INTEROP NAMESPACE

Multiple Namespace Quick Start Guide

APRIL 20, 2023

MICHAEL WALKER

Version 2.0.0

QUICK START GUIDE FOR THE INTEROP NAMESPACE 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 THE INTEROP NAMESPACE

In this guide we will be exploring the interop namespace and explaining why it is important to any CIM implementation. The interop namespace contains common elements for any WBEM Server. It contains profile definitions, accounts for who can access the server, the Roles they play and Indications supported by the server.

This 20-page document highlights information that can be found in about 400 pages of SMI-S, spread across one SMI-S book (the Common Profiles book) and 5 DMTF profile documents.

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
(pip10) 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:

```
(pip10) 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.

REGISTERED PROFILES

The first thing we will look at are instances of CIM_RegisteredProfile. Each instance defines standard behavior supported by the WBEM Server. We ask for four properties and our search will ask for instancs in both the interop and device namespaces.

```
pywbemcli> instance enumerate CIM_RegisteredProfile --pl
RegisteredOrganization,RegisteredName,RegisteredVersion,AdvertiseTypes -n interop,device
Instances: CIM_RegisteredProfile
+-----------+----------------------+-----------------------------+-------------------+-----------------------+
| namespace | RegisteredOrganization | RegisteredName              | RegisteredVersion | AdvertiseTypes        |
|-----------+----------------------+-----------------------------+-------------------+-----------------------|
| interop   | 11 (SNIA)            | "Array"                     | "1.8.0"           | 3 (SLP)               |
| interop   | 11 (SNIA)            | "Block Server Performance"  | "1.8.0"           | 2 (Not Advertised)    |
| interop   | 11 (SNIA)            | "Block Services"            | "1.8.0"           | 2 (Not Advertised)    |
| interop   | 2 (DMTF)             | "Profile Registration"      | "1.0"             | 2 (Not Advertised)    |
| interop   | 11 (SNIA)            | "Disk Drive Lite"           | "1.8.0"           | 2 (Not Advertised)    |
| interop   | 11 (SNIA)            | "FC Target Ports"           | "1.8.0"           | 2 (Not Advertised)    |
| interop   | 2 (DMTF)             | "IP Interface"              | "1.1.1"           |                       |
| interop   | 2 (DMTF)             | "Indications"               | "1.2.2"           | 2 (Not Advertised)    |
| interop   | 2 (DMTF)             | "Job Control"               | "1.0.0"           |                       |
| interop   | 11 (SNIA)            | "Multiple Computer System"  | "1.2.0"           | 2 (Not Advertised)    |
| interop   | 11 (SNIA)            | "Physical Package"          | "1.8.0"           | 2 (Not Advertised)    |
| interop   | 11 (SNIA)            | "Profile Registration"      | "1.7.0"           | 2 (Not Advertised)    |
| interop   | 2 (DMTF)             | "Role Based Authorization"  | "1.0.0"           |                       |
```

```
| interop    | 11 (SNIA)          | "SMI-S"                          | "1.8.0"          | 2 (Not Advertised)|
| interop    | 11 (SNIA)          | "Server"                         | "1.7.0"          | 3 (SLP)           |
| interop    | 2 (DMTF)           |"Simple Identity Management"| "1.1.0"          |                   |
| interop    | 11 (SNIA)          | "Software"                       | "1.8.0"          | 2 (Not Advertised)|
| interop    | 2 (DMTF)           | "WBEM Server"                    | "1.0.1i"         | 3 (SLP)           |
+------------+--------------------+----------------------------------+------------------+-------------------
+pywbemcli>
```

We see there are a lot of profiles (18). Some are autonomous profiles, some are component profiles (that support the autonomous profiles) and one represents SMI-S itself. All of the instances are in the "interop" namespace. The 14th profile in this list is the RegisteredProfile for SMI-S. You can tell which are autonomous profiles because SMI-S requires autonomous profiles be advertised using SLP.

We see that the autonomous profiles in this list are Array, Server and WBEM Server. Technically, the DMTF Profile Registration profile is also an autonomous profile. But in SMI-S, this profile is treated as a component profile.

Of these profiles, the WBEM Server profile and three of its component profiles (IP Interface, Role Based Authorization and Simple Identity Management) were introduced after 1.6.1. Just to verify that all the profiles support SMI-S 1.8.0 we will run the following command:

```
pywbemcli> -o mof instance shrub CIM_RegisteredProfile.? --ac CIM_ElementConformsToProfile -n interop
```

This command is a shrub command. It is looking for everything that is directly related to CIM_RegisteredProfile. The -o mof option says we want to switch to the mof format for the output for this command. The suffix .? means we want to select the registered profile on which to report. We will get a list of registered profiles and we select the one we want. The --ac CIM_ElementConformsToProfile, restricts the report to just items related via CIM_ElementConformsToProfile. And since all the Registered Profiles are in the interop namespace, we limit the command to the interop namespace (-n interop). So, when we run the command we get:

```
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:
```

Since we want to know which profiles conform to SMI-S 1.8.0, we select the SMI-S registered profile (registered profile number 13).

```
Input integer between 0 and 17 or Ctrl-C to exit selection: 13
CIM_ CIM_RegisteredProfile.InstanceID="SMI-S+1.8.0"
 +-- ConformantStandard(Role)
    +-- CIM_ElementConformsToProfile(AssocClass)
       +-- ManagedElement(ResultRole)
          +-- CIM_RegisteredProfile(ResultClass)(17 insts)
             +-- /:CIM_RegisteredProfile.InstanceID="Array+1.8.0"
             +-- /:CIM_RegisteredProfile.InstanceID="Block Server Performance+1.8.0"
             +-- /:CIM_RegisteredProfile.InstanceID="Block Services+1.8.0"
             +-- /:CIM_RegisteredProfile.InstanceID="DMTF Profile Registration+1.0"
             +-- /:CIM_RegisteredProfile.InstanceID="Disk Drive Lite+1.8.0"
             +-- /:CIM_RegisteredProfile.InstanceID="FC Target Ports+1.8.0"
             +-- /:CIM_RegisteredProfile.InstanceID="IP Interface 1.1.1"
             +-- /:CIM_RegisteredProfile.InstanceID="Indications+1.2.2"
             +-- /:CIM_RegisteredProfile.InstanceID="Job Control 1.0.0"
             +-- /:CIM_RegisteredProfile.InstanceID="Multiple Computer System+1.2.0"
             +-- /:CIM_RegisteredProfile.InstanceID="Physical Package+1.8.0"
             +-- /:CIM_RegisteredProfile.InstanceID="Profile Registration+1.7.0"
             +-- /:CIM_RegisteredProfile.InstanceID="Role Based Authorization 1.0.0"
             +-- /:CIM_RegisteredProfile.InstanceID="Server+1.7.0"
             +-- /:CIM_RegisteredProfile.InstanceID="Simple Identity Management 1.1.0"
             +-- /:CIM_RegisteredProfile.InstanceID="Software+1.8.0"
             +-- /:CIM_RegisteredProfile.InstanceID="WBEM Server 1.0.1i"
pywbemcli>
```

We see that all the instances conform to SMI-S 1.8.0.

SMI-S uses something called the "Scoping Methodology". That means the autonomous profiles must have a CIM_ElementConformsToProfile to their central instance. That is a CIM class instance that is central to the implementation of the autonomous profile.

So, let's look at the central instances of our autonomous profiles.

```
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:
```

The first autonomous profile in the list is the Array profile, so we ask for selection 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>
```

For the Array profile, the central instance is an instance of CIM_ComputerSystem. It is sometimes referred to as the top level system of the Array. Note that CIM_ComputerSystem is in the “device” namespace.

The next autonomous profile in the list is the Server Profile.

```
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:
```

For the Server profile, we ask for selection 14:

```
Input integer between 0 and 17 or Ctrl-C to exit selection: 14
CIM_RegisteredProfile.InstanceID="Server+1.7.0"
 +-- ConformantStandard(Role)
 |   +-- CIM_ElementConformsToProfile(AssocClass)
 |       +-- ManagedElement(ResultRole)
 |           +-- CIM_ObjectManager(ResultClass)(1 insts)
 |               +-- /:CIM_ObjectManager.~,Name="ACME:10.336.643.144",~,~
 +-- ManagedElement(Role)
     +-- CIM_ElementConformsToProfile(AssocClass)
         +-- ConformantStandard(ResultRole)
             +-- CIM_RegisteredProfile(ResultClass)(1 insts)
                 +-- /:CIM_RegisteredProfile.InstanceID="SMI-S+1.8.0"
pywbemcli>
```

In this output we are interested in the first association where the ConformantStandard is the Server Profile and the ManagedElement identifies the central instance for the Server Profile. For the Server profile, the central instance is an instance of CIM_ObjectManager.

The last autonomous profile in the list is the DMTF WBEM Server profile.

```
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:
```

For the WBEM Server profile, we ask for selection 17:

```
Input integer between 0 and 17 or Ctrl-C to exit selection: 17
CIM_RegisteredProfile.InstanceID="WBEM Server 1.0.1i"
 +-- ConformantStandard(Role)
 |   +-- CIM_ElementConformsToProfile(AssocClass)
 |       +-- ManagedElement(ResultRole)
 |           +-- CIM_ComputerSystem(ResultClass)(1 insts)
 |               +-- /:CIM_ComputerSystem.~,Name="10.336.643.144"
 +-- ManagedElement(Role)
     +-- CIM_ElementConformsToProfile(AssocClass)
         +-- ConformantStandard(ResultRole)
             +-- CIM_RegisteredProfile(ResultClass)(1 insts)
                 +-- /:CIM_RegisteredProfile.InstanceID="SMI-S+1.8.0"
pywbemcli>
```

For the WBEM Server profile, the central instance is an instance of CIM_ComputerSystem. But it is a different instance than the Array computer system. And this computer system is in the same namespace as its registered profile.

And we see that both the DMTF WBEM Server and the SNIA Server have been implemented. The means the Interop requirements for both servers should be present. The DMTF WBEM Server is new in SMI-S 1.8.0. The SNIA Server was the profile for the Interop Namespace for the first six releases of SMI-S.

Let's start by inspecting what the DMTF WBEM Server standard calls for:

```
pywbemcli> -o mof instance shrub CIM_RegisteredProfile.? --ac CIM_ReferencedProfile -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:
```

For this command we are looking for CIM_ReferencedProfile associations to component profiles from the DMTF WBEM Server profile. So, we select profile number 17.

```
Input integer between 0 and 17 or Ctrl-C to exit selection: 17
CIM_RegisteredProfile.InstanceID="WBEM Server 1.0.1i"
 +-- Antecedent(Role)
    +-- CIM_ReferencedProfile(AssocClass)
       +-- Dependent(ResultRole)
          +-- CIM_RegisteredProfile(ResultClass)(6 insts)
             +-- /:CIM_RegisteredProfile.InstanceID="DMTF Profile Registration+1.0"
             +-- /:CIM_RegisteredProfile.InstanceID="IP Interface 1.1.1"
             +-- /:CIM_RegisteredProfile.InstanceID="Indications+1.2.2"
             +-- /:CIM_RegisteredProfile.InstanceID="Job Control 1.0.0"
             +-- /:CIM_RegisteredProfile.InstanceID="Role Based Authorization 1.0.0"
             +-- /:CIM_RegisteredProfile.InstanceID="Simple Identity Management 1.1.0"
pywbemcli>
```

We see that the DMTF WBEM Server profile has six component profiles (the DMTF Profile Registration, IP Interface, Job Control, Role Based Authorization, Simple Identity Management and Indications profiles).

So, let compare that to the SNIA Server profile component profiles.

```
pywbemcli> -o mof instance shrub CIM_RegisteredProfile.? --ac CIM_ReferencedProfile -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:
```

In this case we are looking for CIM_ReferencedProfile associations to component profiles from the SNIA Server profile. So, we select profile number 14.

```
Input integer between 0 and 17 or Ctrl-C to exit selection: 14
CIM_RegisteredProfile.InstanceID="Server+1.7.0"
 +-- Antecedent(Role)
```

```
    +-- CIM_ReferencedProfile(AssocClass)
       +-- Dependent(ResultRole)
          +-- CIM_RegisteredProfile(ResultClass)(2 insts)
             +-- /:CIM_RegisteredProfile.InstanceID="Indications+1.2.2"
             +-- /:CIM_RegisteredProfile.InstanceID="Profile Registration+1.7.0"
pywbemcli>
```

We see that the SNIA Server profile has two component profiles (the Indications and Profile Registration).

Notice the SNIA Server profile uses the SNIA Profile Registration, whereas the WBEM Server Profile uses the DMTF Profile Registration. The SNIA Profile Registration augments the DMTF Profile Registration by adding the requirement for a CIM_ElementConformsToProfile to the SMI-S RegisteredProfile and Software information about what software supports the interop namespace.

Since the mock server is claiming to support BOTH the DMTF WBEM Server AND the SNIA Server, the SNIA augmentation also applies to the WBEM Server (and vice versa). But SMI-S only requires one of the servers to be implemented.

ACCOUNTS AND IDENTITIES

Accounts and Identities are managed by the DMTF Simple Identity Management profile, which is a new profile (after SMI-S 1.6.1). So, first let's see what the interop namespace has for Accounts with the following command:

```
pywbemcli> instance enumerate CIM_Account --pl
Name,Userid,UserPassword,OrganizationName,RequestedState,EnabledState -n interop,device
Instances: CIM_Account
+-----------+-----------------------------+----------+--------------+------------------+----------------+--------------+
| namespace | Name                        | UserID   | UserPassword | OrganizationName | RequestedState | EnabledState |
|-----------+-----------------------------+----------+--------------+------------------+----------------+--------------|
| interop   | "10.336.643.144+00" | "Jane"   | "0o173"      | "Storage Systems" | 2 (Enabled)    | 2 (Enabled)  |
| interop   | "10.336.643.144+01" | "John"   | "0o710"      | "Client Lib"      | 2 (Enabled)    | 2 (Enabled)  |
+-----------+-----------------------------+----------+--------------+------------------+----------------+--------------+
pywbemcli>
```

We have two accounts (Jane and John). Note that both are in the interop namespace.

Let's see what the accounts are related to:

```
pywbemcli> -o mof instance shrub CIM_Account.? -n interop
Pick Instance name to process
0:
interop:CIM_Account.CreationClassName="CIM_Account",Name="10.336.643.144+00",SystemCreati
onClassName="CIM_ComputerSystem",SystemName="10.336.643.144"
1:
interop:CIM_Account.CreationClassName="CIM_Account",Name="10.336.643.144+01",SystemCreati
onClassName="CIM_ComputerSystem",SystemName="10.336.643.144"
Input integer between 0 and 1 or Ctrl-C to exit selection:
```

We will look at the shrub for the first account:

```
Input integer between 0 and 1 or Ctrl-C to exit selection: 0
CIM_Account.CreationClassName="CIM_Account",Name="10.336.643.144+00",SystemCreationClass
Name="CIM_ComputerSystem",SystemName="10.336.643.144"
 +-- PartComponent(Role)
```

```
 |   +-- CIM_AccountOnSystem(AssocClass)
 |      +-- GroupComponent(ResultRole)
 |         +-- CIM_ComputerSystem(ResultClass)(1 insts)
 |            +-- /:CIM_ComputerSystem.~,Name="10.336.643.144"
 +-- ManagedElement(Role)
    +-- CIM_AssignedIdentity(AssocClass)
       +-- IdentityInfo(ResultRole)
          +-- CIM_Identity(ResultClass)(1 insts)
             +-- /:CIM_Identity.InstanceID="Ident00"
pywbemcli>
```

We see that the CIM_Account is defined on the WBEM Server system and has an assigned identity.

Let's look at the assigned identities.

```
pywbemcli> instance enumerate CIM_Identity  -n interop,device
Instances: CIM_Identity
+-----------------+----------------+--------------------+
| namespace   | InstanceID   | ElementName  |
|-----------------+----------------+--------------------|
| interop         | "Ident00"    | "Jane Doe"       |
| interop         | "Ident01"    | "John Doe"       |
+-----------------+----------------+--------------------+
pywbemcli>
```

We see there are two identities, one for Jane (Jane Doe) and one for John (John Doe). Again, both instances are in the interop namespace.

Let's see what the identities are related to.

```
pywbemcli> -o mof instance shrub CIM_Identity.? -n interop
Pick Instance name to process
0: interop:CIM_Identity.InstanceID="Ident00"
1: interop:CIM_Identity.InstanceID="Ident01"
Input integer between 0 and 1 or Ctrl-C to exit selection:
```

Again, we will look at the shrub for the first identity:

```
Input integer between 0 and 1 or Ctrl-C to exit selection: 0
CIM_Identity.InstanceID="Ident00"
 +-- IdentityInfo(Role)
 |   +-- CIM_AssignedIdentity(AssocClass)
 |      +-- ManagedElement(ResultRole)
 |         +-- CIM_Account(ResultClass)(1 insts)
 |            +-- /:CIM_Account.~,Name="10.336.643.144+00",~,~
 +-- Antecedent(Role)
 |   +-- CIM_ConcreteDependency(AssocClass)
 |      +-- Dependent(ResultRole)
 |         +-- CIM_Role(ResultClass)(1 insts)
 |            +-- /:CIM_Role.~,Name="Admin"
 +-- AffectedElement(Role)
    +-- CIM_ServiceAffectsElement(AssocClass)
```

```
      +-- AffectingElement(ResultRole)
         +-- CIM_AccountManagementService(ResultClass)(1 insts)
            +-- /:CIM_AccountManagementService.~,Name="Account Management Service",~,~
pywbemcli>
```

In addition to being the assigned identity for a CIM_Account, we see that it has a dependency to a CIM_Role and is managed by an account management service.

ROLES

Roles are defined by the Role Based Authorization profile. Another profile that is new in SMI-S 1.7.0 (and SMI-S 1.8.0).

Now let's see what Roles exist with the following command.

```
pywbemcli> instance enumerate CIM_Role -n interop,device
Instances: CIM_Role
+-----------+---------------------+-----------+---------------------+-------------------------------+-------------------------+
| namespace | CreationClassName | Name    | CommonName   | ElementName                | RoleCharacteristics |
|-----------+---------------------+-----------+---------------------+-------------------------------+-------------------------|
| interop   | "CIM_Role"          | "Admin"| "Administrator" | "Storage Administrator" | 2 (Static)              |
| interop   | "CIM_Role"          | "User"   | "User"              | "Storage User"              | 2 (Static)              |
+-----------+---------------------+-----------+---------------------+-------------------------------+-------------------------+
pywbemcli>
```

We see there is a role of "Admin" and a role of "User" (both in the interop namespace).

Let's see what the roles are related to.

```
pywbemcli> -o mof instance shrub CIM_Role.? -n interop
Pick Instance name to process
0: interop:CIM_Role.CreationClassName="CIM_Role",Name="Admin"
1: interop:CIM_Role.CreationClassName="CIM_Role",Name="User"
Input integer between 0 and 1 or Ctrl-C to exit selection:
```

We will look at the shrub for the first role:

```
Input integer between 0 and 1 or Ctrl-C to exit selection: 0
CIM_Role.CreationClassName="CIM_Role",Name="Admin"
 +-- Dependent(Role)
 |   +-- CIM_ConcreteDependency(AssocClass)
 |      +-- Antecedent(ResultRole)
 |         +-- CIM_Identity(ResultClass)(1 insts)
 |            +-- /:CIM_Identity.InstanceID="Ident00"
 +-- OwnedElement(Role)
 |   +-- CIM_OwningCollectionElement(AssocClass)
 |      +-- OwningElement(ResultRole)
 |         +-- CIM_ComputerSystem(ResultClass)(1 insts)
 |            +-- /:CIM_ComputerSystem.~,Name="10.336.643.144"
 +-- DefiningRole(Role)
```

```
 |   +-- CIM_RoleLimitedToTarget(AssocClass)
 |      +-- TargetElement(ResultRole)
 |         +-- CIM_ComputerSystem(ResultClass)(2 insts)
 |            +-- /device:CIM_ComputerSystem.~,Name="ACME+CF2A5091300089"
 |            +-- /:CIM_ComputerSystem.~,Name="10.336.643.144"
+-- AffectedElement(Role)
   +-- CIM_ServiceAffectsElement(AssocClass)
      +-- AffectingElement(ResultRole)
         +-- CIM_RoleBasedAuthorizationService(ResultClass)(1 insts)
            +-- /:CIM_RoleBasedAuthorizationService.~,Name="RBA Service",~,~
pywbemcli>
```

We see that a CIM_ConcreteDependency relates a CIM_Role to CIM_Identity instances that play that role. We also see that the CIM_Role is owned by the server system (via CIM_OwnedElement). And there the role applies to two systems (the Server system and the top level system of the array). And we see that the CIM_Role instance is managed by an instance of CIM_RoleBasedAuthorizationService.

INDICATIONS

Next both the DMTF WBEM Server profile and the SNIA Server profile support the DMTF Indications profile. An indication is an unsolicited message that is sent FROM the WBEM Server to anyone that might be listening for the indication. It is the CIM equivalent of an SNMP alert. The messages supported by a WBEM Server are defined by instances of CIM_IndicationFilter.

Indication filters can be static (pre-defined by the WBEM Server) or dynamic (created by a user of the WBEM Server). In either case, we can see all of them by enumerating instances of CIM_IndicationFilter. So, let's do that.

```
pywbemcli> instance enumerate CIM_IndicationFilter --pl
Name,SystemName,IndividualSubscriptionSupported,SourceNamespaces -n interop,device
Instances: CIM_IndicationFilter
+-----------+-----------------------------------------------+-----------------+-----------------------------------+--------------------------+
| namespace | Name                                          | SystemName      | IndividualSubscriptionSupported | SourceNamespaces |
|-----------+-----------------------------------------------+-----------------+-----------------------------------+--------------------------|
| interop   | "Change in PercentComplete"                   | "10.336.643.144" | true                              | “interop,device”         |
| interop   | "Change in Pool TotalManagedSpace"            | "10.336.643.144" | true                              | “device”                 |
| interop   | "Change in job State"                         | "10.336.643.144" | true                              | “interop,device”         |
| interop   | "Creation of a job"                           | "10.336.643.144" | true                              | “interop,device”         |
| interop   | "Disk Drive Creation"                         | "10.336.643.144" | true                              | “device”                 |
| interop   | "Disk Drive Deletion"                         | "10.336.643.144" | true                              | “device”                 |
| interop   | "Disk Drive Status change"                    | "10.336.643.144" | true                              | “device”                 |
| interop   | "FC Port Creation"                            | "10.336.643.144" | true                              | “device”                 |
| interop   | "FC Port Deletion"                            | "10.336.643.144" | true                              | “device”                 |
| interop   | "FC Port Status change"                       | "10.336.643.144" | true                              | “device”                 |
| interop   | "Job Completed Successfully"                  | "10.336.643.144" | true                              | “interop,device”         |
| interop   | "Job Completed with an "                      | "10.336.643.144" | true                              | “interop,device”         |
|           | "Exception/Error"                             |                 |                                   |                          |
| interop   | "Pool Creation"                               | "10.336.643.144" | true                              | “device”                 |
| interop   | "Pool Deletion"                               | "10.336.643.144" | true                              | “device”                 |
| interop   | "Pool capacity condition has been "           | "10.336.643.144" | true                              | “device”                 |
|           | "cleared"                                     |                 |                                   |                          |
| interop   | "Pool capacity has run out"                   | "10.336.643.144" | true                              | “device”                 |
| interop   | "Pool capacity is running low."               | "10.336.643.144" | true                              | “device”                 |
| interop   | "Status of System Redundancy has "            | "10.336.643.144" | true                              | “device”                 |
|           | "changed"                                     |                 |                                   |                          |
```

```
| interop    | "Storage Volume has failed"          | "10.336.643.144" | true | “device” |
| interop    | "Storage Volume returned to "        | "10.336.643.144" | true | “device” |
|            | "normal service"                     |                  |      |          |
| interop    | "StorageVolume has degraded"         | "10.336.643.144" | true | “device” |
| interop    | "System Creation"                    | "10.336.643.144" | true | “device” |
| interop    | "System Deletion"                    | "10.336.643.144" | true | “device” |
| interop    | "System Status Change"               | "10.336.643.144" | true | “device” |
+------------+--------------------------------------+------------------+------+----------+
pywbemcli>
```

We see that there are 24 indications defined for this WBEM Server. That is there are 24 different events that this WBEM Server may send to clients that are listening for the indications. In this list a number of them are new since 1.6.1. They include the "Change in Pool TotalManagedSpace", "Pool capacity condition has been cleared" and "Storage Volume returned to normal service". Most of the indications are indications that originate from the device namespace, but the “job” indications can come from either namespace.

Next let's look at what elements are related to these indication filters:

```
pywbemcli> -o mof instance shrub CIM_IndicationFilter.? -n interop
Pick Instance name to process
0: interop:CIM_IndicationFilter.CreationClassName="CIM_IndicationFilter",Name="Change in
PercentComplete",SystemCreationClassName="CIM_ComputerSystem",SystemName="10.336.643.144"
1: interop:CIM_IndicationFilter.CreationClassName="CIM_IndicationFilter",Name="Change in Pool
TotalManagedSpace",SystemCreationClassName="CIM_ComputerSystem",SystemName="10.336.643.144"
2: interop:CIM_IndicationFilter.CreationClassName="CIM_IndicationFilter",Name="Change in job
State",SystemCreationClassName="CIM_ComputerSystem",SystemName="10.336.643.144"
3: interop:CIM_IndicationFilter.CreationClassName="CIM_IndicationFilter",Name="Creation of a
job",SystemCreationClassName="CIM_ComputerSystem",SystemName="10.336.643.144"
4: interop:CIM_IndicationFilter.CreationClassName="CIM_IndicationFilter",Name="Disk Drive
Creation",SystemCreationClassName="CIM_ComputerSystem",SystemName="10.336.643.144"
5: interop:CIM_IndicationFilter.CreationClassName="CIM_IndicationFilter",Name="Disk Drive
Deletion",SystemCreationClassName="CIM_ComputerSystem",SystemName="10.336.643.144"
6: interop:CIM_IndicationFilter.CreationClassName="CIM_IndicationFilter",Name="Disk Drive Status
change",SystemCreationClassName="CIM_ComputerSystem",SystemName="10.336.643.144"
7: interop:CIM_IndicationFilter.CreationClassName="CIM_IndicationFilter",Name="FC Port
Creation",SystemCreationClassName="CIM_ComputerSystem",SystemName="10.336.643.144"
8: interop:CIM_IndicationFilter.CreationClassName="CIM_IndicationFilter",Name="FC Port
Deletion",SystemCreationClassName="CIM_ComputerSystem",SystemName="10.336.643.144"
9: interop:CIM_IndicationFilter.CreationClassName="CIM_IndicationFilter",Name="FC Port Status
change",SystemCreationClassName="CIM_ComputerSystem",SystemName="10.336.643.144"
10: interop:CIM_IndicationFilter.CreationClassName="CIM_IndicationFilter",Name="Job Completed
Successfully",SystemCreationClassName="CIM_ComputerSystem",SystemName="10.336.643.144"
11: interop:CIM_IndicationFilter.CreationClassName="CIM_IndicationFilter",Name="Job Completed with an
Exception/Error",SystemCreationClassName="CIM_ComputerSystem",SystemName="10.336.643.144"
12: interop:CIM_IndicationFilter.CreationClassName="CIM_IndicationFilter",Name="Pool
Creation",SystemCreationClassName="CIM_ComputerSystem",SystemName="10.336.643.144"
13: interop:CIM_IndicationFilter.CreationClassName="CIM_IndicationFilter",Name="Pool
Deletion",SystemCreationClassName="CIM_ComputerSystem",SystemName="10.336.643.144"
```

```
14: interop:CIM_IndicationFilter.CreationClassName="CIM_IndicationFilter",Name="Pool capacity condition has
been cleared",SystemCreationClassName="CIM_ComputerSystem",SystemName="10.336.643.144"
15: interop:CIM_IndicationFilter.CreationClassName="CIM_IndicationFilter",Name="Pool capacity has run
out",SystemCreationClassName="CIM_ComputerSystem",SystemName="10.336.643.144"
16: interop:CIM_IndicationFilter.CreationClassName="CIM_IndicationFilter",Name="Pool capacity is running
low.",SystemCreationClassName="CIM_ComputerSystem",SystemName="10.336.643.144"
17: interop:CIM_IndicationFilter.CreationClassName="CIM_IndicationFilter",Name="Status of System
Redundancy has changed",SystemCreationClassName="CIM_ComputerSystem",SystemName="10.336.643.144"
18: interop:CIM_IndicationFilter.CreationClassName="CIM_IndicationFilter",Name="Storage Volume has
failed",SystemCreationClassName="CIM_ComputerSystem",SystemName="10.336.643.144"
19: interop:CIM_IndicationFilter.CreationClassName="CIM_IndicationFilter",Name="Storage Volume returned
to normal service",SystemCreationClassName="CIM_ComputerSystem",SystemName="10.336.643.144"
20: interop:CIM_IndicationFilter.CreationClassName="CIM_IndicationFilter",Name="StorageVolume has
degraded",SystemCreationClassName="CIM_ComputerSystem",SystemName="10.336.643.144"
21: interop:CIM_IndicationFilter.CreationClassName="CIM_IndicationFilter",Name="System
Creation",SystemCreationClassName="CIM_ComputerSystem",SystemName="10.336.643.144"
22: interop:CIM_IndicationFilter.CreationClassName="CIM_IndicationFilter",Name="System
Deletion",SystemCreationClassName="CIM_ComputerSystem",SystemName="10.336.643.144"
23: interop:CIM_IndicationFilter.CreationClassName="CIM_IndicationFilter",Name="System Status
Change",SystemCreationClassName="CIM_ComputerSystem",SystemName="10.336.643.144"
Input integer between 0 and 23 or Ctrl-C to exit selection:
```

We will look at the shrub for item 4 (Disk Drive Creation):

```
Input integer between 0 and 18 or Ctrl-C to exit selection: 4
CIM_IndicationFilter.CreationClassName="CIM_IndicationFilter",Name="Disk Drive
Creation",SystemCreationClassName="CIM_ComputerSystem",SystemName="10.336.643.144"
 +-- Member(Role)
 |   +-- CIM_MemberOfCollection(AssocClass)
 |       +-- Collection(ResultRole)
 |           +-- CIM_FilterCollection(ResultClass)(1 insts)
 |               +-- /:CIM_FilterCollection.InstanceID="10.336.643.144:DDLFilterCollection"
 +-- AffectedElement(Role)
    +-- CIM_ServiceAffectsElement(AssocClass)
        +-- AffectingElement(ResultRole)
            +-- CIM_IndicationService(ResultClass)(1 insts)
                +-- /:CIM_IndicationService.~,Name="Indication Service",~,~
pywbemcli>
```

We see that indication filters are related to two other elements: A FilterCollection and an IndicationService.

Let's first look at the service. For readability, this will be broken into 2 commands.

```
pywbemcli> instance enumerate CIM_IndicationService --pl
Name,DeliveryRetryAttempts,DeliveryRetryInterval,FilterCreationEnabled  -n interop,device
Instances: CIM_IndicationService
```

```
+--------------------------+--------------------------------+--------------------------------+-------------------------------+
| Name                     |  DeliveryRetryAttempts | DeliveryRetryInterval[s] | FilterCreationEnabled  |
|--------------------------+--------------------------------+--------------------------------+-------------------------------|
| "Indication Service" |                3               |                          30 | true                          |
+--------------------------+--------------------------------+--------------------------------+-------------------------------+
pywbemcli>
```

```
pywbemcli> instance enumerate CIM_IndicationService --pl
Name,SubscriptionRemovalAction,SubscriptionRemovalTimeInterval -n interop,device
```

```
Instances: CIM_IndicationService
+-----------+--------------------------+-----------------------------------+-----------------------------------------------+
| namespace | Name                     | SubscriptionRemovalAction  | SubscriptionRemovalTimeInterval [s] |
|-----------+--------------------------+-----------------------------------+-----------------------------------------------|
| interop   | "Indication Service" | 4 (Ignore)                        |                                             0 |
+-----------+--------------------------+-----------------------------------+-----------------------------------------------+

pywbemcli>
```

We see that there will be three attempts to deliver an indication and 30 seconds between the attempts, and filter creation is enabled. We also see that the subscription removal action is 4 (Ignore) and the time interval before removal is 0 seconds. The [s] after DeliveryRetryInterval and SubscriptionRemovalTimeInterval indicates the property units are seconds.

Now let's look at the Filter Collections:

```
pywbemcli> instance enumerate CIM_FilterCollection -n interop,device
```

```
Instances: CIM_FilterCollection
+-----------+-----------------------------------------------------+-----------------------------------------------------------------+
| namespace | InstanceID                                          | CollectionName                                                  |
|-----------+-----------------------------------------------------+-----------------------------------------------------------------|
| interop   | "10.336.643.144:ArrayFilterCollection"          | "SNIA:Array:FilterCollection"                                   |
| interop   | "10.336.643.144:BSFilterCollection"             | "SNIA:Block Services:FilterCollection"                          |
| interop   | "10.336.643.144:DDLFilterCollection"            | "SNIA:Disk Drive Lite:FilterCollection"                         |
| interop   | "10.336.643.144:FCPFilterCollection"            | "SNIA:FC Target Ports:FilterCollection"                         |
| interop   | "10.336.643.144:JobControlFilterCollection" | "DMTF:Job Control:FilterCollection"                             |
| interop   | "10.336.643.144:MCSFilterCollection"            | "SNIA:Multiple Computer System:FilterCollection"                |
| interop   | "10.336.643.144:StaticFilterCollection"         | "StaticFilterCollection"                                        |
+-----------+-----------------------------------------------------+-----------------------------------------------------------------+
pywbemcli>
```

Each of the Indication Filters can be related to a FilterCollection for profiles that support indications, or they may be related to a GlobalFilterCollection. If the filter is related to a specific profile, then it is generated for that specific profile elements. If it is in the Global Filter Collection, it can be generated from multiple profiles.

These filter collections are actually associated to the RegisteredProfiles. We see this from the following command:

```
pywbemcli> -o mof instance shrub CIM_FilterCollection.? --ac CIM_ConcreteDependency -n interop
```

```
Pick Instance name to process
```

```
0: interop:CIM_FilterCollection.InstanceID="10.336.643.144:ArrayFilterCollection"
1: interop:CIM_FilterCollection.InstanceID="10.336.643.144:BSFilterCollection"
2: interop:CIM_FilterCollection.InstanceID="10.336.643.144:DDLFilterCollection"
3: interop:CIM_FilterCollection.InstanceID="10.336.643.144:FCPFilterCollection"
4: interop:CIM_FilterCollection.InstanceID="10.336.643.144:JobControlFilterCollection"
5: interop:CIM_FilterCollection.InstanceID="10.336.643.144:MCSFilterCollection"
6: interop:CIM_FilterCollection.InstanceID="10.336.643.144:StaticFilterCollection"
Input integer between 0 and 6 or Ctrl-C to exit selection:
```

We will see what registered profile for the Array FilterCollection is by selecting 0.

```
Input integer between 0 and 5 or Ctrl-C to exit selection: 0
CIM_FilterCollection.InstanceID="10.336.643.144:ArrayFilterCollection"
 +-- Dependent(Role)
    +-- CIM_ConcreteDependency(AssocClass)
       +-- Antecedent(ResultRole)
          +-- CIM_RegisteredProfile(ResultClass)(1 insts)
             +-- /:CIM_RegisteredProfile.InstanceID="Array+1.8.0"
pywbemcli>
```

And we see the ArrayFilterCollection is associated to the RegisteredProfile for Array via the ConcreteDependency association.

You can look at the indications supported by specific profiles by inspecting the member of collection association from the filter collection.

```
pywbemcli> -o mof instance shrub CIM_FilterCollection.? --ac CIM_MemberOfCollection -n interop
Pick Instance name to process
0: interop:CIM_FilterCollection.InstanceID="10.336.643.144:ArrayFilterCollection"
1: interop:CIM_FilterCollection.InstanceID="10.336.643.144:BSFilterCollection"
2: interop:CIM_FilterCollection.InstanceID="10.336.643.144:DDLFilterCollection"
3: interop:CIM_FilterCollection.InstanceID="10.336.643.144:FCPFilterCollection"
4: interop:CIM_FilterCollection.InstanceID="10.336.643.144:JobControlFilterCollection"
5: interop:CIM_FilterCollection.InstanceID="10.336.643.144:MCSFilterCollection"
6: interop:CIM_FilterCollection.InstanceID="10.336.643.144:StaticFilterCollection"
Input integer between 0 and 6 or Ctrl-C to exit selection:
```

If we select item 2 we will see the indication filters for disk drive lite:

```
Input integer between 0 and 5 or Ctrl-C to exit selection: 2
CIM_FilterCollection.InstanceID="10.336.643.144:DDLFilterCollection"
 +-- Collection(Role)
 |   +-- CIM_MemberOfCollection(AssocClass)
 |      +-- Member(ResultRole)
 |         +-- CIM_IndicationFilter(ResultClass)(3 insts)
 |            +-- /:CIM_IndicationFilter.~,Name="Disk Drive Creation",~,~
 |            +-- /:CIM_IndicationFilter.~,Name="Disk Drive Deletion",~,~
```

```
 |            +-- /:CIM_IndicationFilter.~,Name="Disk Drive Status change",~,~
 +-- Member(Role)
    +-- CIM_MemberOfCollection(AssocClass)
       +-- Collection(ResultRole)
          +-- CIM_FilterCollection(ResultClass)(1 insts)
             +-- /:CIM_FilterCollection.InstanceID="10.336.643.144:StaticFilterCollection"
pywbemcli>
```

And we see the three indication filters for the Disk Drive Lite profile. We also see the disk drive lite filter collection is a member of the static filter collection.

WBEM PROTOCOL MANAGEMENT

Finally, we have the IP Interface profile as a component of the DMTF WBEM Server profile. This profile is required if WBEM Protocol Management is supported. This can be determined by looking for an instance of CIM_WBEMProtocolServiceCapabilities with a value of true for the ListeningPortManagementSupported.

The following command checks to see if this condition is met.

```
pywbemcli> instance enumerate CIM_WBEMProtocolServiceCapabilities --pl
InstanceID,ListeningPortManagementSupported -n interop,device
Instances: CIM_CIMXMLCapabilities
+-----------+-------------------------------+-----------------------------------------+
| namespace | InstanceID                    | ListeningPortManagementSupported        |
|-----------+-------------------------------+-----------------------------------------|
| interop   | "CIMXMLCapabilities Instance" | true                                    |
+-----------+-------------------------------+-----------------------------------------+
pywbemcli>
```

And we see that a CIM_WBEMProtocolServiceCapabilities exists and it has ListeningPortManagementSupported = true.

So, let's see what else we have in the capabilities (we will do this in three requests for readability):

```
pywbemcli> instance enumerate CIM_WBEMProtocolServiceCapabilities --pl
InstanceID,AuthenticationMechanismsSupported,MaxConnections -n interop,device
Instances: CIM_CIMXMLCapabilities
+-----------+-------------------------------+-----------------------------------------+----------------+
| namespace | InstanceID                    | AuthenticationMechanismsSupported       | MaxConnections |
|-----------+-------------------------------+-----------------------------------------+----------------|
| interop   | "CIMXMLCapabilities Instance" | 0 (Unknown)                             |              1 |
+-----------+-------------------------------+-----------------------------------------+----------------+
pywbemcli>
pywbemcli> instance enumerate CIM_WBEMProtocolServiceCapabilities --pl
InstanceID,RequestedStatesSupported,WBEMProtocolVersionsSupported -n interop,device
Instances: CIM_CIMXMLCapabilities
+-----------+-------------------------------+-----------------------------+-----------------------------------+
| namespace | InstanceID                    | RequestedStatesSupported    | WBEMProtocolVersionsSupported     |
|-----------+-------------------------------+-----------------------------+-----------------------------------|
```

```
| interop      | "CIMXMLCapabilities Instance" | 2 (Enabled), 3 (Disabled), 11 (Reset) | "2.0.0"                                    |
+--------------+-------------------------------+---------------------------------------+--------------------------------------------+
pywbemcli>
```

```
pywbemcli> instance enumerate CIM_WBEMProtocolServiceCapabilities --pl
InstanceID,GenericOperationCapabilities -n interop,device
```

```
Instances: CIM_CIMXMLCapabilities
+-------------+-------------------------------+------------------------------------------------------------------------+
| namespace | InstanceID                    | GenericOperationCapabilities                                             |
|-------------+-------------------------------+------------------------------------------------------------------------|
| interop     | "CIMXMLCapabilities Instance" | "instance of CIM_GenericOperationCapabilitiesStructure {\n   "          |
|             |                               | "FQLSupported = false;\n   CQLSupport = { 0 };\n   "                    |
|             |                               | "OperationsSupported = { 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, "|
|             |                               | "15, 16,\n      17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, "   |
|             |                               | "30, 31, 32, 33, 34,\n      35, 36, 37, 38, 39, 40 };\n};\n"           |
+-------------+-------------------------------+------------------------------------------------------------------------+
pywbemcli>
```

We see the AuthenticationMechanismsSupported is unknown, MaxConnections are 1 and the RequestedStatesSupported are Enabled, Disabled and Reset. And we see the ProtocolVersionsSupported is 2.0.0, which means any version 2 release. And we see FQL and CQL are not supported and a whole bunch of generic operations are supported.

Next, we will look at the protocol service:

```
pywbemcli> instance enumerate CIM_ProtocolService --pl
Name,HealthState,MaxConnections,OperationalStatus -n interop,device
```

```
Instances: CIM_ProtocolService
+-------------+-------------------------+-------------+------------------+--------------------+
| namespace | Name                    | HealthState |   MaxConnections | OperationalStatus  |
|-------------+-------------------------+-------------+------------------+--------------------|
| interop     | "CIMXML Protocol Service" | 5 (OK)    |                1 | 2 (OK)             |
+-------------+-------------------------+-------------+------------------+--------------------+
pywbemcli>
```

We see that it is a CIMXML protocol service (as opposed to WS Management) with a maximum of 1 connection. The HealthState and OperationalStatus are OK.

Next, let's see what the protocol service manages.

```
pywbemcli> -o mof instance shrub CIM_ProtocolService.? -n interop
```

```
CIM_ProtocolService.CreationClassName="CIM_ProtocolService",Name="CIMXML Protocol Service",SystemCreationClassName="CIM_ComputerSystem",SystemName="10.336.643.144"
 +-- ManagedElement(Role)
 |   +-- CIM_ElementCapabilities(AssocClass)
 |       +-- Capabilities(ResultRole)
 |           +-- CIM_CIMXMLCapabilities(ResultClass)(1 insts)
 |               +-- /:CIM_CIMXMLCapabilities.InstanceID="CIMXMLCapabilities Instance"
 +-- Dependent(Role)
```

```
|   +-- CIM_HostedService(AssocClass)
|   |   +-- Antecedent(ResultRole)
|   |       +-- CIM_ComputerSystem(ResultClass)(1 insts)
|   |           +-- /:CIM_ComputerSystem.~,Name="10.336.643.144"
|   +-- CIM_ServiceServiceDependency(AssocClass)
|       +-- Antecedent(ResultRole)
|           +-- CIM_WBEMServer(ResultClass)(1 insts)
|               +-- /:CIM_WBEMServer.~,Name="WBEM Server Instance",~,~
+-- Antecedent(Role)
    +-- CIM_ServiceAccessBySAP(AssocClass)
        +-- Dependent(ResultRole)
            +-- CIM_TCPProtocolEndpoint(ResultClass)(1 insts)
                +-- /:CIM_TCPProtocolEndpoint.~,Name="10.336.643.144+TCP11",~,~
pywbemcli>
```

We see that the protocol service has capabilities (CIM_CIMXMLCapabilities), which we have already looked at. We also see from the CIM_ProtocolService shrub report that the CIM_ProtocolService is managing a CIM_TCPProtocolEndpoint instance.

So, let's look at that.

```
pywbemcli> instance enumerate CIM_TCPProtocolEndpoint --pl
Name,InstanceID,NameFormat,OperationalStatus,ProtocolIFType -n interop,device
Instances: CIM_TCPProtocolEndpoint
+-----------+-----------------------------+------------+-----------------------------------------+--------------------+--------------------+
| namespace | Name                        | InstanceID | NameFormat                              | OperationalStatus  | ProtocolIFType     |
|-----------+-----------------------------+------------+-----------------------------------------+--------------------+--------------------|
| interop   | "10.336.643.144+TCP11"      | "TCPPE11"  | "<PortAddress>+<Protocol><N"            | 2 (OK)             | 4111 (TCP)         |
|           |                             |            | "umber>"                                |                    |                    |
+-----------+-----------------------------+------------+-----------------------------------------+--------------------+--------------------+
pywbemcli>
```

This is the protocol endpoint for communicating to the WBEM Server.

CHANGE LOG

1.0.2	11/17/2020	The initial version based on pywbem defaults
1.1.0	11/07/2021	Changed the default namespace to ‘interop’ Changed references to CIM_System to CIM_ComputerSystem
2.0.0	4/20/2023	Changed for the multiple namespace case