 insts)
             +-- /:CIM_StoragePool.InstanceID="ACME+CF2A5091300089+C+0000"
             +-- /:CIM_StoragePool.InstanceID="ACME+CF2A5091300089+C+0001"
             +-- /:CIM_StoragePool.InstanceID="ACME+CF2A5091300089+Primordial"
pywbemcli>
```

It looks like the third one might be our primordial storage pool. But we will verify this with the following command:

```
pywbemcli> instance enumerate CIM_StoragePool --pl
InstanceID,PoolID,Primordial,RemainingManagedSpace,TotalManagedSpace -n device
Instances: CIM_StoragePool
+-----------------------------------------+----------------+----------------+---------------------------------------+---------------------------------+
| InstanceID                              | PoolID         | Primordial     |  RemainingManagedSpace [B] |  TotalManagedSpace [B] |
|-----------------------------------------+----------------+----------------+---------------------------------------+---------------------------------|
| "ACME+CF2A5091300089+C+00"              | "0000"         | false          |                   95289906517 |             766413985109 |
| "00"                                    |                |                |                               |                                 |
| "ACME+CF2A5091300089+C+00"              | "0001"         | false          |                  190950582955 |             191125345621 |
| "01"                                    |                |                |                               |                                 |
| "ACME+CF2A5091300089+Prim"              | "Primordial"   | true           |                 3284276829225 |            4611142320128 |
| "ordial"                                |                |                |                               |                                 |
+-----------------------------------------+----------------+----------------+---------------------------------------+---------------------------------+
pywbemcli>
```

We see that we have 3 storage pools and one of them is primordial (Primordial = true). TotalManagedSpace reports the raw storage in bytes for this Array. The RemainingManagedSpace reports the raw storage in bytes that is available to be allocated to concrete storage pools. The [B] after the property names indicates the value is in Bytes.

CAPACITY ALLOCATED TO CONCRETE POOLS

We also see that we have two concrete (Primordial = false) storage pools. Let's see if they are allocated from our primordial storage pool. We will do this with the following command:

```
pywbemcli> -o mof instance shrub CIM_StoragePool.? --ac CIM_AllocatedFromStoragePool -n device
Pick Instance name to process
0: device:CIM_StoragePool.InstanceID="ACME+CF2A5091300089+C+0000"
1: device:CIM_StoragePool.InstanceID="ACME+CF2A5091300089+C+0001"
2: device:CIM_StoragePool.InstanceID="ACME+CF2A5091300089+Primordial" Input integer between 0 and 2 or Ctrl-C to exit selection:
```

Our primordial pool is item 2, so we select that:

```
Input integer between 0 and 2 or Ctrl-C to exit selection: 2
CIM_StoragePool.InstanceID="ACME+CF2A5091300089+Primordial"
 +-- Antecedent(Role)
    +-- CIM_AllocatedFromStoragePool(AssocClass)
       +-- Dependent(ResultRole)
          +-- CIM_StoragePool(ResultClass)(2 insts)
             +-- /:CIM_StoragePool.InstanceID="ACME+CF2A5091300089+C+0000"
             +-- /:CIM_StoragePool.InstanceID="ACME+CF2A5091300089+C+0001"
pywbemcli>
```

Here we see both our concrete storage pools are allocated directly from the primordial pool.

From the previous command, we have the capacity information for the concrete pools. But there is one more piece of information that we need. That is how much storage they draw from the primordial pool to affect their capacity. We do that with the following command:

```
pywbemcli> instance get CIM_AllocatedFromStoragePool.? --pl Antecedent,Dependent,SpaceConsumed -n device
Pick Instance name to process
0:
device:CIM_AllocatedFromStoragePool.Antecedent="device:CIM_StoragePool.InstanceID=\"ACME+CF2A5091300089+C+0000\"",Dependent="device:CIM_StorageVolume.CreationClassName=\"CIM_StorageVolume\",DeviceID=\"00005\",SystemCreationClassName=\"CIM_ComputerSystem\",SystemName=\"ACME+CF2A5091300089\""
1:
device:CIM_AllocatedFromStoragePool.Antecedent="device:CIM_StoragePool.InstanceID=\"ACME+CF2A5091300089+Primordial\"",Dependent="device:CIM_StoragePool.InstanceID=\"ACME+CF2A5091300089+C+0000\""
2:
device:CIM_AllocatedFromStoragePool.Antecedent="device:CIM_StoragePool.InstanceID=\"ACME+CF2A5091300089+Primordial\"",Dependent="device:CIM_StoragePool.InstanceID=\"ACME+CF2A5091300089+C+0001\""
Input integer between 0 and 2 or Ctrl-C to exit selection:
```

We see the concrete pools are items 1 and 2. So, let's see the first one:

```
Input integer between 0 and 2 or Ctrl-C to exit selection: 1
Instances: CIM_AllocatedFromStoragePool
+-----------------------------------------------------------+-----------------------------------------------------------+----------------------+
| Antecedent                                                | Dependent                                                 |SpaceConsumed |
|                                                           |                                                           | [B]                  |
|-----------------------------------------------------------+-----------------------------------------------------------+----------------------|
|"/device:CIM_StoragePool.InstanceID=\"ACM"                 |"/device:CIM_StoragePool.InstanceID=\"ACM"                 | 948437306573 |
| "E+CF2A5091300089+Primordial\""                           | "E+CF2A5091300089+C+0000\""                               |                      |
+-----------------------------------------------------------+-----------------------------------------------------------+----------------------+
pywbemcli>
```

We see that concrete pool 0, consumed 948,437,306,573 bytes (883 GB) to support its 766,413,985,109 bytes (714 GB) of storage.

And we run the same command for the second concrete pool:

```
pywbemcli> instance get CIM_AllocatedFromStoragePool.? --pl Antecedent,Dependent,SpaceConsumed -n device
Pick Instance name to process
0:
device:CIM_AllocatedFromStoragePool.Antecedent="device:CIM_StoragePool.InstanceID=\"ACME+CF2A5091300089+C+0000\"",Dependent="device:CIM_StorageVolume.CreationClassName=\"CIM_StorageVolume\",DeviceID=\"00005\",SystemCreationClassName=\"CIM_ComputerSystem\",SystemName=\"ACME+CF2A5091300089\""
1:
device:CIM_AllocatedFromStoragePool.Antecedent="device:CIM_StoragePool.InstanceID=\"ACME+CF2A5091300089+Primordial\"",Dependent="device:CIM_StoragePool.InstanceID=\"ACME+CF2A5091300089+C+0000\""
2:
device:CIM_AllocatedFromStoragePool.Antecedent="device:CIM_StoragePool.InstanceID=\"ACME+CF2A5091300089+Primordial\"",Dependent="device:CIM_StoragePool.InstanceID=\"ACME+CF2A5091300089+C+0001\""
Input integer between 0 and 2 or Ctrl-C to exit selection:
```

```
Input integer between 0 and 2 or Ctrl-C to exit selection: 2
Instances: CIM_AllocatedFromStoragePool
+-----------------------------------------------------------+-----------------------------------------------------------+----------------------+
| Antecedent                                                | Dependent                                                 |SpaceConsumed |
|                                                           |                                                           | [B]                  |
|-----------------------------------------------------------+-----------------------------------------------------------+----------------------|
|"/device:CIM_StoragePool.InstanceID=\"ACM"                 |"/device:CIM_StoragePool.InstanceID=\"ACM"                 | 378428184330 |
| "E+CF2A5091300089+Primordial\""                           | "E+CF2A5091300089+C+0001\""                               |                      |
+-----------------------------------------------------------+-----------------------------------------------------------+----------------------+
pywbemcli>
```

We see that concrete pool 1, consumed 378,428,184,330 bytes (352 GB) to support its 191,125,345,621 bytes (178 GB) of storage.

You may have to iterate on allocated from storage pools to see all the pools that are built on other pools. Eventually you will find Storage Volumes allocated out of the concrete pools.

Let's look at what, if anything, is allocated out of our concrete pools using a shrub command:

```
pywbemcli> -o mof instance shrub CIM_StoragePool.? --ac CIM_AllocatedFromStoragePool -n device
Pick Instance name to process
0: device:CIM_StoragePool.InstanceID="ACME+CF2A5091300089+C+0000"
1: device:CIM_StoragePool.InstanceID="ACME+CF2A5091300089+C+0001"
2: device:CIM_StoragePool.InstanceID="ACME+CF2A5091300089+Primordial"
Input integer between 0 and 2 or Ctrl-C to exit selection:
```

We will look at the 0 concrete pool first (selecting 0):

```
Input integer between 0 and 2 or Ctrl-C to exit selection: 0
CIM_StoragePool.InstanceID="ACME+CF2A5091300089+C+0000"
 +-- Antecedent(Role)
 |   +-- CIM_AllocatedFromStoragePool(AssocClass)
 |       +-- Dependent(ResultRole)
 |           +-- CIM_StorageVolume(ResultClass)(1 insts)
 |               +-- /:CIM_StorageVolume.~,DeviceID="00005",~,~
 +-- Dependent(Role)
     +-- CIM_AllocatedFromStoragePool(AssocClass)
         +-- Antecedent(ResultRole)
             +-- CIM_StoragePool(ResultClass)(1 insts)
                 +-- /:CIM_StoragePool.InstanceID="ACME+CF2A5091300089+Primordial"
pywbemcli>
```

The first association shows a Storage Volume has been allocated out of concrete pool 0.

Now let's look to see if there is anything allocated out of concrete pool 1:

```
pywbemcli> -o mof instance shrub CIM_StoragePool.? --ac CIM_AllocatedFromStoragePool -n device
Pick Instance name to process
0: device:CIM_StoragePool.InstanceID="ACME+CF2A5091300089+C+0000"
1: device:CIM_StoragePool.InstanceID="ACME+CF2A5091300089+C+0001"
2: device:CIM_StoragePool.InstanceID="ACME+CF2A5091300089+Primordial"
Input integer between 0 and 2 or Ctrl-C to exit selection:
```

We will look at the concrete pool 1 by selecting 1:

```
Input integer between 0 and 2 or Ctrl-C to exit selection: 1
CIM_StoragePool.InstanceID="ACME+CF2A5091300089+C+0001"
 +-- Dependent(Role)
     +-- CIM_AllocatedFromStoragePool(AssocClass)
         +-- Antecedent(ResultRole)
             +-- CIM_StoragePool(ResultClass)(1 insts)
                 +-- /:CIM_StoragePool.InstanceID="ACME+CF2A5091300089+Primordial"
pywbemcli>
```

We see that concrete pool 1 is allocated out of the primordial pool, but there is nothing allocated out of concrete pool 1.

VOLUME CAPACITY

Next, we will look at Volume capacity. This represents the capacity that is exposed by the Array for use by application systems. We will start with a simple enumeration:

```
pywbemcli> instance enumerate CIM_StorageVolume --pl
InstanceID,Name,BlockSize,NumberOfBlocks,ConsumableBlocks -n device
Instances: CIM_StorageVolume
+----------------+------------+-------------------+------------------------+---------------------------+
| InstanceID   | Name     |   BlockSize [B] |   NumberOfBlocks |   ConsumableBlocks |
|----------------+------------+-------------------+------------------------+---------------------------|
| "SV5"          | "00005" |               512 |                692060 |                  692060 |
+----------------+------------+-------------------+------------------------+---------------------------+
pywbemcli>
```

First, we notice storage volumes (as well as storage extents) report capacity in terms of NumberOfBlocks and BlockSize, instead of bytes. Storage Volume 5 has a size of 354,334,720 bytes (330 MB).

Now let's see how much Storage Volume 5 consumes out of concrete pool 0.

```
pywbemcli> instance get CIM_AllocatedFromStoragePool.? --pl Antecedent,Dependent,SpaceConsumed -n
device
Pick Instance name to process
0:
device:CIM_AllocatedFromStoragePool.Antecedent="device:CIM_StoragePool.InstanceID=\"ACME+CF2A509
1300089+C+0000\"",Dependent="device:CIM_StorageVolume.CreationClassName=\"CIM_StorageVolume\",
DeviceID=\"00005\",SystemCreationClassName=\"CIM_ComputerSystem\",SystemName=\"ACME+CF2A509
1300089\""
1:
device:CIM_AllocatedFromStoragePool.Antecedent="device:CIM_StoragePool.InstanceID=\"ACME+CF2A509
1300089+Primordial\"",Dependent="device:CIM_StoragePool.InstanceID=\"ACME+CF2A5091300089+C+000
0\""
2:
device:CIM_AllocatedFromStoragePool.Antecedent="device:CIM_StoragePool.InstanceID=\"ACME+CF2A509
1300089+Primordial\"",Dependent="device:CIM_StoragePool.InstanceID=\"ACME+CF2A5091300089+C+000
1\""
Input integer between 0 and 2 or Ctrl-C to exit selection:
```

The storage volume is item 0, so we select that:

```
Input integer between 0 and 2 or Ctrl-C to exit selection: 0
Instances: CIM_AllocatedFromStoragePool
+-----------------------------------------------------------+-----------------------------------------------------------+----------------------+
| Antecedent                                                | Dependent                                                 |SpaceConsumed |
|                                                           |                                                           |  [B]                 |
|-----------------------------------------------------------+-----------------------------------------------------------+----------------------|
|"/device:CIM_StoragePool.InstanceID=\"ACM"      |"/device:CIM_StorageVolume.CreationClassN "        |      354334802 |
| " E+CF2A5091300089+C+0000\""                               |"ame=\"CIM_StorageVolume\",DeviceID=\"00005\""|                      |
|                                                           |",SystemCreationClassName=\"CIM_ComputerSyste"|                      |
|                                                           |"m\",SystemName=\"ACME+CF2A5091300089\""     |                      |
+-----------------------------------------------------------+-----------------------------------------------------------+----------------------+
pywbemcli>
```

We see that Storage Volume 5, consumed 354,334,802 bytes to support its 354,334,720 bytes of storage.

DISK DRIVE CAPACITY

Finally, we should look at the disk drive capacities. They are reported as primordial storage extents. We find them by following CIM_AssociatedComponentExtent from the primordial storage pool to the disk drive extents using the following command:

```
pywbemcli> -o mof instance shrub CIM_StoragePool.? --ac CIM_AssociatedComponentExtent -n
device
Pick Instance name to process
0: device:CIM_StoragePool.InstanceID="ACME+CF2A5091300089+C+0000"
1: device:CIM_StoragePool.InstanceID="ACME+CF2A5091300089+C+0001"
2: device:CIM_StoragePool.InstanceID="ACME+CF2A5091300089+Primordial"
Input integer between 0 and 2 or Ctrl-C to exit selection:
```

We select item 2 (the primordial storage pool):

```
Input integer between 0 and 2 or Ctrl-C to exit selection: 2
CIM_StoragePool.InstanceID="ACME+CF2A5091300089+Primordial"
 +-- GroupComponent(Role)
    +-- CIM_AssociatedComponentExtent(AssocClass)
       +-- PartComponent(ResultRole)
          +-- CIM_StorageExtent(ResultClass)(8 insts)
             +-- /:CIM_StorageExtent.~,DeviceID="ACME+CF2A5091300089+2;3;;0_0_0;;;0;0;;",~,~
             +-- /:CIM_StorageExtent.~,DeviceID="ACME+CF2A5091300089+2;3;;0_0_1;;;0;0;;",~,~
             +-- /:CIM_StorageExtent.~,DeviceID="ACME+CF2A5091300089+2;3;;0_0_2;;;0;0;;",~,~
             +-- /:CIM_StorageExtent.~,DeviceID="ACME+CF2A5091300089+2;3;;0_0_3;;;0;0;;",~,~
             +-- /:CIM_StorageExtent.~,DeviceID="ACME+CF2A5091300089+2;3;;0_0_4;;;0;0;;",~,~
             +-- /:CIM_StorageExtent.~,DeviceID="ACME+CF2A5091300089+2;3;;0_0_5;;;0;0;;",~,~
             +-- /:CIM_StorageExtent.~,DeviceID="ACME+CF2A5091300089+2;3;;0_0_6;;;0;0;;",~,~
             +-- /:CIM_StorageExtent.~,DeviceID="ACME+CF2A5091300089+2;3;;0_0_7;;;0;0;;",~,~
pywbemcli>
```

We see the primordial storage pool is made up of 8 disk drives. So, let's look at one to determine its capacity information using the following command:

```
pywbemcli> instance get CIM_StorageExtent.? --pl
InstanceID,Name,BlockSize,NumberOfBlocks,ConsumableBlocks -n device
Pick Instance name to process
0:
device:CIM_StorageExtent.CreationClassName="CIM_StorageExtent",DeviceID="ACME+CF2A5091300089+
2;3;;0_0_0;;;0;0;;",SystemCreationClassName="CIM_ComputerSystem",SystemName="ACME+CF2A509130
0089"
1:
device:CIM_StorageExtent.CreationClassName="CIM_StorageExtent",DeviceID="ACME+CF2A5091300089+
2;3;;0_0_1;;;0;0;;",SystemCreationClassName="CIM_ComputerSystem",SystemName="ACME+CF2A509130
0089"
2:
device:CIM_StorageExtent.CreationClassName="CIM_StorageExtent",DeviceID="ACME+CF2A5091300089+
2;3;;0_0_2;;;0;0;;",SystemCreationClassName="CIM_ComputerSystem",SystemName="ACME+CF2A509130
0089"
```

```
3:
device:CIM_StorageExtent.CreationClassName="CIM_StorageExtent",DeviceID="ACME+CF2A5091300089+2;3;;0_0_3;;;0;0;;",SystemCreationClassName="CIM_ComputerSystem",SystemName="ACME+CF2A5091300089"
4:
device:CIM_StorageExtent.CreationClassName="CIM_StorageExtent",DeviceID="ACME+CF2A5091300089+2;3;;0_0_4;;;0;0;;",SystemCreationClassName="CIM_ComputerSystem",SystemName="ACME+CF2A5091300089"
5:
device:CIM_StorageExtent.CreationClassName="CIM_StorageExtent",DeviceID="ACME+CF2A5091300089+2;3;;0_0_5;;;0;0;;",SystemCreationClassName="CIM_ComputerSystem",SystemName="ACME+CF2A5091300089"
6:
device:CIM_StorageExtent.CreationClassName="CIM_StorageExtent",DeviceID="ACME+CF2A5091300089+2;3;;0_0_6;;;0;0;;",SystemCreationClassName="CIM_ComputerSystem",SystemName="ACME+CF2A5091300089"
7:
device:CIM_StorageExtent.CreationClassName="CIM_StorageExtent",DeviceID="ACME+CF2A5091300089+2;3;;0_0_7;;;0;0;;",SystemCreationClassName="CIM_ComputerSystem",SystemName="ACME+CF2A5091300089"
8:
device:CIM_StorageVolume.CreationClassName="CIM_StorageVolume",DeviceID="00005",SystemCreationClassName="CIM_ComputerSystem",SystemName="ACME+CF2A5091300089"
Input integer between 0 and 8 or Ctrl-C to exit selection:
```

Let's look at the first Storage Extent:

```
Input integer between 0 and 8 or Ctrl-C to exit selection: 0
Instances: CIM_StorageExtent
+-----------+-------------------+-------------------------+---------------------------+
| Name      |  BlockSize [B] |   NumberOfBlocks |   ConsumableBlocks |
|-----------+-------------------+-------------------------+---------------------------|
| "0_0_0" |              512 |        1125767168 |          1125767168 |
+-----------+-------------------+-------------------------+---------------------------+
pywbemcli>
```

There are 8 disk drives, each with 1,125,767,168 blocks and the block size is 512. That means there are 4,611,142,320,128 bytes (4.2 TB) on the disk drives supporting the Primordial pool.

CHANGE LOG

1.0.1	10/20/2020	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