ARRAY SOFTWARE

Multiple Namespace Quick Start Guide

APRIL 1, 2023

MICHAEL WALKER

Version 2.0.0

QUICK START GUIDE FOR ARRAY SOFTWARE 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 SOFTWARE

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

This 6-page document highlights information that can be found in about 32 pages of SMI-S, the software and software inventory profiles in the Common book and disk drive lite profile in the block book of SMI-S.

In this script we will be working with python 3.8 (or above), pywbem 1.6.1 and version 1.2.0 of pywbemtools (pywbemcli). It is important that you are running with pywbemtools 1.2.0. In developing these scripts, the pywbem team was asked to make enhancements for multiple namespace access. The enhancements are only available in version 1.2.0 and beyond.

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

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

We are now in our virtual environment for mocks.

We will be working with a mock server that supports an SMI-S 1.8.0 Array. 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 (--name 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.

SOFTWARE IDENTITIES

The elements that reports software information are instances of CIM_SoftwareIdentity (and its subclasses).

So, we start by looking for classes that report software information (CIM_SoftwareIdentity)

```
pywbemcli> instance enumerate CIM_SoftwareIdentity --pl
InstanceID,VersionString,Manufacturer,MajorVersion,MinorVersion,RevisionNumber -n
interop,device
Instances: CIM_SoftwareIdentity
+-----------+----------------------------------+---------------+--------------------------+--------------+--------------+----------------+
| namespace | InstanceID                       | VersionString | Manufacturer             | MajorVersion | MinorVersion | RevisionNumber |
| interop   | "ManagementServerSo"             | "V4.6.1.2"    | "SW Inc"                 | 4            | 6            | 1              |
|           | "ftwareIdentity"                 |               |                          |              |              |                |
| device    | "ACME "                          | "01.8.6"      | "ACME Corporation"       |              |              |                |
|           | "Software+CF2A50913"             |               |                          |              |              |                |
|           | "00089"                          |               |                          |              |              |                |
| device    | "ACME+CF2A509130008"             | "2.3.1"       | "ByteStor"               |              |              |                |
|           | "9+ACME+0_0_0"                   |               |                          |              |              |                |
| device    | "ACME+CF2A509130008"             | "2.3.1"       | "ByteStor"               |              |              |                |
|           | "9+ACME+0_0_1"                   |               |                          |              |              |                |
| device    | "ACME+CF2A509130008"             | "2.3.1"       | "ByteStor"               |              |              |                |
|           | "9+ACME+0_0_2"                   |               |                          |              |              |                |
| device    | "ACME+CF2A509130008"             | "2.3.1"       | "ByteStor"               |              |              |                |
|           | "9+ACME+0_0_3"                   |               |                          |              |              |                |
| device    | "ACME+CF2A509130008"             | "2.3.1"       | "ByteStor"               |              |              |                |
|           | "9+ACME+0_0_4"                   |               |                          |              |              |                |
| device    | "ACME+CF2A509130008"             | "2.3.1"       | "ByteStor"               |              |              |                |
|           | "9+ACME+0_0_5"                   |               |                          |              |              |                |
| device    | "ACME+CF2A509130008"             | "2.3.1"       | "ByteStor"               |              |              |                |
|           | "9+ACME+0_0_6"                   |               |                          |              |              |                |
| device    | "ACME+CF2A509130008"             | "2.3.1"       | "ByteStor"               |              |              |                |
|           | "9+ACME+0_0_7"                   |               |                          |              |              |                |
+-----------+----------------------------------+---------------+--------------------------+--------------+--------------+----------------+
pywbemcli>
```

We see we have 10 software elements from 3 software manufacturers. One is in the interop namespace and nine are in the device namespace.

WHAT THE SOFTWARE SUPPORTS

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

We will determine this by iterating on the following command to determine logical elements supported by the software. We will first look at the software in the device namespace.

```
pywbemcli> -o mof instance shrub CIM_SoftwareIdentity.? -n device
Pick Instance name to process
0: device:CIM_SoftwareIdentity.InstanceID="ACME Software+CF2A5091300089"
1: device:CIM_SoftwareIdentity.InstanceID="ACME+CF2A5091300089+ACME+0_0_0"
2: device:CIM_SoftwareIdentity.InstanceID="ACME+CF2A5091300089+ACME+0_0_1"
3: device:CIM_SoftwareIdentity.InstanceID="ACME+CF2A5091300089+ACME+0_0_2"
4: device:CIM_SoftwareIdentity.InstanceID="ACME+CF2A5091300089+ACME+0_0_3"
5: device:CIM_SoftwareIdentity.InstanceID="ACME+CF2A5091300089+ACME+0_0_4"
6: device:CIM_SoftwareIdentity.InstanceID="ACME+CF2A5091300089+ACME+0_0_5"
7: device:CIM_SoftwareIdentity.InstanceID="ACME+CF2A5091300089+ACME+0_0_6"
8: device:CIM_SoftwareIdentity.InstanceID="ACME+CF2A5091300089+ACME+0_0_7"
Input integer between 0 and 8 or Ctrl-C to exit selection:
```

We will start with the first software element by selecting item 0

```
 Input integer between 0 and 9 or Ctrl-C to exit selection: 0
CIM_SoftwareIdentity.InstanceID="ACME Software+CF2A5091300089"
 +-- Antecedent(Role)
 |   +-- CIM_ElementSoftwareIdentity(AssocClass)
 |       +-- Dependent(ResultRole)
|          +-- interop:CIM_RegisteredProfile(ResultClass)(8 insts)
 |              +-- /interop:CIM_RegisteredProfile.InstanceID="Array+1.8.0"
 |              +-- /interop:CIM_RegisteredProfile.InstanceID="Block Server Performance+1.8.0"
 |              +-- /interop:CIM_RegisteredProfile.InstanceID="Block Services+1.8.0"
 |              +-- /interop:CIM_RegisteredProfile.InstanceID="Disk Drive Lite+1.8.0"
 |              +-- /interop:CIM_RegisteredProfile.InstanceID="FC Target Ports+1.8.0"
 |              +-- /interop:CIM_RegisteredProfile.InstanceID="Multiple Computer System+1.2.0"
 |              +-- /interop:CIM_RegisteredProfile.InstanceID="Physical Package+1.8.0"
 |              +-- /interop:CIM_RegisteredProfile.InstanceID="Software+1.8.0"
+-- InstalledSoftware(Role)
    +-- CIM_InstalledSoftwareIdentity(AssocClass)
        +-- System(ResultRole)
            +-- CIM_ComputerSystem(ResultClass)(1 insts)
                +-- /:CIM_ComputerSystem.~,Name="ACME+CF2A5091300089"
pywbemcli>
```

We see the ACME Software has a CIM_ElementSoftwareIdentity to a set of profiles. These are the Array profile and its component profiles. The ACME software is the provider code for the Array. The ACME Software also has a CIM_InstalleddSoftwareIdentity to a Computer System. The system is the top level system of the Array. This means the software is installed on the array.

Let's move on to the second software element in the device namespace:

```
pywbemcli> -o mof instance shrub CIM_SoftwareIdentity.? -n device
Pick Instance name to process
0: device:CIM_SoftwareIdentity.InstanceID="ACME Software+CF2A5091300089"
1: device:CIM_SoftwareIdentity.InstanceID="ACME+CF2A5091300089+ACME+0_0_0"
2: device:CIM_SoftwareIdentity.InstanceID="ACME+CF2A5091300089+ACME+0_0_1"
3: device:CIM_SoftwareIdentity.InstanceID="ACME+CF2A5091300089+ACME+0_0_2"
4: device:CIM_SoftwareIdentity.InstanceID="ACME+CF2A5091300089+ACME+0_0_3"
5: device:CIM_SoftwareIdentity.InstanceID="ACME+CF2A5091300089+ACME+0_0_4"
6: device:CIM_SoftwareIdentity.InstanceID="ACME+CF2A5091300089+ACME+0_0_5"
7: device:CIM_SoftwareIdentity.InstanceID="ACME+CF2A5091300089+ACME+0_0_6"
8: device:CIM_SoftwareIdentity.InstanceID="ACME+CF2A5091300089+ACME+0_0_7"
Input integer between 0 and 8 or Ctrl-C to exit selection:
```

We will now get information for the second software element by selecting item 1

```
Input integer between 0 and 9 or Ctrl-C to exit selection:  1
CIM_SoftwareIdentity.InstanceID="ACME+CF2A5091300089+ACME+0_0_0"
 +-- Antecedent(Role)
    +-- CIM_ElementSoftwareIdentity(AssocClass)
       +-- Dependent(ResultRole)
          +-- CIM_DiskDrive(ResultClass)(1 insts)
             +-- /:CIM_DiskDrive.~,DeviceID="ACME+0_0_0",~,~
pywbemcli>
```

We see the ByteStor Software has a CIM_ElementSoftwareIdentity to a disk drive. The ByteStor software is the firmware for the disk drive.

Notice that the ByteStor Software does NOT have a CIM_InstalleddSoftwareIdentity to anything. This is because it would be redundant with the CIM_ElementSoftwareIdentity. That is, the firmware is on the disk drive.

Let's move on to the software element in the interop namespace:

```
pywbemcli> -o mof instance shrub CIM_SoftwareIdentity.? -n interop
CIM_SoftwareIdentity.InstanceID="ManagementServerSoftwareIdentity"
 +-- Antecedent(Role)
 |   +-- CIM_ElementSoftwareIdentity(AssocClass)
 |      +-- Dependent(ResultRole)
 |         +-- CIM_RegisteredProfile(ResultClass)(9 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="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="WBEM Server 1.0.1i"
+-- InstalledSoftware(Role)
   +-- CIM_InstalledSoftwareIdentity(AssocClass)
      +-- System(ResultRole)
         +-- CIM_ComputerSystem(ResultClass)(1 insts)
            +-- /:CIM_ComputerSystem.~,Name="10.336.643.144"
```

Since there is only one item to select from, we immediately get the results for that one element.

We see the Management Server Software has a CIM_ElementSoftwareIdentity to a set of profiles. These are the WBEM Server and SNIA Server profiles and their component profiles. The Management Server software is the provider code for the interop namespace. The Management Server Software also has a CIM_InstalleddSoftwareIdentity to a System. The system is the system of the server profiles. This means the software is installed on the server system.

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