QUICK START GUIDE FOR ARRAY HARDWARE INFORMATION

TABLE OF CONTENTS

Introduction

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. It is a command line interface for accessing any CIM Server. It uses pywbem, an interface for python program access to any CIM Server. The pywbemcli and pywbem are python programs that use a set of python packages. Pywbem and the pywbemcli 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 HARDWARE

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

This 7 page document highlights information that can be found in about 30 pages of SMI-S, the Physical Package in the Common Book and disk drive lite profile in the block book of SMI-S.

In this script we will be working with beta3 of pywbem 1.0.0 and version 0.7.1 of pywbemtools (pywbemcli).

So, let's begin. First, we go to our virtual environment for beta3:

```
C:\Users\FarmerMike>workon beta3
```

```
(beta3) c:\Users\FarmerMike\devenv>
```

We are now in our virtual environment for beta3.

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:

```
(beta3) c:\Users\FarmerMike\devenv>pywbemcli -o table --name ArrayMock
```

```
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 (--name ArrayMock). The command worked and we get a pywbemcli prompt to start entering commands on the ArrayMock.

PHYSICAL PACKAGES

The elements that reports hardware information are instances of CIM_PhysicalPackage (and its subclasses).

So, we start by looking for classes that report hardware information (CIM_PhysicalPackage)

```
pywbemcli> instance enumerate CIM_PhysicalPackage --pl Tag,Manufacturer,Model
Instances: CIM_PhysicalPackage
+--------------------------------------------+--------------------+------------------------------+
| Tag                                        | Manufacturer       | Model                        |
|--------------------------------------------+--------------------+------------------------------|
| "ACME+CF2A5091300089"                      | "ACME Corporation" | "ACME Array 101"             |
| "ACME+CF2A5091300089+0_0+"                 | "ACME Corporation" |"Rack Mounted Disk Chassis"|
| "ACME+CF2A5091300089+0_0+ACME+0_0_0" | "ByteStor"         | "STE30065 ACME300"           |
| "ACME+CF2A5091300089+0_0+ACME+0_0_1" | "ByteStor"         | "STE30065 ACME300"           |
| "ACME+CF2A5091300089+0_0+ACME+0_0_2" | "ByteStor"         | "STE30065 ACME300"           |
| "ACME+CF2A5091300089+0_0+ACME+0_0_3" | "ByteStor"         | "STE30065 ACME300"           |
| "ACME+CF2A5091300089+0_0+ACME+0_0_4" | "ByteStor"         | "STE30065 ACME300"           |
| "ACME+CF2A5091300089+0_0+ACME+0_0_5" | "ByteStor"         | "STE30065 ACME300"           |
| "ACME+CF2A5091300089+0_0+ACME+0_0_6" | "ByteStor"         | "STE30065 ACME300"           |
| "ACME+CF2A5091300089+0_0+ACME+0_0_7" | "ByteStor"         | "STE30065 ACME300"           |
+--------------------------------------------+--------------------+------------------------------+
pywbemcli>
```

There are 10 hardware packages in this Array. We can see the manufacturer and model, but it does not tell us what the hardware does or what it supports.

WHAT THE HARDWARE SUPPORTS

The next question to answer is what part of the Array does the hardware support.

We will determine this by iterating on the following command to determine logical elements supported by the hardware.

```
pywbemcli> -o mof instance shrub CIM_PhysicalPackage.?
Pick Instance name to process
0:
root/cimv2:CIM_PhysicalPackage.CreationClassName="CIM_Chassis",Tag="ACME+CF2A5091300089"
1:
root/cimv2:CIM_PhysicalPackage.CreationClassName="CIM_PhysicalPackage",Tag="ACME+CF2A5091
300089+0_0+ACME+0_0_0"
2:
root/cimv2:CIM_PhysicalPackage.CreationClassName="CIM_PhysicalPackage",Tag="ACME+CF2A5091
300089+0_0+ACME+0_0_1"
```

```
3:
root/cimv2:CIM_PhysicalPackage.CreationClassName="CIM_PhysicalPackage",Tag="ACME+CF2A5091
300089+0_0+ACME+0_0_2"
4:
root/cimv2:CIM_PhysicalPackage.CreationClassName="CIM_PhysicalPackage",Tag="ACME+CF2A5091
300089+0_0+ACME+0_0_3"
5:
root/cimv2:CIM_PhysicalPackage.CreationClassName="CIM_PhysicalPackage",Tag="ACME+CF2A5091
300089+0_0+ACME+0_0_4"
6:
root/cimv2:CIM_PhysicalPackage.CreationClassName="CIM_PhysicalPackage",Tag="ACME+CF2A5091
300089+0_0+ACME+0_0_5"
7:
root/cimv2:CIM_PhysicalPackage.CreationClassName="CIM_PhysicalPackage",Tag="ACME+CF2A5091
300089+0_0+ACME+0_0_6"
8:
root/cimv2:CIM_PhysicalPackage.CreationClassName="CIM_PhysicalPackage",Tag="ACME+CF2A5091
300089+0_0+ACME+0_0_7"
9:
root/cimv2:CIM_PhysicalPackage.CreationClassName="CIM_PhysicalPackage",Tag="ACME+CF2A5091
300089+0_0+"
Input integer between 0 and 9 or Ctrl-C to exit selection:
```

We will start with the first one (selection 0):

```
Input integer between 0 and 9 or Ctrl-C to exit selection: 0
```

```
CIM_PhysicalPackage.CreationClassName="CIM_Chassis",Tag="ACME+CF2A5091300089"
 +-- GroupComponent(Role)
 |   +-- CIM_Container(AssocClass)
 |       +-- PartComponent(ResultRole)
 |           +-- CIM_PhysicalPackage(ResultClass)(1 insts)
 |               +-- /:CIM_PhysicalPackage.~,Tag="ACME+CF2A5091300089+0_0+"
 +-- PartComponent(Role)
 |   +-- CIM_ProductPhysicalComponent(AssocClass)
 |       +-- GroupComponent(ResultRole)
 |           +-- CIM_Product(ResultClass)(1 insts)
 |                                                                                    +--
/:CIM_Product.IdentifyingNumber="ACME+CF2A5091300089",Name="ACME",Vendor="ACME
Corporation",Version="1.1.0"
 +-- Antecedent(Role)
     +-- CIM_SystemPackaging(AssocClass)
         +-- Dependent(ResultRole)
             +-- CIM_ComputerSystem(ResultClass)(1 insts)
                 +-- /:CIM_ComputerSystem.~,Name="ACME+CF2A5091300089"
pywbemcli>
```

The first two associations will be talked about later. The third association (CIM_SystemPackaging) links the physical package to the Array System. Also notice that the CreationClassName is a CIM_Chassis. CIM_Chassis is a subclass of CIM_PhysicalPackage.

Now let's run the same command and select the second item in the list:

```
pywbemcli> -o mof instance shrub CIM_PhysicalPackage.?
Pick Instance name to process
0:
root/cimv2:CIM_PhysicalPackage.CreationClassName="CIM_Chassis",Tag="ACME+CF2A5091300089"
1:
root/cimv2:CIM_PhysicalPackage.CreationClassName="CIM_PhysicalPackage",Tag="ACME+CF2A5091
300089+0_0+ACME+0_0_0"
2:
root/cimv2:CIM_PhysicalPackage.CreationClassName="CIM_PhysicalPackage",Tag="ACME+CF2A5091
300089+0_0+ACME+0_0_1"
3:
root/cimv2:CIM_PhysicalPackage.CreationClassName="CIM_PhysicalPackage",Tag="ACME+CF2A5091
300089+0_0+ACME+0_0_2"
4:
root/cimv2:CIM_PhysicalPackage.CreationClassName="CIM_PhysicalPackage",Tag="ACME+CF2A5091
300089+0_0+ACME+0_0_3"
5:
root/cimv2:CIM_PhysicalPackage.CreationClassName="CIM_PhysicalPackage",Tag="ACME+CF2A5091
300089+0_0+ACME+0_0_4"
6:
root/cimv2:CIM_PhysicalPackage.CreationClassName="CIM_PhysicalPackage",Tag="ACME+CF2A5091
300089+0_0+ACME+0_0_5"
7:
root/cimv2:CIM_PhysicalPackage.CreationClassName="CIM_PhysicalPackage",Tag="ACME+CF2A5091
300089+0_0+ACME+0_0_6"
8:
root/cimv2:CIM_PhysicalPackage.CreationClassName="CIM_PhysicalPackage",Tag="ACME+CF2A5091
300089+0_0+ACME+0_0_7"
9:
root/cimv2:CIM_PhysicalPackage.CreationClassName="CIM_PhysicalPackage",Tag="ACME+CF2A5091
300089+0_0+"
Input integer between 0 and 9 or Ctrl-C to exit selection:
```

And we select item 1:

```
 Input integer between 0 and 9 or Ctrl-C to exit selection: 1
CIM_PhysicalPackage.CreationClassName="CIM_PhysicalPackage",Tag="ACME+CF2A5091300089+0_
0+ACME+0_0_0"
 +-- PartComponent(Role)
  |   +-- CIM_Container(AssocClass)
  |       +-- GroupComponent(ResultRole)
  |           +-- CIM_PhysicalPackage(ResultClass)(1 insts)
 |            +-- /:CIM_PhysicalPackage.~,Tag="ACME+CF2A5091300089+0_0+"
 +-- Antecedent(Role)
    +-- CIM_Realizes(AssocClass)
       +-- Dependent(ResultRole)
          +-- CIM_DiskDrive(ResultClass)(1 insts)
             +-- /:CIM_DiskDrive.~,DeviceID="ACME+0_0_0",~,~
pywbemcli>
```

Again, we will ignore the first association (CIM_Container) which will be covered later. The second association (CIM_Realizes) links the physical package to a Disk Drive. So, the package is a Disk Drive. And it appears the same is true for the next 7 packages.

Now we will at the shrub for the last physical package.

```
pywbemcli> -o mof instance shrub CIM_PhysicalPackage.?
Pick Instance name to process
0:
root/cimv2:CIM_PhysicalPackage.CreationClassName="CIM_Chassis",Tag="ACME+CF2A5091300089"
1:
root/cimv2:CIM_PhysicalPackage.CreationClassName="CIM_PhysicalPackage",Tag="ACME+CF2A5091
300089+0_0+ACME+0_0_0"
2:
root/cimv2:CIM_PhysicalPackage.CreationClassName="CIM_PhysicalPackage",Tag="ACME+CF2A5091
300089+0_0+ACME+0_0_1"
3:
root/cimv2:CIM_PhysicalPackage.CreationClassName="CIM_PhysicalPackage",Tag="ACME+CF2A5091
300089+0_0+ACME+0_0_2"
4:
root/cimv2:CIM_PhysicalPackage.CreationClassName="CIM_PhysicalPackage",Tag="ACME+CF2A5091
300089+0_0+ACME+0_0_3"
5:
root/cimv2:CIM_PhysicalPackage.CreationClassName="CIM_PhysicalPackage",Tag="ACME+CF2A5091
300089+0_0+ACME+0_0_4"
6:
root/cimv2:CIM_PhysicalPackage.CreationClassName="CIM_PhysicalPackage",Tag="ACME+CF2A5091
300089+0_0+ACME+0_0_5"
7:
root/cimv2:CIM_PhysicalPackage.CreationClassName="CIM_PhysicalPackage",Tag="ACME+CF2A5091
300089+0_0+ACME+0_0_6"
8:
root/cimv2:CIM_PhysicalPackage.CreationClassName="CIM_PhysicalPackage",Tag="ACME+CF2A5091
300089+0_0+ACME+0_0_7"
9:
root/cimv2:CIM_PhysicalPackage.CreationClassName="CIM_PhysicalPackage",Tag="ACME+CF2A5091
300089+0_0+"
Input integer between 0 and 9 or Ctrl-C to exit selection:
```

So, we will select item 9:

```
 Input integer between 0 and 9 or Ctrl-C to exit selection: 9
CIM_PhysicalPackage.CreationClassName="CIM_PhysicalPackage",Tag="ACME+CF2A5091300089+0_
0+"
 +-- GroupComponent(Role)
 |   +-- CIM_Container(AssocClass)
 |       +-- PartComponent(ResultRole)
 |           +-- CIM_PhysicalPackage(ResultClass)(8 insts)
 |               +-- /:CIM_PhysicalPackage.~,Tag="ACME+CF2A5091300089+0_0+ACME+0_0_0"
 |               +-- /:CIM_PhysicalPackage.~,Tag="ACME+CF2A5091300089+0_0+ACME+0_0_1"
 |               +-- /:CIM_PhysicalPackage.~,Tag="ACME+CF2A5091300089+0_0+ACME+0_0_2"
 |               +-- /:CIM_PhysicalPackage.~,Tag="ACME+CF2A5091300089+0_0+ACME+0_0_3"
 |               +-- /:CIM_PhysicalPackage.~,Tag="ACME+CF2A5091300089+0_0+ACME+0_0_4"
 |               +-- /:CIM_PhysicalPackage.~,Tag="ACME+CF2A5091300089+0_0+ACME+0_0_5"
 |               +-- /:CIM_PhysicalPackage.~,Tag="ACME+CF2A5091300089+0_0+ACME+0_0_6"
 |               +-- /:CIM_PhysicalPackage.~,Tag="ACME+CF2A5091300089+0_0+ACME+0_0_7"
 +-- PartComponent(Role)
     +-- CIM_Container(AssocClass)
         +-- GroupComponent(ResultRole)
             +-- CIM_PhysicalPackage(ResultClass)(1 insts)
                 +--
/:CIM_PhysicalPackage.CreationClassName="CIM_Chassis",Tag="ACME+CF2A5091300089"
pywbemcli>
```

In this case the physical package shrub ONLY includes the CIM_Container association. The thing it represents is a rack for a set of disk drives (the PartComponents in the first Container association) and the rack itself is contained in the Chassis for the Array.