IBM Flash Management Console 3.9.0

Tools Guide

Copyright Notice

Part Number: 00D2446

Published: June 11, 2014

Table of Contents

Introduction

The ioSphere Management Solution Tools Guide contains documentation on the following topics:

- Unified command-line utilities
- Remote command-line utilities
- SMI-S interface
- SNMP interface

Remote Command-line Utilities

Unlike the binary command-line utilities and the unified command-line utilities which are run on the host where the High IOPS device is installed, the remote command-line utilities can be run on a separate host. The remote utilities communicate with a CIM provider running on the host where the High IOPS device is installed, and they provide a management experience similar to running the command-line utilities on a local host.

NOTE-

IBM recommends using these remote command-line utilities to interface with the SMI-S provider, especially if you are unfamiliar with CIM browsers. If you are familiar with CIM models and browsers (such as YAWN), and you would like to use that method, see See Using a CIM Browser for SMI-S Management on page 53.

Prerequisites

The following software is required to be installed on your local host to run the unified command-line utilities:

- a Python interpreter
- a Python argparse package
- a Python PyWBEM module
- the remote utilities (downloaded as fio-remote-util-<version>.rpm)

In addition, the following software is required to be installed and functioning on the remote host:

- the High IOPS device and its drivers
- a CIM provider

Attention!

The version of the SCOM management pack needs to match the version of the IBM Flash Management Console SNMP agent. SCOM has no restrictions on what VSL version can be used because the SCOM management pack doesn't communicate directly with the VSL driver, accessing the SNMP agent instead.

For details on installing CIM providers on remote hosts see See Installing the SMI-S Provider on Linux on page 20 or See Installing the SMI-S WMI Provider on Windows on page 34. A CIM provider is install by default on ESX/ESXi servers.

Using the Remote Command-line Utilities

You can run the remote command-line utilities by explicitly invoking the Python interpreter:

```
python fio-remote-status.py --server <ip-address>:<port> --username
<username> --password <password> -a
```

Or you can set the utilities to be executable and the run the utilities directly:

```
./fio-remote-status.py --server <ip-address>:<port> --username <username>
--password <password> -a
```

By default, the remote command-line utilities are configured to communicate with VMware ESX/ESXi servers. That is, they are configured to use the fio/cimv2 name space. For example, the two commands below are identical and will only work against the VMware CIM provider:

```
python fio-remote-status.py —server 10.1.100.1 --username root —password
centric
```

```
python fio-remote-status.py —server 10.1.100.1 --username root —password
centric —namespace fio/cimv2
```

If you want to communicate with a CIM provider running on Linux, you should explicitly use the root/fio name space.

```
python fio-remote-status.py —server 10.1.100.2 --username root —password
centric —namespace root/fio
```

Remote Command-line Utility Reference

The following remote command-line utilities are available:

Script	Purpose
fio-remote-attach.py	Makes an High IOPS device available to the OS
fio-remote-beacon.py	Lights the High IOPS device's external LEDs
fio-remote-detach.py	Temporarily removes an High IOPS device from OS access
fio-remote-format.py	Used to perform a low-level format of an High IOPS device
fio-remote-status.py	Displays information about the device
fio-remote-update.py	Upgrades the firmware on the device

NOTE-

There are **`-h`** (Help) and **`-v`** (Version) options for all of the scripts.

Connection Issues

```
$ ./fio-remote-status.py --server 10.10.10.110 -a
... python stack trace ...
pywbem.cim_operations.CIMError: (0, 'Socket error: [Errno 111] Connection
refused')
```

If the connection is refused (see above example), check the following:

- Make sure the remote host where the High IOPS device is installed is on and functioning properly.
- Make sure the SFCB server and SMI-S provider are installed and running on the remote host.
- Make sure the address / host name are correct, including port number (if needed).

fio-remote-attach.py

NOTE-

These Python management scripts run on a remote system.

The fio-attach.py script requires that the High IOPS VSL software be loaded, but with the High IOPS device (s) detached. See See fio-remote-detach.py on page 11 for more information on detaching devices.

Attention!

For full script functionality, including remounting the device, the vCLI MUST be installed on the same remote machine that you run the script on. You can test to see if the vCLI is properly installed by running:

`esxcli --server <server> --username <user> --password <password>`

Correct installation will result in a usage menu for the relevant ESX(i)ESXi host version, otherwise an error message will print. vCLI 5.0 or newer is recommended.

Description

Attaches the High IOPS device and makes it available to the operating system. This creates a block device. You can then add it to ESX(i)ESXi host as a storage area.The script will return one of these three results:

- complete
- failed
- request timed out

Syntax Example

```
fio-attach.py [options] [remote options] <device>
```

where **<device>** is the name of the device node (**/dev/fctx**), where *x* indicates the card number: 0, 1, 2, etc. For example, **/dev/fct0** indicates the first High IOPS device installed on the system. You can use **fio-status.py** to display the device node(s) of installed device(s).

Option	Description
--quiet	Quiet: disables the display of progress.

Remote Options	Description
--timeout	Seconds to wait for request to complete (default:30; a timeout value of 0 means don't wait) **NOTE-** If you have a larger device and/or multiple devices, this process might take more than 30 seconds to complete. If the script does time out, you can run fio-status.py to check the status of the device(s).
--server	Remote host address/dns name (required). This address can include the port, for example: `--server 10.10.1.1:5989`
--username	Remote user (default:anonymous)
--password	Remote user's password (default:none)
--no-ssl	Disable secure HTTP on connection

fio-remote-beacon.py

NOTE-

These Python management scripts run on a remote system.

Description

Lights the High IOPS device's three LEDs to locate the device. You should first detach the High IOPS device and then run **fio-beacon.py.**

Syntax Example

```
fio-beacon.py [options] [remote options] <device>
```

where **<device>** is the name of the device node (**/dev/fctx**), where *x* indicates the card number: 0, 1, 2, etc. For example, **/dev/fct0** indicates the first High IOPS device installed on the system. You can use **fio-status.py** to display the device node(s) of installed device(s).

Options	Description
-0	Off: Turns off the three LEDs.
-1	On: Lights the three LEDs.
-p	Prints the PCI bus ID of the device at <device> to standard output. Usage and error information may be written to standard output rather than to standard error.

Option	Description
--quiet	Quiet: disables the display of progress.

Remote Options	Description
--server	Remote host address/dns name (required). This address can include the port, for example: `--server 10.10.1.1:5989`
--username	Remote user (default:anonymous)
--password	Remote user's password (default:none)
--no-ssl	Disable secure HTTP on connection

fio-remote-detach.py

NOTE-

These Python management scripts run on a remote system.

Attention!

For full script functionality, including unmounting/unclaiming the device, the vCLI MUST be installed on the
same remote machine that you run the script on. You can test to see if the vCLI is properly installed by running:

```
esxcli --server <server> --username <user> --password <password>
```

Correct installation will result in a usage menu for the relevant ESX(i)ESXi host version, otherwise an
error message will print. vCLI 5.0 or newer is recommended.

Attention!

Detaching a device while mounted, or under use, can cause errors, data loss and/or corruption. Make sure the vCLI tools are properly installed to make sure the device is properly unmounted.

Description

Detaches the High IOPS device and removes the corresponding block device. The script will return one of these three results:

- complete
- failed
- request timed out.

Syntax Example

```
fio-detach.py [options] [remote options] <device>
```

where **<device>** is the name of the device node (**/dev/fctx**), where *x* indicates the card number: 0, 1, 2, etc. For example, **/dev/fct0** indicates the first High IOPS device installed on the system. You can use **fio-status.py** to display the device node(s) of installed device(s).

Options	Description
-q	Quiet: Will not display progress.

Remote Options	Description
--timeout	Seconds to wait for request to complete (default:30; a timeout value of 0 means don't wait) **NOTE-** If you have a larger device and/or multiple devices, this process might take more than 30 seconds to complete. If the script does time out, you can run **fio-status.py** to check the status of the device(s).
--server	Remote host address/dns name (required). This address can include the port, for example: `--server 10.10.1.1:5989`
--username	Remote user (default:anonymous)

Remote Options	Description
--password	Remote user's password (default:none)
--no-ssl	Disable secure HTTP on connection

NOTE-

If the device continues to fail to detach, it may be because the High IOPS device is mounted, or some process has the device open.

fio-remote-format.py

NOTE-

These Python management scripts run on a remote system.

Attention!

The **`fio-format.py`** script requires that the High IOPS VSL software be loaded with the High IOPS device(s) detached. Refer to See fio-remote-detach.py on page 11 for details.

Description

Performs a low-level format of the board. The script will return one of these three results:

- complete
- failed
- request timed out.

Attention!

Use this utility with care, as it deletes all user information on the card. You will be prompted as to whether you want to proceed with the format.

NOTE-

VMFS, the default filesystem employed by ESX(i), requires 512 byte sector size.

Syntax Example

```
fio-format.py [options] [remote options] <device>
```

where **`<device>`** is the name of the device node (**`/dev/fctx`**), where x indicates the card number: 0, 1, 2, etc. For example, **`/dev/fct0`** indicates the first High IOPS device installed on the system. You can use **`fio-status.py`** to display the device node(s) of installed device(s).

Options	Description
-b <size B|K>	Set the block (sector) size, in bytes or KiBytes (base 2). The default is 512 bytes. For example: -b 512B or -b 4K (B in 512B is optional). **Attention!** ESX(i) only supports 512b sector sizes for use in VMDKs. Do not format your High IOPS device with any other sector size if you plan to use VMDKs. If you are passing the device through to a VM (using VMDirectPathIO), then the guest VM can use any sector size appropriate for the guest OS. In this case, formatting is done in the guest.
-q	Quiet mode: Disable the display of the progress.
-s <size M|G|T|%>	Set the device capacity as a specific size (in TB, GB, or MB) or as a percentage of the advertised capacity, for example: • T Number of terabytes (TB) to format • G Number of gigabytes (GB) to format • M Number of megabytes (MB) to format • % Percentage, such as 70% (the percent sign must be included).
-y	Auto-answer "yes" to all queries from the application (bypass prompts).

Remote Options	Description
--timeout	Seconds to wait for request to complete (default:60; a timeout value of 0 means don't wait) **NOTE-** If you have a larger device, this process might take more than 60 seconds to complete. If the script does time out, you can run fio-status.py to check the status of the device(s).
--server	Remote host address/dns name (required). This address can include the port, for example: `--server 10.10.1.1:5989`
--username	Remote user (default:anonymous)
--password	Remote user's password (default:none)
--no-ssl	Disable secure HTTP on connection

You must re-attach the device in order to use the High IOPS device. See **fio-attach.py** for details.

fio-remote-status.py

NOTE-

These Python management scripts run on a remote system.

Description

Provides detailed information about the installed devices. This script operates on **/dev/fctX** nodes. The script depends on having the High IOPS VSL software loaded.

Syntax

```
fio-status.py [options] [remote options] [<device>]
```

where **<device>** is the name of the device node (**/dev/fctx**), where *x* indicates the card number: 0, 1, 2, etc. For example, **/dev/fct0** indicates the first High IOPS device installed on the system.

If **<device>** is not specified, **fio-status.py** displays information for all cards in the system.

Options	Description
-c	Count: Report only the number of ioMemory devices installed.
-a	Print all available information for each device.
-l	List: returns the output in a format that reflects the CIM class hierarchy.

Remote Options	Description
--server	Remote host address/dns name (required). This address can include the port, for example: `--server 10.10.1.1:5989`
--username	Remote user (default:anonymous)
--password	Remote user's password (default:none)
--no-ssl	Disable secure HTTP on connection

Output

Basic Information: If no options are used, **fio-status** reports the following basic information:

- Number and type of devices installed in the system
- High IOPS VSL software version

Adapter information:

- Adapter type
- Product number
- External power status
- PCIe power limit threshold (if available)
- Connected High IOPS devices

Block device information:

- Attach status
- Product name
- Product number
- Serial number
- PCIe address and slot
- Firmware version
- Size of the device, out of total capacity
- Internal temperature (average and maximum, since High IOPS VSL software load) in degrees Celsius
- Health status: healthy, nearing wearout, write-reduced or read-only
- Reserve capacity (percentage)
- Warning capacity threshold (percentage)

fio-remote-update.py

Attention!

The **`fio-update-iodrive.py`** remote script is currently unsupported on **ESXi 5.x**. To perform a firmware update on ESXi 5.x, use the **`fio-update-iodrive`** utility.

Description

Attention!

Your High IOPS devices must be detached before running **`fio-update-iodrive.`** See **`fio-detach.py`** for details or **Common Maintenance Tasks** for information on disabling auto-attach.

Attention!

It is extremely important that the power not be turned off during a firmware upgrade, as this could cause device failure. If a UPS is not already in place, consider adding one to the system prior to performing a firmware upgrade.

Attention!

Note that when running multiple upgrades in sequence, it is critical to reboot the system after each upgrade. Otherwise the on-device format will not be changed, and there will be data loss.

Attention!

Do not use this utility to downgrade the High IOPS device to an earlier version of the firmware. Doing so may result in data loss and void your warranty. Contact customer support if you need to downgrade your firmware.

Attention!

Upgrade Path: There is a specific upgrade path that you must take when upgrading High IOPS device. Consult the High IOPS VSL Release Notes for this High IOPS VSL software release before upgrading High IOPS devices.

NOTE-

If you receive an error message when updating the firmware that instructs you to update the midprom information, contact Customer Support.

Syntax

```
fio-update-iodrive.py -d <directory-path> -u <firmware-file.fff> [options]
[remote-options] <device>
```

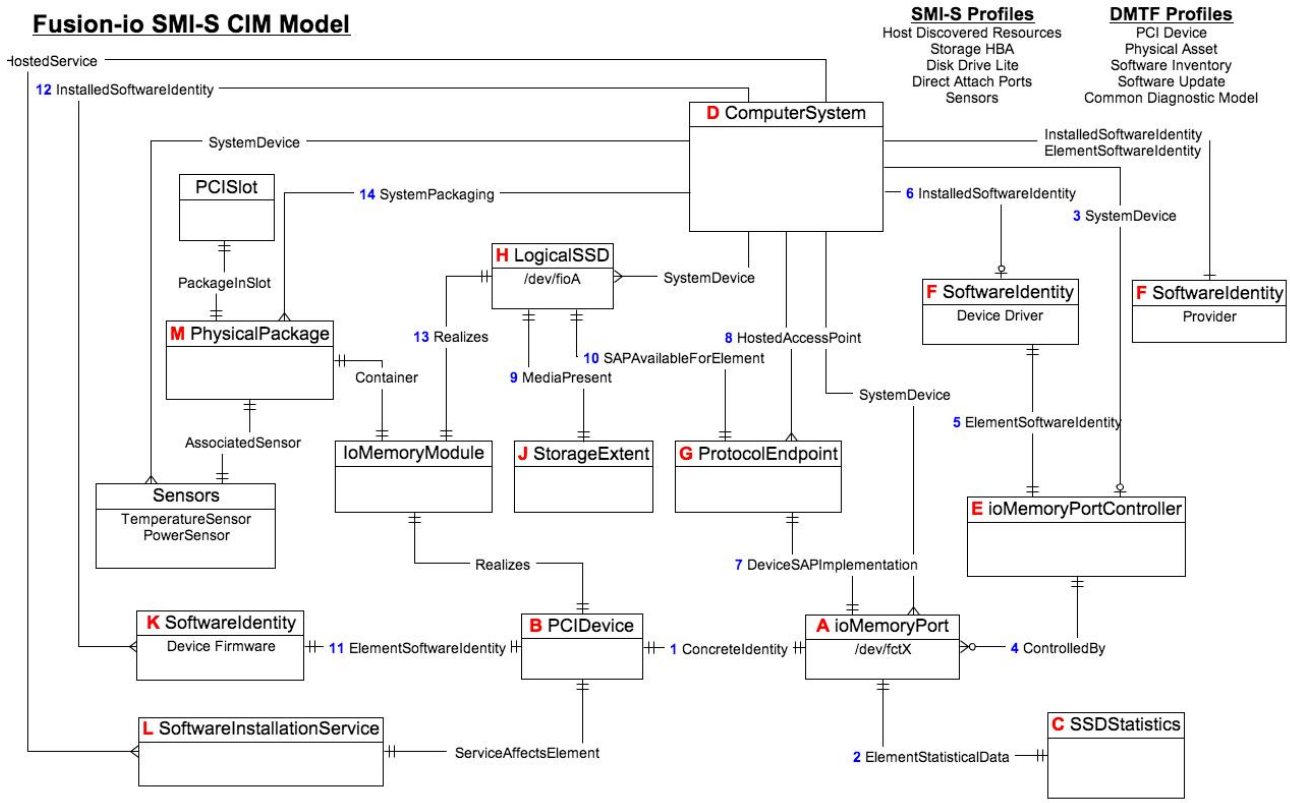
Required Parameters	Description
-d <directory-path>	Directory: Where **<directory-path>** the directory where the firmware file resides. You can transfer the file to a datastore. Example datastore path: **/vmfs/volumes/datastore1/directory-name/**
-u <firmware-file.fff>	Use File: where **<firmware-file.fff>** is the firmware filename. Example: **fusion_ <version>-<date>.fffdell_iodrive_ <version>-<date>.fff cisco_iodrive_ <version>-<date>.fff highiops_ <version>-<date>.fff**
<device>	Where **<device>** is the name of the device node (**/dev/fctx**), where x indicates the card number: 0, 1, 2, etc. For example, **/dev/fct0** indicates the first High IOPS device installed on the system. You can use **fio-status.py** to display the device node(s) of installed device(s).

Options	Description
-f	Force upgrade (used primarily to downgrade to an earlier firmware version).

Options	Description
	Attention! Use the -f option with care, as it could damage your card.
-l	List the firmware available in the archive.
-p	Pretend: Shows what updates would be done. However, the actual firmware is not modified.
-s	Show: Display the current device software version and exit.
-n	No prompt, don't confirm before committing.
-q	Runs the update process without displaying the progress bar or percentage.

Remote Options	Description
--timeout	Seconds to wait for request to complete (a timeout value of 0 means don't wait). Default time is 30 minutes. **NOTE-** If the script does time out, you can run **`fio-status.py`** to check the status of the device(s).
--server	Remote host address/dns name (required). This address can include the port, for example: `--server 10.10.1.1:5989`
--username	Remote user (default:anonymous)
--password	Remote user's password (default:none)
--no-ssl	Disable secure HTTP on connection

SMI-S CIM Model



SMI-S Interface (Linux)

Attention!

With High IOPS VSL software version 3.x and later, the SMI-S provider has a new CIM namespace: **`root/fio`**

Introduction to the SMI-S Interface

The SMI-S interface is based on Web-Based Enterprise Management (WBEM) and provides a Common Information Model (CIM) model that represents the High IOPS device and associated software, in accordance with existing Distributed Management Task Force (DMTF), Storage Networking Industry Association (SNIA), and Storage Management Initiative Specification (SMI-S) standards. This model permits backward-compatible extension, accommodating new hardware and software features developed by .

References

CIM Schema v2.22
http://www.dmtf.org/standards/cim/cim_schema_v2220

DMTF DSP1011, Physical Asset Profile
http://www.dmtf.org/standards/published_documents/DSP1011_1.0.2.pdf

DMTF DSP1023, Software Inventory Profile
http://www.dmtf.org/standards/published_documents/DSP1023_1.0.1.pdf

DMTF DSP1033, Profile Registration Profile
http://www.dmtf.org/standards/published_documents/DSP1033_1.0.0.pdf

DMTF DSP1075 PCI Device Profile
http://www.dmtf.org/standards/published_documents/DSP1075_1.0.0.pdf

DMTF DSP1002, Diagnostics Profile
http://www.dmtf.org/standards/published_documents/DSP1002_2.0.0.pdf

SMI-S v1.4 Architecture
http://www.snia.org/sites/default/files/SMI-Sv1.4r6_Architecture.book_.pdf

SMI-S v1.4 Common Profiles
http://www.snia.org/sites/default/files/SMI-Sv1.4r6_CommonProfiles.book_.pdf

SMI-S v1.4 Host Profiles
http://www.snia.org/sites/default/files/SMI-Sv1.4r6_Host.book_.pdf

SMI-S v1.4 Common Diagnostic Model
http://www.dmtf.org/standards/mgmt/cdm/

Installing the SMI-S Provider on Linux

The SMI-S provider implements a standard WBEM interface based on DMTF and SNIA standards for remote management of ioMemory devices. The provider is a CMPI-based provider and should work with popular CIMOMs including SFCB, OpenPegasus, and OpenWBEM. We also have a version that supports WMI on Windows.

In order to use the provider, a CIMOM must be installed and configured. The provider and associated MOF files must then be installed and registered with the CIMOM. The MOF files define the CIM objects available from the SMI-S provider.

Initially, the provider has been tested with SFCB on Linux and WMI on Windows.

1. Install the SFCB CIM broker (CIMOM).
    - On Debian-like:

```
$ sudo apt-get install sfcb sfcb-test wbemcli sblim-cmpi-base rsync
```

    - Others: Install **sblim-sfcc, sblim-sfcc-devel, cim-schema-2.21.0, sblim-sfcb, sblim-indication-helper, sblim-cmpi-base** and **sblim-testsuite**

NOTE-

RPMs are available for SLES, RHEL, and others on OpenSUSE Build Service.

2. Configure SFCB: Copy the file (**sfcb.cfg**) to **/etc/sfcb**
3. Install SMI-S provider: Install **fio-smis** package from distribution and copy **/usr/lib/fio/libfiosmis.so** to **/usr/lib/sfcb** (or **/usr/lib64/sfcb** as appropriate).
4. Register the SMI-S provider with SFCB:

```
cd /usr/share/fio/cim

sh /usr/share/sblim-cmpi-base/provider-register.sh -r fiosmis.reg -m
cimv226-dmtf.mof -t sfcb -n root/fio -v
```

5. Restart SFCB:

```
$ /etc/init.d/sblim-sfcb restart
```

Linux Testing

The **wbemcli** utility can be used to test the SMI-S provider.

1. Query the provider for the High IOPS VSL software version and the firmware version for each High IOPS device in the system:

```
$ wbemcli -nl ei http://localhost:5988/root/cimv2:FIO_SoftwareIdentity
```

The output should look something like this (values may change as development continues):

```
localhost:5988/root/cimv2:FIO_SoftwareIdentity.InstanceID="Fusion-io
drive driver"
-InstanceID="Fusion-io drive driver"
-TargetTypes=
-OtherExtendedResourceTypeDescription=
-MinExtendedResourceTypeRevisionNumber=
. . .
```

2. Query the SMI-S provider for each High IOPS device's health:

```
wbemcli -nl ei http://localhost:5988/root/cimv2:FIO_IoMemoryPort
```

The output should look something like this (values may change as development continues):

```
localhost:5988/root/cimv2:FIO_
IoMemoryPort.DeviceID="fct1",CreationClassName="FIO_IoMemoryPort",. .
```

3. Query capacity and usage counters of a specific ioMemory device (in this case fct0):

```
$ wbemcli -nl gi
http://localhost:5988/root/cimv2:FIO_SSDStatistics.InstanceID="fct0"
```

The output should look something like this (values may change as development continues):

```
localhost:5988/root/cimv2:FIO_SSDStatistics.InstanceID="fct0"
-InstanceID="fct0"
-WriteOperations=0
-ReadOperations=6887
-PhysicalMBytesWritten=1523769
. . .
```

The Linux SMI-S provider can be tested remotely with the wbemcli utility by replacing localhost in the examples above with the hostname or IP address of the remote host. This method cannot be used to test the Windows SMI-S provider remotely, however, since (of course) Windows doesn't follow the emerging standard.

The SMI-S provider indications can be tested as well.

About SMI-S - Linux

SMI-S is a collection of specifications that traditionally focus on Storage Area Network (SAN) systems based on the SCSI command set, such as Fibre Channel, iSCSI, and SAS. However, the general pattern used to model these storage systems can be applied to solid-state, direct-attached storage systems such as those provided by .

High IOPS devices are modeled using the SMI-S patterns established in the Storage HBA, Direct Attached (DA) Ports, and Host Discovered Resources Profiles. The physical aspects of the ioMemory device and all firmware and High IOPS VSL software are modeled using published DMTF specifications, including the Physical Asset, Software Inventory, PCI Device Profiles, and Common Diagnostic Model Profile.

See See SMI-S CIM Model on page 19. This chart describes the SMI-S CIM model, with High IOPS devices and their associated firmware and software. For simplicity, the prefix FIO_ has been removed from the class names.

A: iOMemoryPort Class

The central instance of the model is of the **`IOMemoryPort`** class (A in the figure), a logical representation of the ioMemory device. It supports the extrinsic methods necessary to provision the drive. An instance of **`PCIDevice`** (B) and **`IOMemoryPort`** exist for each installed ioMemory device, and they are associated with instances of **`ConcreteIdentity`** (1). An instance of **`SSDStatistics`** (C), which contains important performance and capacity data for the device, is associated by an **`ElementStatisticalData`** association (2) to each **`IOMemoryPort`**. **`IOMemoryPort`** is scoped by an instance of the **`ComputerSystem`** class. The **`SystemDevice`** (3) aggregation aggregates **`IOMemoryPort`** within the containing ComputerSystem.

E: iOMemoryPortController Class

An instance of **`IOMemoryPortController`** (E) represents the High IOPS VSL software used to control the installed High IOPS devices. **`IOMemoryPortController`** specializes **`CIM_PortController`**, and it aggregates **`IoMemoryPort`** with the **`ControlledBy`** (4) aggregation. The software version and vendor information are represented by the **`SoftwareIdentity`** (F) instance that is associated to **`IOMemoryPortController`** (E) via **`ElementSoftwareIdentity`** (5). The **`SoftwareIdentity`** that represents the installed High IOPS VSL software is associated to the scoping **`ComputerSystem`** using the **`InstalledSoftwareIdentity`** association (6).

An instance of the **`ProtocolEndpoint`** class (G) represents both ends of the logical data path between the **`IOMemoryPort`** and the solid-state storage. This aspect of the model is derived from the pattern in the DA Ports Profile, where the port is both an initiator and target. **`ProtocolEndpoint`** is associated to the **`IOMemoryPort`** by **`DeviceSAPImplementation`** (7) and to the **`ComputerSystem`** by **`HostedAccessPoint`** (8).

H: LogicalSSD Class (Block Device)

The block device exposed to applications (file systems, database, and logical volume manager) is modeled using an instance of **`LogicalSSD`** (H), a subclass of **`CIM_DiskDrive`**. It is associated with a **`StorageExtent`** (J) using the **`MediaPresent`** association (9), but the **`StorageExtent`** will always be present. It is also associated to the **`ProtocolEndpoint`** (G) representing the **`IOMemoryPort`** using **`SAPAvailableForElement`** (10) and to the scoping **`ComputerSystem`** using **`SystemDevice`** (3).

High IOPS devices, being PCIe devices, are also represented by an instance of the **PCIDevice** class (B). **IOMemoryPort** is an alternate representation of the **PCIDevice** and its associated control device. It is associated to it by the **ConcreteIdentity** association.

K: SoftwareIdentity

The High IOPS VSL software is also represented with **SoftwareIdentity**, which is associated to the **PCIDevice** by the **ElementSoftwareIdentity** association (11). The **SoftwareIdentity** (firmware) is associated to the scoping **ComputerSystem** by the **InstalledSoftwareIdentity** association (12). An instance of **SoftwareInstallationService** (L) is associated with each **PCIDevice**, which can be used to update device firmware.

M: PhysicalPackage

The physical aspects of High IOPS devices are represented by an instance of the **PhysicalPackage** class (M), that is associated to the **PCIDevice** by **Realizes** (13) and to the scoping **ComputerSystem** by **SystemPackaging** (14). The temperature sensors on High IOPS devices are represented by an instance of **TemperatureSensor** (N) and is associated to the **PhysicalPackage** by **AssociatedSensor.**

Implementation - Linux

This section describes the arrangement of instances and associations for the device CIM model. Not all class properties are described in detail. Consult the CIM schema for detailed description of all properties.

The device health is indicated by the value of the **HealthLevel** property. Values include: Healthy, Warning, Reduced Write, and Read Only. These values are mapped to **standardHealthState** values – OK, Degraded/Warning, and Critical Failure – as appropriate.

Extrinsic methods for device provisioning include attach, detach, format, and update. The attach method creates a block device for the High IOPS device. Detach disables the block device. A format option enables users to specify the device size in either megabytes or a percentage. The update method allows users to upgrade the firmware on the device.

Device longevity is indicated by the value of the **HealthPercentage** property. **FlashbackAvailability** indicates whether or not this feature of the High IOPS device is online.

IOMemoryPorts are aggregated by **IOMemoryPortController** via the **ControlledBy** aggregation. Instances of **IOMemoryPort** are associated to their corresponding **PCIDevice** with the **ConcreteIdentity** association. The **IOMemoryPort** is a logical device of the scoping **ComputerSystem** and is indicated as such by the **SystemDevice** aggregation.

Products with two or more High IOPS devices, such as the IBM High IOPS Duo Adapter do appear like two separate High IOPS devices. For products with multiple devices, the **IOMemoryPort** class is extended to include information about the carrier card type, serial number, and external power connection for the product as a whole.

IOMemoryPort

One instance of **IOMemoryPort** exists for each High IOPS device installed in the **ComputerSystem**.

The **LocationIndicator** property reflects the state of the device indicator beacon (e.g., all LEDs on solid). Reading the value gives the current state of the indicator. Writing the value with "On" or "Off" turns the indicator on or off and can be used to determine the device's physical location.

SSDStatistics

One instance of **SSDStatistics** exists for each **IOMemoryPort** instance. Properties of this object provide performance and capacity information. Some of this information is only available when the drive is attached (i.e., the state of the associated **IOMemoryPort** is "Attached").

IOMemoryPortController

Only one instance of **IOMemoryPortContoller** exists, representing the High IOPS VSL software used to control **IOMemoryPorts**. The **IOMemoryPortController** specializes the **CIM_ PortController**. **IOMemoryPortController** is aggregated to the scoping **ComputerSystem** using the **SystemDevice** aggregation. I**OMemoryPortController** is associated with a **SoftwareInventory** instance representing the High IOPS VSL software properties via the **ElementSoftwareIdentity** association.

ProtocolEndpoint

One instance of **ProtocolEndpoint** exists for each instance of **IOMemoryPort**. It is associated to the **IOMemoryPort** using **DeviceSAPImplementation** and to **LogicalSSD** using **SAPAvailableForElement**. Because an **IOMemoryPort** represents both the initiator and target ports, only one **ProtocolEndpoint** per **IOMemoryPort** is needed to model the connection between **IOMemoryPort** and **LogicalSSD.**

LogicalSSD

One instance of **LogicalSSD**, a subclass of **CIM_DiskDrive**, exists for each block device (**/dev/fioX**) exposed by an High IOPS device. Correlatable IDs are used, based on operating system device names. This enables client applications to associate block devices discovered through this model with resources discovered from other SMI-S models instrumented on the host system.

ComputerSystem aggregates **LogicalSSDs** via **SystemDevice**. The **LogicalSSD** instances are associated to their **ProtocolEndpoints** via **SAPAvailableForElement**. If the **IOMemoryPort** associated to the endpoint is not attached, then the Availability property is set to "Off Line," and the **DeviceID** property value is "Unknown."

StorageExtent

One instance of **StorageExtent** is associated with each **LogicalSSD** and represents the logical storage of the associated device.

SoftwareIdentity

One instance of **SoftwareIdentity** exists to represent the High IOPS VSL software. The firmware is also modeled using **SoftwareIdentity** but requires an instance for each High IOPS device installed. The **IsEntity** property has a value of True, indicating that the **SoftwareIdentity** instance corresponds to a discrete copy of the High IOPS VSL software or firmware. The **MajorVersion, MinorVersion, RevisionNumber,** and **BuildNumber** properties convey the driver/firmware version information. The **Manufacturer** property can be used to identify IBM.

Another option for the firmware is to omit the **InstalledSoftwareIdentity** association with **ComputerSystem**, because the firmware is not really installed on **ComputerSystem**. This option would depend on how users want to model the firmware.

SoftwareInstallationService

An instance of **SoftwareInstallationService** exists for each **PCIDevice** and can be used to update the associated device's firmware.

PCIDevice

An instance of **PCIDevice** is instantiated for each High IOPS device (PCIe card) in the computer. Properties are set as follows:

- **BusNumber** – bus number where the PCIe device exists
- **DeviceNumber** – device number assigned to the PCI device for this bus.
- **FunctionNumber** – set to the function number for the PCI device.
- **SubsystemID, SubsystemVendorID, PCIDeviceID, VendorID,** and **RevisionID** are optional but can be populated if values can be extracted from the configuration registers of the PCI device.

PCIDevice is associated with **IOMemoryPort**, its alternate logical representation, using **ConcreteIdentity**. The **PCIDevice** is also associated with **PhysicalPackage,** representing the physical aspects of the High IOPS device, via **Realizes**.

PhysicalPackage

One instance of **PhysicalPackage** exists for each discrete, physical High IOPS device installed in the computer system. The **Manufacturer, Model, SKU, SerialNumber, Version,** and **PartNumber** properties can be used to describe these aspects of the physical card. **PhysicalPackage** is associated with **PCIDevice** via **Realizes** and the scoping **ComputerSystem** via **SystemPackaging.**

TemperatureSensor

One instance of **TemperatureSensor** exists for each **PhysicalPackage**. Temperature information for the drive is stored in the properties of this object.

Diagnostic Test

One instance of **DiagnosticTest** will exist. The **RunDiagnostic()** method will trigger a snapshot of device status for the specified **ManagedElement** which must be an instance of **IoMemoryPort.** The diagnostic run is synchronous and runs instantaneously. The resulting **ConcreteJob** object will associate to the originating **DiagnosticTest** instance and the respective **IoMemoryPort** instance that was specified (see See SMI-S CIM Model on page 19). At this time, **RunDiagnostic()** can only be used with the default **DiagnosticSettingData** provided.

Each run will add a single entry of **DiagnosticSettingDataRecord** and associated **DiagnosticCompletionRecord** in the **DiagnosticLog.** The **RecordData** property of the **DiagnosticCompletionRecord** will record critical device status at the time of the run. The format of the **RecordData** string can be found in the **RecordFormat** property.

The format is a series of status strings, each of which can hold one of the following values delimited by an asterisk (*) character: *Unknown, OK, Warning,* or *Error*. Currently, seven status values are recorded: *WearoutStatus, WritabilityStatus, FlashbackStatus, TemperatureStatus, MinimalModeStatus, PciStatus* and *InternalErrorStatus*. All of these should report *OK* under normal operating conditions.

`WearoutStatus` will be set to *Warning* when less than 10% reserve space is left on the device. It will be set to *Error*. when there is no more reserved space.

`WritabilityStatus` will be set to *Error* whenever the device is write throttling or in read-only mode. This can happen due to a variety of conditions including device wearout and insufficient power. The warnings and errors are:

- **`FlashbackStatus`** will report *Warning* if a catastrophic error causes Flashback protection to be degraded.
- **`TemperatureStatus`** will report *Warning* when the device temperature is nearing the maximum safe temperature and *Error* when the maximum safe temperature is reached or surpassed.
- **`MinimalModeStatus`** will report either *Warning* or *Error* whenever the device is in minimal mode.
- **`PciStatus`** will report *Warning* or *Error* if there are compatibility problems with the host PCIe bus.
- **`InternalErrorStatus`** will report *Error* if there are any internal problems with the ioMemory VSL software.

The **`CompletionState`** property will summarize the results and may be set to *Unknown, OK, Warning* or *Failed*. If any status is in error, the state will report as *Failed*. Otherwise, if there is any warning status, the state will report *Warning*. The **`Message`** property will be set to indicate the appropriate action if there are any warnings or errors.

DiagnosticSetting Data

There is an instance of **`DiagnosticSettingData`** associated with the **`DiagnosticTest`** instance (see See SMI-S CIM Model on page 19). It records the default settings for each call to **`RunDiagnostic`**.

DiagnosticServiceCapabilities

There is an instance of **`DiagnosticServiceCapabilities`** associated with the **`DiagnosticTest`** instance that records the capabilities of the **`DiagnosticTest`** service.

DiagnosticLog

An instance of **`DiagnosticLog`** is associated with the **`DiagnosticTest`** instance and stores the results of each run.

DiagnosticSettingRecord

A copy of the default **`DiagnosticSettingData`** is stored in a **`DiagnosticSettingDataRecord`** each time a diagnostic is run and is associated with an instance of **`DiagnosticCompletionRecord.`**

DiagnosticCompletionRecord

An instance of **`DiagnosticCompletionRecord`** stores the results of each **`RunDiagnostic`** execution. The details are explained in **`DiagnosticTest.`**

RegisteredDiskDriveLiteProfile

Only one instance of this class is needed. It resides in the **/root/interop** namespace and indicates the implementation of the Disk Drive Lite Profile. The following properties are set as follows:

- **InstanceID –** set to *SNIA:DiskDriveLiteProfile-1.4.0*
- **RegisteredOrganization –** set to "11" (SNIA)
- **RegisteredName –** set to *DirectAccess Ports Profile*
- **RegisteredVersion –** set to "1.4.0"

RegisteredDAPortsProfile

Only one instance of this class is needed. It resides in the **/root/interop** namespace and indicates the implementation of the DA Ports Profile. The properties are set as follows:

- **InstanceID** – set to *SNIA:DAPortsProfile-1.4.0*
- **RegisteredOrganization –** set to "11" (SNIA)
- **RegisteredName –** set to *DirectAccess Ports Profile*
- **RegisteredVersion –** set to "1.4.0"

RegisteredStorageHBAProfile

Only one instance of this class is needed. It resides in the **/root/interop** namespace and indicates the implementation of the Storage HBA Profile. The properties are set as follows:

- **InstanceID –** set to *SNIA:StorageHBAProfile-1.4.0*
- **RegisteredOrganization –** set to "11" (SNIA)
- **RegisteredName –** set to *Storage HBA Profile*
- **RegisteredVersion –** set to "1.4.0"

RegisteredHostDiscoveredResourcesProfile

Only one instance of this class is needed. It resides in the **/root/interop** namespace and indicates the implementation of the Host Discovered Resources Profile. The properties are set as follows:

InstanceID – set to *SNIA:HostDiscoveredResourcesProfile-1.2.0*

RegisteredOrganization – set to "11" (SNIA)

RegisteredName – set to *Host Discovered Resources Profile*

RegisteredVersion – set to "1.2.0"

RegisteredPCIDeviceProfile

Only one instance of this class is needed. It resides in the **/root/interop** namespace and indicates the implementation of the PCI Device Profile. The properties are set as follows:

InstanceID – set to *DMTF:DSP1075-PCIDevice-1.0.0a*

RegisteredOrganization – set to "2" (DMTF)

RegisteredName – set to *PCIDevice Profile*

RegisteredVersion – set to "1.0.0a"

RegisteredSoftwareInventoryProfile

Only one instance of this class is needed. It resides in the **/root/interop** namespace and indicates the implementation of the Software Inventory Profile. The properties are set as follows:

InstanceID – set to *DMTF:DSP1023-SoftwareInventory-1.0.1*

RegisteredOrganization – set to "2" (DMTF)

RegisteredName – set to *Software Inventory Profile*

RegisteredVersion – set to "1.0.1"

RegisteredSoftwareUpdateProfile

Only one instance of this class is needed. It resides in the **/root/interop** namespace and indicates the implementation of the Software Update Profile. The properties are set as follows:

InstanceID – set to *DMTF:DSP1023-SoftwareUpdate-1.0.0*

RegisteredOrganization – set to "2" (DMTF)

RegisteredName – set to *Software Update Profile*

RegisteredVersion – set to "1.0.0"

RegisteredPhysicalAssetProfile

Only one instance of this class is needed. It resides in the **/root/interop** namespace and indicates the implementation of the Physical Asset Profile. The properties are set as follows:

InstanceID – set to *DMTF:PhysicalAssetProfile-1.0.2*

RegisteredOrganization – set to "2" (DMTF)

RegisteredName – set to *PhysicalAsset Profile*

RegisteredVersion – set to "1.0.2"

RegisteredSensorsProfile

Only one instance of this class is needed. It resides in the **/root/interop** namespace and indicates the implementation of the Sensors Profile. The properties are set as follows:

InstanceID – set to *SNIA:SensorsProfile-1.0.0*

RegisteredOrganization – set to "11" (SNIA)

RegisteredName – set to *Sensors Profile*

RegisteredVersion – set to "1.0.0"

RegisteredCommonDiagnosticProfile

Only one instance of this class is needed. It will reside in the **/root/interop** namespace and indicate the implementation of the Common Diagnostic Model Profile. The **InstanceID** property will be set to a value of *DMTF:DiagnosticsProfile-2.0.0a*. The **RegisteredOrganization** property will be set to a value of "2" (DMTF). The **RegisteredName** property will be set to a value of *Diagnostics Profile*. The **RegisteredVersion** property will be set to a value of "2.0.0a".

Indications - Linux

An indication is generated periodically when a serious condition exists for a particular High IOPS device. The WBEM provider currently supports six types of indications. They alert users of the SMI-S provider to conditions such as imminent wearout, degradation of writability, degradation of the flashback feature, higher temperature, and internal error states.

The indications will be instances of the **`FIO_AlertIndiecation`** class that simply specializes the **`CIM_AlertIndication`** class.

FIO_AlertIndication

Property	Value
IndicationIdentifier	See below for each type
IndicationTime	Timestamp when sent
AlertingManagedElement	IoMemoryPort.DeviceID=<device ID>
AlertingElementFormat	CIMObjectPath (2)
AlertType	Device Alert (5)
PerceivedSeverity	See below for each type
ProbableCause	See below for each type
SystemCreationClassName	"FIO_AlertIndication"
SystemName	<hostname>
ProviderName	"fiosmis"
CorrelatedIndications	Not used
Description	Class description
OtherAlertType	Not used
OtherSeverity	Not used
ProbableCauseDescription	Not used
EventID	Same as IndicationIdentifier
OwningEntity	<vendor>
MessageID	Not used
Message	Not used
MessageArguments	Not used

Reduced Writability Indication

The High IOPS VSL software can dramatically reduce write throughput to manage device conditions such as excessive wear, high temperature, and insufficient power. The reduced writability indication is generated while the drive is in this mode. If the triggering condition is excessive wear, the **`IoMemoryPort`** health percentage will report 0% health.

Property	Value
IndicationIdentifier	<mfr>":"<hostname>":write"
PerceivedSeverity	Degraded/Warning (3)
ProbableCause	Threshold Crossed (52) Temperature Unacceptable (51) Power Problem (36)

Read-only Indication

When the drive has reached the end-of-life, it can no longer be written to and can only be read from. The read-only indication will be sent when this occurs. The **`IoMemoryPort`** health percentage will continue to report 0% health when this happens.

Property	Value
IndicationIdentifier	<mfr>":"<hostname>":read_only"
PerceivedSeverity	Degraded/Warning (3)
ProbableCause	Threshold Crossed (52)

Wearout Indication

As the drive wears out, this indication is generated as a warning when the drive health percentage drops below 10%, before write throughput is reduced.

Property	Value
IndicationIdentifier	<mfr>":"<hostname>":wearout"
PerceivedSeverity	Degraded/Warning (3)
ProbableCause	Threshold Crossed (52)

Flashback Indication

If a catastrophic part failure degrades the effectiveness of the flashback feature, this indication will be sent.

Property	Value
IndicationIdentifier	<mfr>":"<hostname>":flashback"
PerceivedSeverity	Degraded/Warning (3)
ProbableCause	Loss of Redundancy (88)

High Temperature Indication

This indication will be sent when the temperature of the card becomes excessive.

Property	Value
IndicationIdentifier	<mfr>":"<hostname>":temperature"
PerceivedSeverity	Critical (6)
ProbableCause	Temperature Unacceptable (51)

Error Indication

If the High IOPS VSL software is in an error state the error indication will be sent.

Property	Value
IndicationIdentifier	<mfr>":"<hostname>":error"
PerceivedSeverity	Major (6)
ProbableCause	Other (1)

SMI-S Interface (Windows)

Attention!

With High IOPS VSL software version 3.x and later, the SMI-S provider has a new CIM namespace: **`root/fio`**

Introduction to the SMI-S Interface

The SMI-S interface is based on Web-Based Enterprise Management (WBEM) and provides a Common Information Model (CIM) model that represents the ioMemory device and associated software, in accordance with existing Distributed Management Task Force (DMTF) and Storage Networking Industry Association (SNIA) Storage Management Initiative Specification (SMI-S) standards. This model permits backward-compatible extension, accommodating new hardware and software features developed by IBM.

References

CIM Schema v2.22
http://www.dmtf.org/standards/cim/cim_schema_v2220

DMTF DSP1011, Physical Asset Profile
http://www.dmtf.org/standards/published_documents/DSP1011_1.0.2.pdf

DMTF DSP1023, Software Inventory Profile
http://www.dmtf.org/standards/published_documents/DSP1023_1.0.1.pdf

DMTF DSP1033, Profile Registration Profile
http://www.dmtf.org/standards/published_documents/DSP1033_1.0.0.pdf

DMTF DSP1075 PCI Device Profile
http://www.dmtf.org/standards/published_documents/DSP1075_1.0.0.pdf

DMTF DSP1002, Diagnostics Profile
http://www.dmtf.org/standards/published_documents/DSP1002_2.0.0.pdf

SMI-S v1.4 Architecture
http://www.snia.org/sites/default/files/SMI-Sv1.4r6_Architecture.book.pdf

SMI-S v1.4 Common Profiles
http://www.snia.org/sites/default/files/SMI-Sv1.4r6_CommonProfiles.book.pdf

SMI-S v1.4 Host Profiles
http://www.snia.org/sites/default/files/SMI-Sv1.4r6_Host.book.pdf

SMI-S v1.4 Common Diagnostic Model
http://www.dmtf.org/standards/mgmt/cdm/

Installing the SMI-S WMI Provider on Windows

To install the SMI-S WMI provider on Windows:

1. Go to **Control Panel > Add & Remove Programs.**
2. Right-click **Management and Monitoring Tools** and select **Details**. Make sure the WMI Windows Installer Provider is selected.

The SMI-S WMI provider for High IOPS devices will be installed and the WMI service will be restarted automatically.

Expected Warning Message

When you install the WMI provider, a warning will appear in the Windows event log with the following description:

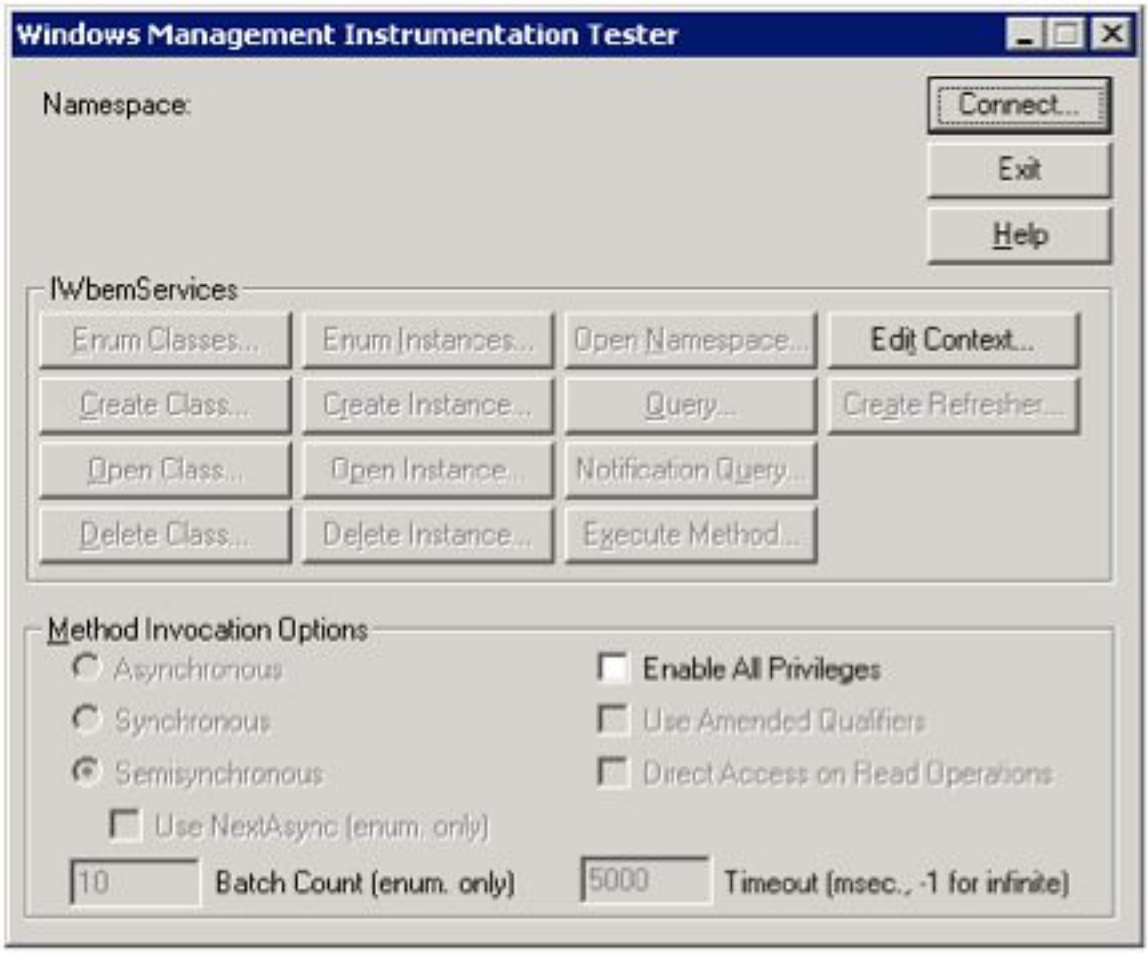
```
A provider, fio-smis-wmi, has been registered in the Windows Management
Instrumentation namespace root\fio to use the LocalSystem account. This
account is privileged and the provider may cause a security violation if
it does not correctly impersonate user requests.
```

This warning is expected. The WMI provider only interfaces with the High IOPS VSL software and does not modif system data.

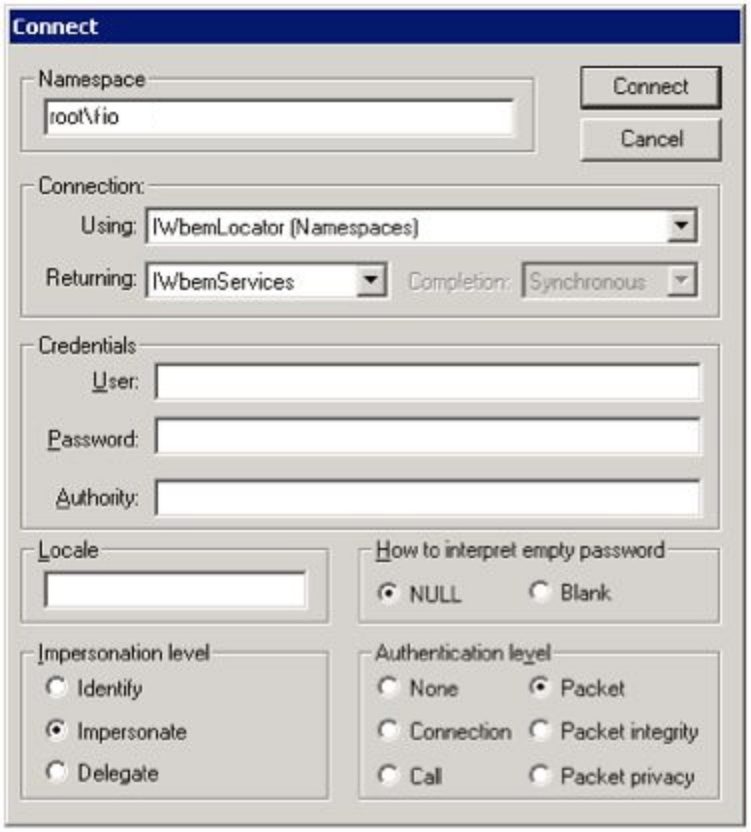
Verifying SMI-S Installation on Windows

To verify the SMI-S WMI provider on Windows:

1. Run the **wbemtest.exe** program. The WMI Tester window appears.

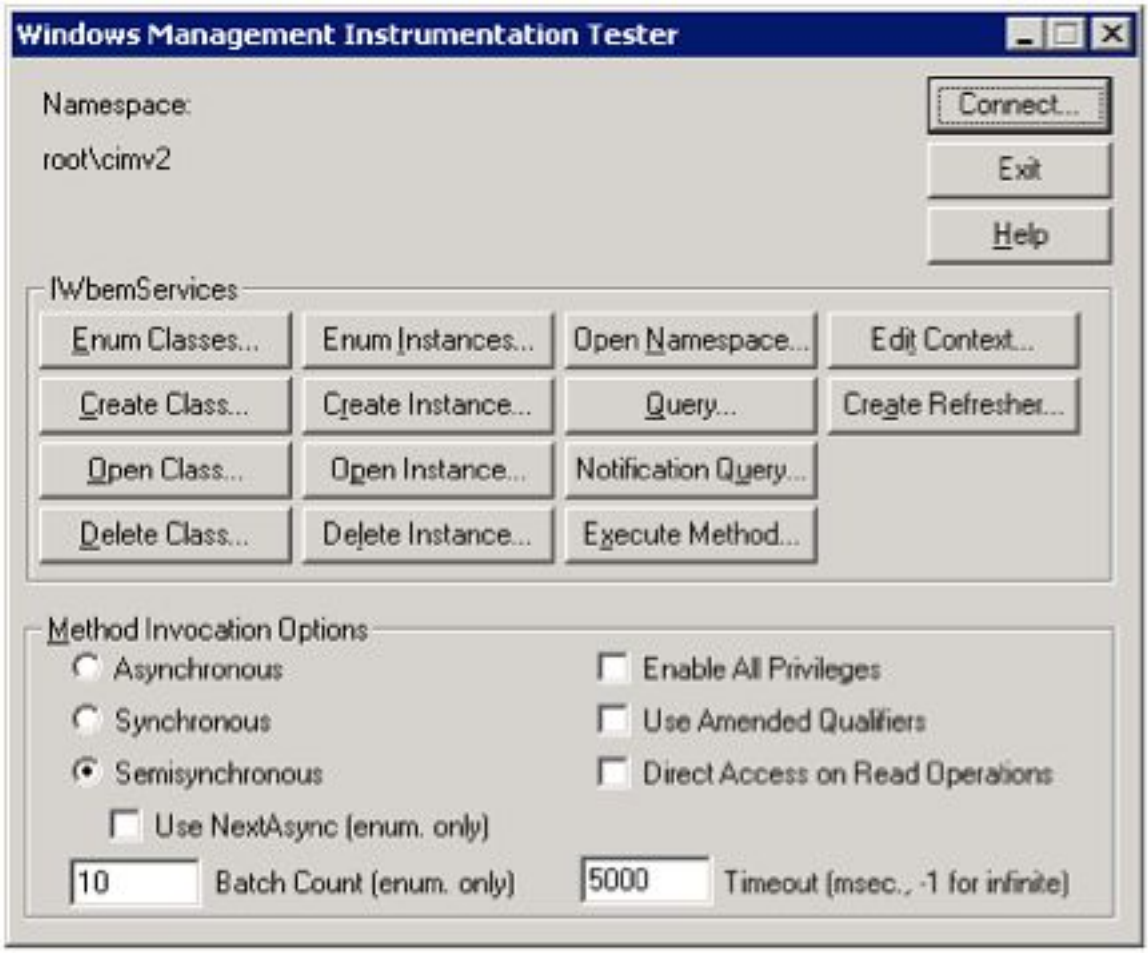


2. Click **Connect** to display the Connect dialog. The CIM provider namespace is **`root\fio`**

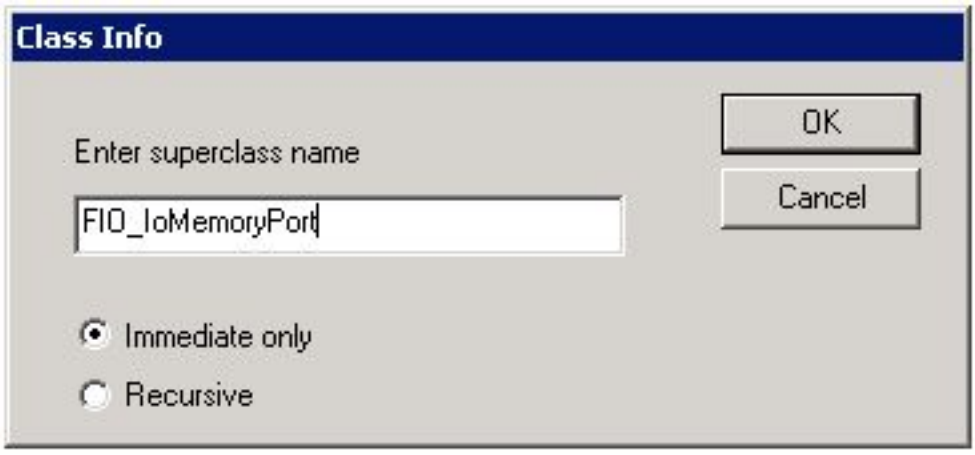


3. Type the namespace value shown in the screenshot above and click **Connect.**

The WMI Tester window appears, with the namespace value filled in.



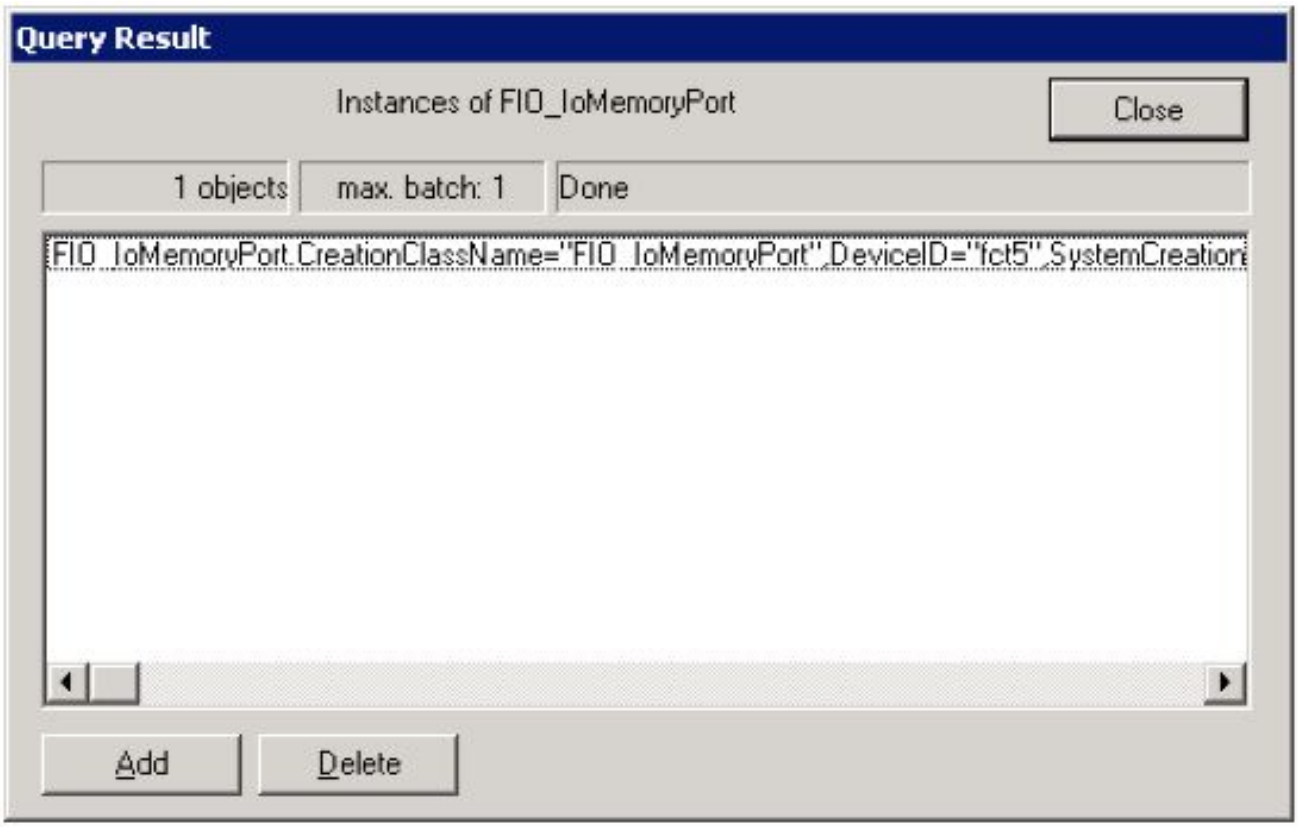
4. Click Enum Instances (second button on the first row) to bring up the **Class Info** dialog.



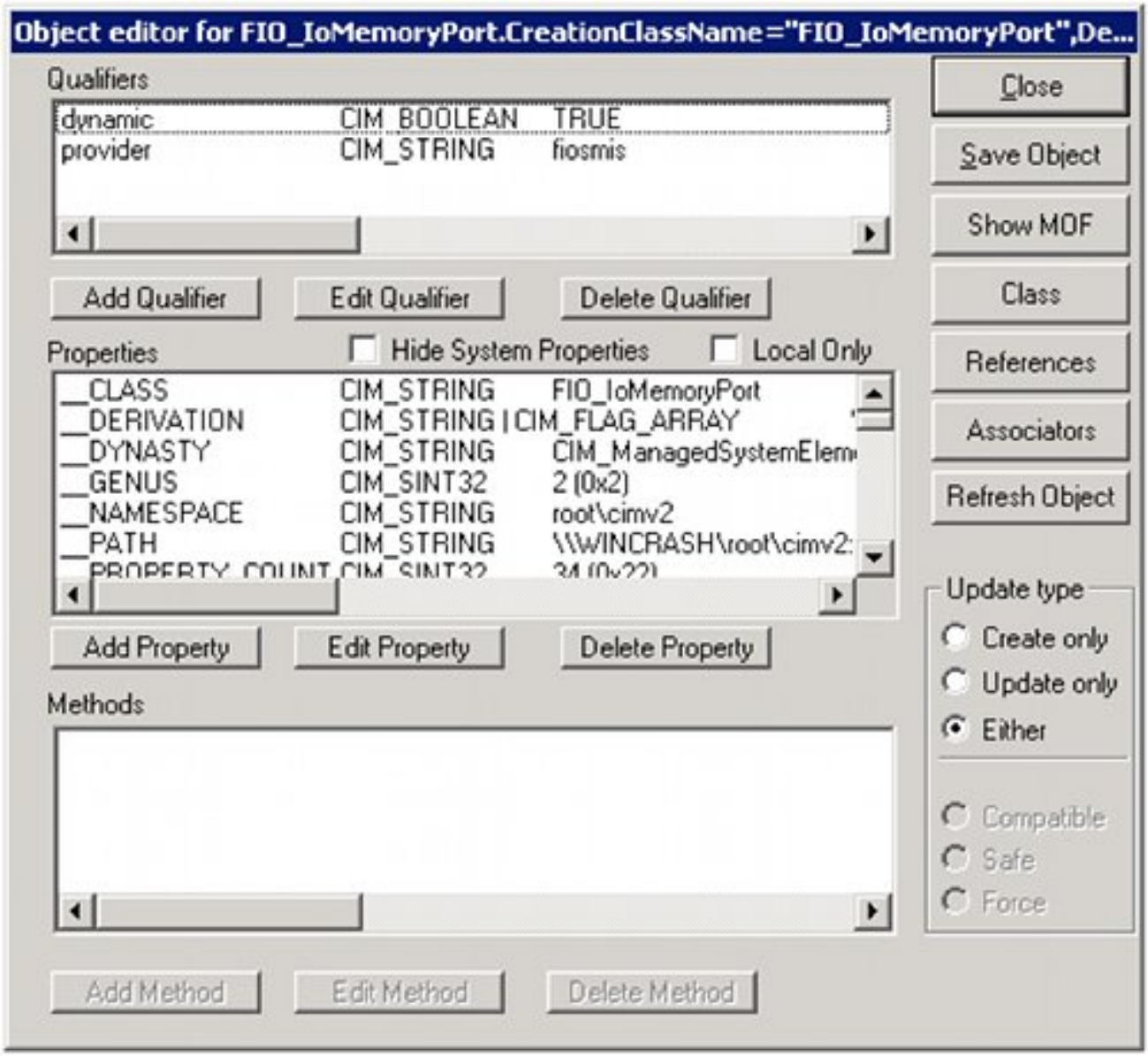
5. Type FIO_IoMemoryPort as shown above and then click OK.

   If the provider is installed correctly, the result will look like the following example, with an entry for

each High IOPS device in the system:



6. Double-click an entry to bring up detailed information, such as in this example:



Manual Registration

If the automatic installation fails to register the provider, follow these steps to manually register it:

1. Stop the WMI (winmgmt) service via the services tool or the following command line:

```
net stop winmgmt
```

2. Browse to the **Fusion-io\SMIS\cim-schema** directory using the command-line interface and run the following:

```
mofcomp fio-reg-wmi.mof
```

3. Browse to **Fusion-io\SMIS\WMI** directory
4. Un-register and re-register the **fio-smis-wmi.dll** using the following commands:

```
regsvr32 /u fio-smis-wmi.dll
```

```
regsvr32 fio-smis-wmi.dll
```

5. Start the winmgmt service via the services toolor the following command line:

```
net start winmgmt
```

About SMI-S - Windows

SMI-S is a collection of specifications that traditionally focus on Storage Area Network (SAN) systems based on the SCSI command set, such as Fibre Channel, iSCSI, and SAS. However, the general pattern used to model these storage systems can be applied to solid-state, direct-attached storage systems such as those provided by .

High IOPS devices are modeled using the SMI-S patterns established in the Storage HBA, Direct Attached (DA) Ports, and Host Discovered Resources Profiles. The physical aspects of the High IOPS device and all firmware and High IOPS VSL software are modeled using published DMTF specifications, including the Physical Asset, Software Inventory, PCI Device Profiles, and Common Diagnostic Model Profile.

See See SMI-S CIM Model on page 19. This chart describes the SMI-S CIM model, with High IOPS devices and their associated firmware and software. For simplicity, the prefix FIO_ has been removed from the class names.

A: iOMemoryPort Class

The central instance of the model is of the **IOMemoryPort** class (A in the figure), a logical representation of the High IOPS device. It supports the extrinsic methods necessary to provision the drive. An instance of **PCIDevice** (B) and **IOMemoryPort** exist for each installed High IOPS device, and they are associated with instances of **ConcreteIdentity** (1). An instance of **SSDStatistics** (C), which contains important performance and capacity data for the device, is associated by an **ElementStatisticalData** association (2) to each **IOMemoryPort. IOMemoryPort** is scoped by an instance of the **ComputerSystem** class. The **SystemDevice** (3) aggregation aggregates **IOMemoryPort** within the containing **ComputerSystem.**

E: iOMemoryPortController Class

An instance of **IOMemoryPortController** (E) represents the High IOPS VSL software used to control the installed High IOPS devices. **IOMemoryPortController** specializes **CIM_PortController,** and it aggregates **IoMemoryPort** with the **ControlledBy** (4) aggregation. The software version and vendor information are represented by the **SoftwareIdentity** (F) instance that is associated to **IOMemoryPortController** (E) via **ElementSoftwareIdentity** (5). The **SoftwareIdentity** that represents the installed ioMemory VSL software is associated to the scoping **ComputerSystem** using the **InstalledSoftwareIdentity** association (6).

An instance of the **ProtocolEndpoint** class (G) represents both ends of the logical data path between the **IOMemoryPort** and the solid-state storage. This aspect of the model is derived from the pattern in the

DA Ports Profile, where the port is both an initiator and target. **ProtocolEndpoint** is associated to the **IOMemoryPort** by **DeviceSAPImplementation** (7) and to the **ComputerSystem** by **HostedAccessPoint** (8).

H: LogicalSSD Class (Block Device)

The block device exposed to applications (file systems, database, and logical volume manager) is modeled using an instance of **LogicalSSD** (H), a subclass of **CIM_DiskDrive**. It is associated with a **StorageExtent** (J) using the **MediaPresent** association (9), but the **StorageExtent** will always be present. It is also associated to the **ProtocolEndpoint** (G) representing the **IOMemoryPort** using **SAPAvailableForElement** (10) and to the scoping **ComputerSystem** using **SystemDevice** (3).

High IOPSdevices, being PCIe devices, are also represented by an instance of the **PCIDevice class** (B). **IOMemoryPort** is an alternate representation of the **PCIDevice** and its associated control device. It is associated to it by the **ConcreteIdentity** association.

K: SoftwareIdentity

The High IOPS VSL software is also represented with **SoftwareIdentity**, which is associated to the **PCIDevice** by the **ElementSoftwareIdentity** association (11). The **SoftwareIdentity** (firmware) is associated to the scoping **ComputerSystem** by the **InstalledSoftwareIdentity** association (12). An instance of **SoftwareInstallationService** (L) is associated with each **PCIDevice**, which can be used to update device firmware.

M: PhysicalPackage

The physical aspects of High IOPS devices are represented by an instance of the **PhysicalPackage** class (M), which is associated to the **PCIDevice** by **Realizes** (13) and to the scoping **ComputerSystem** by **SystemPackaging** (14). The temperature sensors on High IOPS devices are represented by an instance of **TemperatureSensor** (N) and is associated to the **PhysicalPackage** by **AssociatedSensor.**

Implementation - Windows

This section describes the arrangement of instances and associations for the device CIM model. Not all class properties are described in detail. Consult the CIM schema for detailed description of all properties.

The device health is indicated by the value of the **HealthLevel** property. Values include: *Healthy, Warning, Reduced Write*, and *Read Only*. These values are mapped to **standardHealthState** values – *OK, Degraded/Warning*, and *Critical Failure* – as appropriate.

Extrinsic methods for device provisioning include attach, detach, format, and update. The attach method creates a block device for the High IOPS device. Detach disables the block device. A format option enables users to specify the device size in either megabytes or a percentage. The update method allows users to upgrade the firmware on the device.

Device longevity is indicated by the value of the **HealthPercentage** property. **FlashbackAvailability** indicates whether or not this feature of the High IOPS device is online. **IOMemoryPorts** are aggregated by **IOMemoryPortController** via the **ControlledBy** aggregation. Instances of **IOMemoryPort** are associated to their corresponding **PCIDevice** with the **ConcreteIdentity** association. The **IOMemoryPort** is a logical device of the scoping **ComputerSystem** and is indicated as such by the **SystemDevice** aggregation.

Products with two or more High IOPS devices, such as the Legacy IBM High IOPS Adapter do appear like two separate High IOPS devices. For products with multiple devices, the **IOMemoryPort** class is extended to include information about the carrier card type, serial number, and external power connection for the product as a whole.

IOMemoryPort

One instance of **IOMemoryPort** exists for each High IOPS device installed in the **ComputerSystem.**

The **LocationIndicator** property reflects the state of the device indicator beacon (e.g., all LEDs on solid). Reading the value gives the current state of the indicator. Writing the value with "On" or "Off" turns the indicator on or off and can be used to determine the device's physical location.

SSDStatistics

One instance of **SSDStatistics** exists for each **IOMemoryPort** instance. Properties of this object provide performance and capacity information. Some of this information is only available when the drive is attached (i.e., the state of the associated **IOMemoryPort** is "Attached").

IOMemoryPortController

Only one instance of **IOMemoryPortContoller** exists, representing the High IOPS VSL software used to control **IOMemoryPorts**. The **IOMemoryPortController** specializes the **CIM_ PortController.**

IOMemoryPortController is aggregated to the scoping **ComputerSystem** using the **SystemDevice** aggregation. **IOMemoryPortController** is associated with a **SoftwareInventory** instance representing the High IOPS VSL software properties via the **ElementSoftwareIdentity** association.

ProtocolEndpoint

One instance of **ProtocolEndpoint** exists for each instance of **IOMemoryPort**. It is associated to the **IOMemoryPort** using **DeviceSAPImplementation** and to **LogicalSSD** using **SAPAvailableForElement**. Because an **IOMemoryPort** represents both the initiator and target ports, only one **ProtocolEndpoint** per **IOMemoryPort** is needed to model the connection between **IOMemoryPort** and LogicalSSD.

LogicalSSD

One instance of **LogicalSSD**, a subclass of **CIM_DiskDrive**, exists for each block device (**/dev/fioX**) exposed by anHigh IOPS device. **Correlatable IDs** are used, based on operating system device names. This enables client applications to associate block devices discovered through this model with resources discovered from other SMI-S models instrumented on the host system.

ComputerSystem aggregates **LogicalSSDs** via **SystemDevice.** The **LogicalSSD** instances are associated to their **ProtocolEndpoints** via **SAPAvailableForElement.** If the **IOMemoryPort** associated to the endpoint is not attached, then the **Availability** property is set to "Off Line," and the **DeviceID** property value is "Unknown."

StorageExtent

One instance of **StorageExtent** is associated with each **LogicalSSD** and represents the logical storage of the associated device.

SoftwareIdentity

One instance of **SoftwareIdentity** exists to represent the High IOPS VSL software. The firmware is also modeled using **SoftwareIdentity** but requires an instance for each High IOPS device installed. **The IsEntity** property has a value of *True*, indicating that the **SoftwareIdentity** instance corresponds to a discrete copy of the High IOPS VSL software or firmware. The **MajorVersion, MinorVersion, RevisionNumber,** and **BuildNumber** properties convey the driver/firmware version information. The **Manufacturer** property can be used to identify IBM.

Another option for the firmware is to omit the **InstalledSoftwareIdentity** association with **ComputerSystem**, because the firmware is not really installed on **ComputerSystem**. This option would depend on how users want to model the firmware.

SoftwareInstallationService

An instance of **SoftwareInstallationService** exists for each **PCIDevice** and can be used to update the associated device's firmware.

PCIDevice

An instance of **PCIDevice** is instantiated for each High IOPS device (PCIe card) in the computer. Properties are set as follows:

- **BusNumber –** bus number where the PCIe device exists
- **DeviceNumber –** device number assigned to the PCI device for this bus.
- **FunctionNumber –** set to the function number for the PCI device.
- **SubsystemID, SubsystemVendorID, PCIDeviceID, VendorID,** and **RevisionID** are optional but can be populated if values can be extracted from the configuration registers of the PCI device.

PCIDevice is associated with **IOMemoryPort**, its alternate logical representation, using **ConcreteIdentity**. The **PCIDevice** is also associated with **PhysicalPackage,** representing the physical aspects of the High IOPS device, via **Realizes.**

PhysicalPackage

One instance of **PhysicalPackage** exists for each discrete, physical High IOPS device installed in the computer system. The Manufacturer, Model, SKU, SerialNumber, Version, and PartNumber properties can be used to describe these aspects of the physical card. **PhysicalPackage** is associated with **PCIDevice** via **Realizes** and the scoping **ComputerSystem** via **SystemPackaging.**

TemperatureSensor

One instance of **TemperatureSensor** exists for each **PhysicalPackage.** Temperature information for the drive is stored in the properties of this object.

Diagnostic Test

One instance of **DiagnosticTest** will exist. The **RunDiagnostic()** method will trigger a snapshot of device status for the specified **ManagedElement** which must be an instance of **IoMemoryPort**. The diagnostic run is synchronous and runs instantaneously. The resulting **ConcreteJob** object will associate to the originating **DiagnosticTest** instance and the respective **IoMemoryPort** instance that was specified. For more information, see See SMI-S CIM Model on page 19. At this time, **RunDiagnostic()** can only be used with the default **DiagnosticSettingData** provided.

Each run will add a single entry of **DiagnosticSettingDataRecord** and associated **DiagnosticCompletionRecord** in the **DiagnosticLog**. The **RecordData** property of the **DiagnosticCompletionRecord** will record critical device status at the time of the run. The format of the **RecordData** string can be found in the **RecordFormat** property.

The format is a series of status strings, each of which can hold one of the following values delimited by an asterisk (*) character: "Unknown", "OK", "Warning", or "Error". Currently, seven status values are recorded: **WearoutStatus, WritabilityStatus, FlashbackStatus, TemperatureStatus, MinimalModeStatus, PciStatus** and **InternalErrorStatus**. All of these should report "OK" under normal operating conditions.

WearoutStatus will be set to "Warning" when less than 10% reserve space is left on the device. It will be set to "Error" when there is no more reserved space. The messages will be:

- **WritabilityStatus** will be set to "Error" whenever the device is write throttling or in read-only mode. This can happen due to a variety of conditions including device wearout and insufficient power.
- **FlashbackStatus** will report "Warning" if a catastrophic error causes Flashback protection to be degraded.
- **TemperatureStatus** will report "Warning" when the device temperature is nearing the maximum safe temperature and "Error" when the maximum safe temperature is reached or surpassed.
- **MinimalModeStatus** will report either "Warning" or "Error" whenever the device is in minimal mode.
- **PciStatus** will report "Warning" or "Error" if there are compatibility problems with the host PCIe bus.
- **InternalErrorStatus** will report "Error" if there are any internal problems with the High IOPS VSL software.

The **CompletionState** property will summarize the results and may be set to Unknown, OK, Warning or Failed. If any status is in error the state will report as Failed. Otherwise, if there is any warning status the state will report Warning. The Message property will be set to indicate the appropriate action if there are any warnings or errors.

DiagnosticSetting Data

There will be an instance of **DiagnosticSettingData** associated with the **DiagnosticTest** instance. For more information, see See SMI-S CIM Model on page 19. It records the default settings for each call to **RunDiagnostic.**

DiagnosticServiceCapabilities

There is an instance of **DiagnosticServiceCapabilities** associated with the **DiagnosticTest** instance that records the capabilities of the **DiagnosticTest** service.

DiagnosticLog

An instance of **DiagnosticLog** is associated with the **DiagnosticTest** instance and will store the results of each run.

DiagnosticSettingRecord

A copy of the default **DiagnosticSettingData** will be stored in a **DiagnosticSettingDataRecord** each time a diagnostic is run and will be associated with an instance of **DiagnosticCompletionRecord**.

DiagnosticCompletionRecord

An instance of **DiagnosticCompletionRecord** will store the results of each **RunDiagnostic** execution. The details are explained in **DiagnosticTest.**

RegisteredDiskDriveLiteProfile

Only one instance of this class is needed. It resides in the **/root/interop** namespace and indicates the implementation of the Disk Drive Lite Profile. The following properties are set as follows:

- **InstanceID –** set to *SNIA:DiskDriveLiteProfile-1.4.0*
- **RegisteredOrganization –** set to "11" (SNIA)
- **RegisteredName –** set to *DirectAccess Ports Profile*
- **RegisteredVersion –** set to "1.4.0"

RegisteredDAPortsProfile

Only one instance of this class is needed. It resides in the **/root/interop** namespace and indicates the implementation of the DA Ports Profile. The properties are set as follows:

- **InstanceID** – set to *SNIA:DAPortsProfile-1.4.0*
- **RegisteredOrganization –** set to "11" (SNIA)
- **RegisteredName –** set to *DirectAccess Ports Profile*
- **RegisteredVersion –** set to "1.4.0"

RegisteredStorageHBAProfile

Only one instance of this class is needed. It resides in the **/root/interop** namespace and indicates the implementation of the Storage HBA Profile. The properties are set as follows:

- **InstanceID –** set to *SNIA:StorageHBAProfile-1.4.0*
- **RegisteredOrganization –** set to "11" (SNIA)
- **RegisteredName –** set to *Storage HBA Profile*
- **RegisteredVersion –** set to "1.4.0"

RegisteredHostDiscoveredResourcesProfile

Only one instance of this class is needed. It resides in the **/root/interop** namespace and indicates the implementation of the Host Discovered Resources Profile. The properties are set as follows:

- **InstanceID –** set to *SNIA:HostDiscoveredResourcesProfile-1.2.0*
- **RegisteredOrganization –** set to "11" (SNIA)
- **RegisteredName –** set to *Host Discovered Resources Profile*
- **RegisteredVersion –** set to "1.2.0"

RegisteredPCIDeviceProfile

Only one instance of this class is needed. It resides in the **/root/interop** namespace and indicates the implementation of the PCI Device Profile. The properties are set as follows:

- **InstanceID –** set to *DMTF:DSP1075-PCIDevice-1.0.0a*
- **RegisteredOrganization –** set to "2" (DMTF)
- **RegisteredName –** set to *PCIDevice Profile*
- **RegisteredVersion –** set to "1.0.0a"

RegisteredSoftwareInventoryProfile

Only one instance of this class is needed. It resides in the **/root/interop** namespace and indicates the implementation of the Software Inventory Profile. The properties are set as follows:

- **InstanceID –** set to *DMTF:DSP1023-SoftwareInventory-1.0.1*
- **RegisteredOrganization** – set to "2" (DMTF)
- **RegisteredName –** set to *Software Inventory Profile*
- **RegisteredVersion –** set to "1.0.1"

RegisteredSoftwareUpdateProfile

Only one instance of this class is needed. It resides in the **/root/interop** namespace and indicates the implementation of the Software Update Profile. The properties are set as follows:

- **InstanceID –** set to *DMTF:DSP1023-SoftwareUpdate-1.0.0*
- **RegisteredOrganization –** set to "2" (DMTF)
- **RegisteredName –** set to *Software Update Profile*
- **RegisteredVersion –** set to "1.0.0"

RegisteredPhysicalAssetProfile

Only one instance of this class is needed. It resides in the **/root/interop** namespace and indicates the implementation of the Physical Asset Profile. The properties are set as follows:

- **InstanceID –** set to *DMTF:PhysicalAssetProfile-1.0.2*
- **RegisteredOrganization –** set to "2" (DMTF)
- **RegisteredName –** set to *PhysicalAsset Profile*
- **RegisteredVersion –** set to "1.0.2"

RegisteredSensorsProfile

Only one instance of this class is needed. It resides in the **/root/interop** namespace and indicates the implementation of the Sensors Profile. The properties are set as follows:

- **InstanceID –** set to *SNIA:SensorsProfile-1.0.0*
- **RegisteredOrganization –** set to "11" (SNIA)
- **RegisteredName –** set to *Sensors Profile*
- **RegisteredVersion –** set to "1.0.0"

RegisteredCommonDiagnosticProfile

Only one instance of this class is needed. It will reside in the **/root/interop** namespace and indicate the implementation of the Common Diagnostic Model Profile. The **InstanceID** property will be set to a value of "DMTF:DiagnosticsProfile-2.0.0a". The **RegisteredOrganization** property will be set to a value of "2" (DMTF). The **RegisteredName** property will be set to a value of "Diagnostics Profile". The **RegisteredVersion** property will be set to a value of "2.0.0a".

Indications - Windows

An indication will be generated periodically when a serious condition exists for a particular High IOPS device. The WBEM provider currently supports six types of indications. They alert users of the SMI-S provider to conditions such as imminent wearout, degradation of writability, degradation of the flashback feature, higher temperature, and internal error states.

The indications will be instances of the **FIO_AlertIndiecation** class which simply specializes the **CIM_AlertIndication** class.

The values for the properties of the **FIO_AlertIndication** instances are under development and may change as testing proceeds and feedback is received.

FIO_AlertIndication

Property	Value
IndicationIdentifier	See below for each type
IndicationTime	Timestamp when sent
AlertingManagedElement	IoMemoryPort.DeviceID=<device ID>
AlertingElementFormat	CIMObjectPath (2)
AlertType	Device Alert (5)
PerceivedSeverity	See below for each type
ProbableCause	See below for each type
SystemCreationClassName	"FIO_AlertIndication"
SystemName	<hostname>
ProviderName	"fiosmis"
CorrelatedIndications	Not used
Description	Class description
OtherAlertType	Not used
OtherSeverity	Not used
ProbableCauseDescription	Not used
EventID	Same as IndicationIdentifier

Property	Value
OwningEntity	<vendor>
MessageID	Not used
Message	Not used
MessageArguments	Not used

Reduced Writability Indication

The High IOPS VSL software can dramatically reduce write throughput to manage device conditions such as excessive wear, high temperature, and insufficient power. The reduced writability indication is generated while the drive is in this mode. If the triggering condition is excessive wear, the **`IoMemoryPort`** health percentage will report 0% health.

Property	Value
IndicationIdentifier	<mfr>":"<hostname>":write"
PerceivedSeverity	Degraded/Warning (3)
ProbableCause	Threshold Crossed (52) Temperature Unacceptable (51) Power Problem (36)

Read-only Indication

When the drive has reached the end-of-life, it can no longer be written to and can only be read from. The read-only indication will be sent when this occurs. The **`IoMemoryPort`** health percentage will continue to report 0% health when this happens.

Property	Value
IndicationIdentifier	<mfr>":"<hostname>":read_only"
PerceivedSeverity	Degraded/Warning (3)
ProbableCause	Threshold Crossed (52)

Wearout Indication

As the drive wears out, this indication is generated as a warning when the drive health percentage drops below 10%, before write throughput is reduced.

Property	Value
IndicationIdentifier	<mfr>":"<hostname>":wearout"
PerceivedSeverity	Degraded/Warning (3)
ProbableCause	Threshold Crossed (52)

Flashback Indication

If a catastrophic part failure degrades the effectiveness of the flashback feature, this indication will be sent.

Property	Value
IndicationIdentifier	<mfr>":"<hostname>":flashback"
PerceivedSeverity	Degraded/Warning (3)
ProbableCause	Loss of Redundancy (88)

High Temperature Indication

This indication will be sent when the temperature of the card becomes excessive.

Property	Value
IndicationIdentifier	<mfr>":"<hostname>":temperature"
PerceivedSeverity	Critical (6)
ProbableCause	Temperature Unacceptable (51)

Error Indication

If the High IOPS VSL software is in an error state the error indication will be sent.

Property	Value
IndicationIdentifier	<mfr>":"<hostname>":error"
PerceivedSeverity	Major (6)
ProbableCause	Other (1)

SMI-S Interface (VMware)

To manage the High IOPS VSL, you must use the provided management utilities. There are two options available for managing the VSL:

- **COS/Shell/TSM command-line utilities:** These utilities are installed with the ioMemory VSL software. In order to use these utilities on ESXi, the Shell/TSM (Tech Support Mode) must be enabled.
  - The Compile Host Log troubleshooting utility is only available as a COS/Shell/TSM command-line utility.
  - For more information about these utilities, see See Remote Command-line Utilities on page 7.
- **Remote SMI-S Scripts:** These provide remote management of the software and devices without enabling Tech Support Mode (TSM) or logging in to the COS.
  - To use the SMI-S interface, you must install the CIM (SMI-S) provider on the ESX(i) host and the Python SMI-S Management Scripts on a remote machine.
  - This section explains how to install the CIM provider.

IBM's SMI-S interface allows you to remotely manage the High IOPS VSL software on your ESX(i) system. The SMI-S provider works with popular CIM servers, including SFCB. SFCB is part of a typical ESX(i) installation, and it is used by vSphere software to manage the ESX(i) system.

Installing the SMI-S Provider on ESXi 5.0

Be sure to transfer the CIM (SMI-S) provider offline bundle to the host (hypervisor) machine's local storage.

1. Stop all VMs and put the host in Maintenance Mode.
2. Install the CIM provider while in Maintenance Mode by running the following command:

```
esxcli --server <servername> software vib install -d <offline-bundle> --
no-sig-check
```

Where <offline-bundle> is the absolute path to the offline bundle on the hypervisor host. For example, if the offline bundle is in the bundles directory of a datastore with the name of **`datastore1`**, the path would be: **`vmfs/volumes/datastore1/bundles/<offline-bundle>`**

NOTE-

Command-line Installation:
You can install the CIM provider on the ESXi 5.0 host using the SSH/TSM. Simply use the same esxcli command without the **`--server`** option.

3. Reboot your ESXi system.

This installs the SMI-S provider and registers it with the SFCB server. You are now able to connect to the SMI-S provider.

Installing the SMI-S Provider on ESX(i) 4.x Using the vCLI

1. Stop all VMs and put the host in Maintenance Mode.
2. Navigate to the folder (on the remote machine) with the downloaded files.
3. Install the SMI-S provider.

```
vihostupdate --server <server-name> --install --bundle --no-sig-check
./fusionio-cimprovider-<esx-version>-bundle-<version>.zip
```

4. Reboot the ESXi host.

This installs the SMI-S provider and registers it with the SFCB server. You are now able to connect to the SMI-S provider.

Installing the SMI-S Provider on ESX(i) 4.x using the Command-line Interface

To install the ioMemory VSL on an existing ESX(i) host using esxupdate:

1. Turn on the host and log in as administrator.
2. Stop all VMs and enter maintenance mode.
3. Navigate to the directory where you have transferred offline bundle.
4. Run the esxupdate command to install drivers using the offline bundle.

```
$ esxupdate --bundle
fusionio-cimprovider-<esx-version>-bundle-<version>.zip update --no-sig-
check
```

5. Reboot the host system.

This installs the SMI-S provider and registers it with the SFCB server. You are now able to connect to the SMI-S provider.

Interfacing with the SMI-S Provider

There are two standard methods for managing your High IOPS devices through the SMI-S provider. These are:

- Python Management Scripts (recommended): IBM provides Python scripts that can be implemented remotely on a Linux machine with the proper Python packages installed. For more information, see See Remote Command-line Utilities on page 7.
- CIM Browsers: If you are familiar with Common Information Models, and are comfortable using CIM browsers (such as YAWN), then you can connect to the SMI-S provider using your preferred browser. For more information, including the IBM SMI-S CIM model, see See Using a CIM Browser for SMI-S Management on page 53.

Using a CIM Browser for SMI-S Management

This section outlines our SMI-S CIM model, including the instances and associations within that model. You can use this model along with a CIM browser to interface with the SMI-S provider installed on an ESX (i) host. This will allow you to manage your ioMemory device(s).

Before you can use a CIM browser to interface with the SMI-S provider, you must first install the SMI-S provider on your ESX(i) host system.

Attention!

We recommend using our Python WBEM scripts to interface with the SMI-S provider instead of a CIM browser, especially if you are unfamiliar with CIM browsers. This section is meant for users who are versed in WBEM, SMI-S and DMTF standards. For more information on using the remote command-line utilities , see See Remote Command-line Utilities on page 7.

SMI-S Interface Background

The SMI-S interface is based on Web-Based Enterprise Management (WBEM) and provides a Common Information Model (CIM) model that represents the ioMemory device and associated software, in accordance with existing Distributed Management Task Force (DMTF), Storage Networking Industry Association (SNIA), and Storage Management Initiative Specification (SMI-S) standards. This model permits backward-compatible extension, accommodating new hardware and software features developed by .

References

CIM Schema v2.26
http://www.dmtf.org/standards/cim/cim_schema_v2260

DMTF DSP1011, Physical Asset Profile
http://www.dmtf.org/standards/published_documents/DSP1011_1.0.2.pdf

DMTF DSP1023, Software Inventory Profile
http://www.dmtf.org/standards/published_documents/DSP1023_1.0.1.pdf

DMTF DSP1033, Profile Registration
http://www.dmtf.org/standards/published_documents/DSP1033_1.0.0.pdf

DMTF DSP1075 PCI Device Profile
http://www.dmtf.org/standards/published_documents/DSP1075_1.0.0.pdf

DMTF DSP1002, Diagnostics Profile
http://www.dmtf.org/standards/published_documents/DSP1002_2.0.0.pdf

SMI-S v1.4 Architecture
http://www.snia.org/sites/default/files/SMI-Sv1.4r6_Architecture.book_.pdf

SMI-S v1.4 Common Profiles
http://www.snia.org/sites/default/files/SMI-Sv1.4r6_CommonProfiles.book_.pdf

SMI-S v1.4 Host Profiles
http://www.snia.org/sites/default/files/SMI-Sv1.4r6_Host.book_.pdf

SMI-S v1.4 Common Diagnostic Model
http://www.dmtf.org/standards/mgmt/cdm/

Description

SMI-S is a collection of specifications that traditionally focus on Storage Area Network (SAN) systems based on the SCSI command set, such as Fibre Channel, iSCSI, and SAS. However, the general pattern used to model these storage systems can be applied to solid-state, direct-attached storage systems such as those provided by . High IOPS devices are modeled using the SMI-S patterns established in the Storage HBA, Direct Attached (DA) Ports, and Host Discovered Resources Profiles. The physical aspects of the High IOPSdevice and all firmware and High IOPS VSL software are modeled using published DMTF specifications, including the Physical Asset, Software Inventory, PCI Device Profiles, and Common Diagnostic Model Profile.

See See SMI-S CIM Model on page 19 for the chart that describes the SMI-S CIM model, with High IOPS devices and their associated firmware and software. For simplicity, the prefix FIO_ has been removed from the class names.

A: IOMemoryPort Class

The central instance of the model is of the **`IOMemoryPort`** class (A in the figure), a logical representation of the High IOPS device. It supports the extrinsic methods necessary to provision the drive. An instance of **`PCIDevice`** (B) and **`IOMemoryPort`** exist for each installed High IOPS device, and they are associated with instances of **`ConcreteIdentity`** (1). An instance of SSDStatistics (C), that contains important performance and capacity data for the device, is associated by an **`ElementStatisticalData`** association (2) to each **`IOMemoryPort`**. **`IOMemoryPort`** is scoped by an instance of the **`ComputerSystem`** class. The **`SystemDevice`** (3) aggregation aggregates **`IOMemoryPort`** within the containing **`ComputerSystem.`**

E: IOMemoryPortController Class

An instance of **`IOMemoryPortController`** (E) represents the High IOPS VSL software used to control the installed High IOPS devices. **`IOMemoryPortController`** specializes **`CIM_PortController`**, and it aggregates **`IoMemoryPort`** with the **`ControlledBy`** (4) aggregation. The software version and vendor information are represented by the **`SoftwareIdentity`** (F) instance that is associated to **`IOMemoryPortController`** (E) via **`ElementSoftwareIdentity`** (5). The **`SoftwareIdentity`** that represents the installed High IOPS VSL software is associated to the scoping **`ComputerSystem`** using the **`InstalledSoftwareIdentity`** association (6).

An instance of the **`ProtocolEndpoint`** class (G) represents both ends of the logical data path between the **`IOMemoryPort`** and the solid-state storage. This aspect of the model is derived from the pattern in the DA Ports Profile, where the port is both an initiator and target. **`ProtocolEndpoint`** is associated to the **`IOMemoryPort`** by **`DeviceSAPImplementation`** (7) and to the **`ComputerSystem`** by **`HostedAccessPoint`** (8).

H: LogicalSSD Class (Block Device)

The block device exposed to applications (file systems, database, and logical volume manager) is modeled using an instance of **LogicalSSD** (H), a subclass of **CIM_DiskDrive**. It is associated with a **StorageExtent** (J) using the **MediaPresent** association (9), but the **StorageExtent** will always be present. It is also associated to the **ProtocolEndpoint** (G) representing the **IOMemoryPort** using **SAPAvailableForElement** (10) and to the scoping **ComputerSystem** using **SystemDevice** (3).

High IOPS devices, being PCIe devices, are also represented by an instance of the **PCIDevice** class (B). **IOMemoryPort** is an alternate representation of the **PCIDevice** and its associated control device. It is associated to it by the **ConcreteIdentity** association.

K: SoftwareIdentity

The High IOPS VSL software is also represented with **SoftwareIdentity**, which is associated to the **PCIDevice** by the **ElementSoftwareIdentity** association (11). The **SoftwareIdentity** (firmware) is associated to the scoping **ComputerSystem** by the **InstalledSoftwareIdentity** association (12). An instance of **SoftwareInstallationService** (L) is associated with each **PCIDevice**, which can be used to update device firmware.

M: Physical Aspects

The physical aspects of High IOPS devices are represented by an instance of the **PhysicalPackage** class (M), which is associated to the **PCIDevice** by **Realizes** (13) and to the scoping **ComputerSystem** by **SystemPackaging** (14). The temperature sensors on High IOPS devices are represented by an instance of **TemperatureSensor** (N) and is associated to the **PhysicalPackage** by **AssociatedSensor.**

Implementation

This section describes the arrangement of instances and associations for the device CIM model. Not all class properties are described in detail. Consult the CIM schema for detailed description of all properties.

The device health is indicated by the value of the **HealthLevel** property. Values include: *Healthy, Warning, Reduced Write,* and *Read Only*. These values are mapped to **standardHealthState** values – *OK, Degraded/Warning*, and *Critical Failure* – as appropriate.

Extrinsic methods for device provisioning include attach, detach, format, and update. The attach method creates a block device for the High IOPS device. Detach disables the block device. A format option enables users to specify the device size in either megabytes or a percentage. The update method allows users to upgrade the firmware on the device.

Device longevity is indicated by the value of the **HealthPercentage** property. **FlashbackAvailability** indicates whether or not this feature of the High IOPS device is online.

IOMemoryPorts are aggregated by **IOMemoryPortController** via the **ControlledBy** aggregation. Instances of **IOMemoryPort** are associated to their corresponding **PCIDevice** with the **ConcreteIdentity** association. The **IOMemoryPort** is a logical device of the scoping **ComputerSystem** and is indicated as such by the **SystemDevice** aggregation.

Products with two or more High IOPS devices, such as the IBM High IOPS Duo Adapterdo appear like two separate High IOPS devices. For products with multiple devices, the **IOMemoryPort** class is extended to include information about the carrier card type, serial number, and external power connection for the product as a whole.

IOMemoryPort

One instance of **IOMemoryPort** exists for each High IOPS device installed in the **ComputerSystem**.

The **LocationIndicator** property reflects the state of the device indicator beacon (e.g., all LEDs on solid). Reading the value gives the current state of the indicator. Writing the value with "On" or "Off" turns the indicator on or off and can be used to determine the device's physical location.

SSDStatistics

One instance of **SSDStatistics** exists for each **IOMemoryPort** instance. Properties of this object provide performance and capacity information. Some of this information is only available when the drive is attached (i.e., the state of the associated **IOMemoryPort** is "Attached").

IOMemoryPortController

Only one instance of **IOMemoryPortContoller** exists, representing the High IOPS VSL software used to control **IOMemoryPorts**. The **IOMemoryPortController** specializes the **CIM_ PortController**.

IOMemoryPortController is aggregated to the scoping **ComputerSystem** using the **SystemDevice** aggregation. **IOMemoryPortController** is associated with a **SoftwareInventory** instance representing the High IOPS VSL software properties via the **ElementSoftwareIdentity** association.

ProtocolEndpoint

One instance of **ProtocolEndpoint** exists for each instance of **IOMemoryPort**. It is associated to the **IOMemoryPort** using **DeviceSAPImplementation** and to **LogicalSSD** using **SAPAvailableForElement**. Because an **IOMemoryPort** represents both the initiator and target ports, only one **ProtocolEndpoint** per **IOMemoryPort** is needed to model the connection between **IOMemoryPort** and **LogicalSSD**.

LogicalSSD

One instance of **LogicalSSD**, a subclass of **CIM_DiskDrive,** exists for each block device (**/dev/fioX**) exposed by an High IOPS device. Correlatable IDs are used, based on operating system device names. This enables client applications to associate block devices discovered through this model with resources discovered from other SMI-S models instrumented on the host system.

ComputerSystem aggregates **LogicalSSDs** via **SystemDevice**. The **LogicalSSD** instances are associated to their **ProtocolEndpoints** via **SAPAvailableForElement**. If the **IOMemoryPort** associated to the endpoint is not attached, then the **Availability** property is set to "Off Line," and the **DeviceID** property value is "Unknown."

StorageExtent

One instance of **StorageExtent** is associated with each **LogicalSSD** and represents the logical storage of the associated device.

SoftwareIdentity

One instance of **SoftwareIdentity** exists to represent the High IOPS VSL software. The firmware is also modeled using **SoftwareIdentity** but requires an instance for each High IOPS device installed. The **IsEntity** property has a value of True, indicating that the **SoftwareIdentity** instance corresponds to a discrete copy of the High IOPS VSL software or firmware. The **MajorVersion, MinorVersion, RevisionNumber,** and **BuildNumber** properties convey the driver/firmware version information. The **Manufacturer** property can be used to identify IBM.

Another option for the firmware is to omit the **InstalledSoftwareIdentity** association with **ComputerSystem**, because the firmware is not really installed on **ComputerSystem**. This option would depend on how users want to model the firmware.

SoftwareInstallationService

An instance of **SoftwareInstallationService** exists for each **PCIDevice** and can be used to update the associated device's firmware.

PCIDevice

An instance of **PCIDevice** is instantiated for each High IOPS device (PCIe card) in the computer. Properties are set as follows:

- **BusNumber –** bus number where the PCIe device exists
- **DeviceNumber –** device number assigned to the PCI device for this bus.
- **FunctionNumber –** set to the function number for the PCI device.
- **SubsystemID, SubsystemVendorID, PCIDeviceID, VendorID,** and **RevisionID** are optional but can be populated if values can be extracted from the configuration registers of the PCI device.

PCIDevice is associated with **IOMemoryPort**, its alternate logical representation, using **ConcreteIdentity**. The **PCIDevice** is also associated with **PhysicalPackage**, representing the physical aspects of the High IOPS device, via **Realizes.**

PhysicalPackage

One instance of **PhysicalPackage** exists for each discrete, physical High IOPS device installed in the computer system. The Manufacturer, Model, SKU, SerialNumber, Version, and PartNumber properties can be used to describe these aspects of the physical card. **PhysicalPackage** is associated with **PCIDevice** via **Realizes** and the scoping **ComputerSystem** via **SystemPackaging.**

TemperatureSensor

One instance of **TemperatureSensor** exists for each **PhysicalPackage.** Temperature information for the drive is stored in the properties of this object.

Diagnostic Test

One instance of **DiagnosticTest** will exist. The **RunDiagnostic()** method will trigger a snapshot of device status for the specified **ManagedElement** which must be an instance of **IoMemoryPort**. The diagnostic run is synchronous and runs instantaneously. The resulting **ConcreteJob** object will associate to the originating **DiagnosticTest** instance and the respective **IoMemoryPort** instance that was specified (see See SMI-S CIM Model on page 19). At this time, **RunDiagnostic()** can only be used with the default **DiagnosticSettingData** provided.

Each run will add a single entry of **DiagnosticSettingDataRecord** and associated **DiagnosticCompletionRecord** in the **DiagnosticLog**. The **RecordData** property of the **DiagnosticCompletionRecord** will record critical device status at the time of the run. The format of the **RecordData** string can be found in the **RecordFormat** property.

The format is a series of status strings, each of which can hold one of the following values delimited by an asterisk (*) character: "Unknown", "OK", "Warning", or "Error". Currently, seven status values are recorded: **WearoutStatus, WritabilityStatus, FlashbackStatus, TemperatureStatus, MinimalModeStatus, PciStatus** and **InternalErrorStatus**. All of these should report "OK" under normal operating conditions.

WearoutStatus will be set to "Warning" when less than 10% reserve space is left on the device. It will be set to "Error" when there is no more reserved space. The messages are:

- **WritabilityStatus** will be set to "Error" whenever the device is write throttling or in read-only mode. This can happen due to a variety of conditions including device wearout and insufficient power.
- **FlashbackStatus** will report "Warning" if a catastrophic error causes Flashback protection to be degraded.
- **TemperatureStatus** will report "Warning" when the device temperature is nearing the maximum safe temperature and "Error" when the maximum safe temperature is reached or surpassed.
- **MinimalModeStatus** will report either "Warning" or "Error" whenever the device is in minimal mode.
- **PciStatus** will report "Warning" or "Error" if there are compatibility problems with the host PCIe bus.
- **InternalErrorStatus** will report "Error" if there are any internal problems with the ioMemory VSLsoftware.

The **CompletionState** property will summarize the results and may be set to Unknown, OK, Warning or Failed. If any status is in error the state will report as Failed. Otherwise, if there is any warning status the state will report Warning. The **Message** property will be set to indicate the appropriate action if there are any warnings or errors.

DiagnosticSetting Data

There will be an instance of **DiagnosticSettingData** associated with the **DiagnosticTest** instance (see See SMI-S CIM Model on page 19). It records the default settings for each call to **RunDiagnostic**.

DiagnosticServiceCapabilities

There is an instance of **DiagnosticServiceCapabilities** associated with the **DiagnosticTest** instance that records the capabilities of the **DiagnosticTest** service.

DiagnosticLog

An instance of **DiagnosticLog** is associated with the **DiagnosticTest** instance and will store the results of each run.

DiagnosticSettingRecord

A copy of the default **DiagnosticSettingData** will be stored in a **DiagnosticSettingDataRecord** each time a diagnostic is run and will be associated with an instance of **DiagnosticCompletionRecord**.

DiagnosticCompletionRecord

An instance of **DiagnosticCompletionRecord** will store the results of each **RunDiagnostic** execution. The details are explained in DiagnosticTest.

RegisteredDiskDriveLiteProfile

Only one instance of this class is needed. It resides in the **/root/interop** namespace and indicates the implementation of the Disk Drive Lite Profile. The following properties are set as follows:

- **InstanceID –** set to *SNIA:DiskDriveLiteProfile-1.4.0*
- **RegisteredOrganization** – set to "11" (SNIA)
- **RegisteredName –** set to *DirectAccess Ports Profile*
- **RegisteredVersion –** set to "1.4.0"

RegisteredDAPortsProfile

Only one instance of this class is needed. It resides in the **/root/interop** namespace and indicates the implementation of the DA Ports Profile. The properties are set as follows:

- **InstanceID** – set to *SNIA:DAPortsProfile-1.4.0*
- **RegisteredOrganization** – set to "11" (SNIA)
- **RegisteredName** – set to *DirectAccess Ports Profile*
- **RegisteredVersion** – set to "1.4.0"

RegisteredStorageHBAProfile

Only one instance of this class is needed. It resides in the **/root/interop** namespace and indicates the implementation of the Storage HBA Profile. The properties are set as follows:

- **InstanceID** – set to *SNIA:StorageHBAProfile-1.4.0*
- **RegisteredOrganization** – set to "11" (SNIA)
- **RegisteredName** – set to *Storage HBA Profile*
- **RegisteredVersion** – set to "1.4.0"

RegisteredHostDiscoveredResourcesProfile

Only one instance of this class is needed. It resides in the **/root/interop** namespace and indicates the implementation of the Host Discovered Resources Profile. The properties are set as follows:

- **InstanceID** – set to *SNIA:HostDiscoveredResourcesProfile-1.2.0*
- **RegisteredOrganization** – set to "11" (SNIA)
- **RegisteredName** – set to *Host Discovered Resources Profile*
- **RegisteredVersion** – set to "1.2.0"

RegisteredPCIDeviceProfile

Only one instance of this class is needed. It resides in the **/root/interop** namespace and indicates the implementation of the PCI Device Profile. The properties are set as follows:

- **InstanceID** – set to *DMTF:DSP1075-PCIDevice-1.0.0a*
- **RegisteredOrganization** – set to "2" (DMTF)
- **RegisteredName** – set to *PCIDevice Profile*
- **RegisteredVersion** – set to "1.0.0a"

RegisteredSoftwareInventoryProfile

Only one instance of this class is needed. It resides in the **/root/interop** namespace and indicates the implementation of the Software Inventory Profile. The properties are set as follows:

- **InstanceID** – set to *DMTF:DSP1023-SoftwareInventory-1.0.1*
- **RegisteredOrganization** – set to "2" (DMTF)
- **RegisteredName** – set to *Software Inventory Profile*
- **RegisteredVersion** – set to "1.0.1"

RegisteredSoftwareUpdateProfile

Only one instance of this class is needed. It resides in the **/root/interop** namespace and indicates the implementation of the Software Update Profile. The properties are set as follows:

- **InstanceID** – set to *DMTF:DSP1023-SoftwareUpdate-1.0.0*
- **RegisteredOrganization** – set to "2" (DMTF)
- **RegisteredName –** set to *Software Update Profile*
- **RegisteredVersion** – set to "1.0.0"

RegisteredPhysicalAssetProfile

Only one instance of this class is needed. It resides in the **/root/interop** namespace and indicates the implementation of the Physical Asset Profile. The properties are set as follows:

- **InstanceID** – set to *DMTF:PhysicalAssetProfile-1.0.2*
- **RegisteredOrganization** – set to "2" (DMTF)
- **RegisteredName** – set to *PhysicalAsset Profile*
- **RegisteredVersion** – set to "1.0.2"

RegisteredSensorsProfile

Only one instance of this class is needed. It resides in the **/root/interop** namespace and indicates the implementation of the Sensors Profile. The properties are set as follows:

- **InstanceID** – set to *SNIA:SensorsProfile-1.0.0*
- **RegisteredOrganization** – set to "11" (SNIA)
- **RegisteredName** – set to *Sensors Profile*
- **RegisteredVersion** – set to "1.0.0"

RegisteredCommonDiagnosticProfile

Only one instance of this class is needed. It will reside in the **/root/interop** namespace and indicate the implementation of the Common Diagnostic Model Profile. The **InstanceID** property will be set to a value of "DMTF:DiagnosticsProfile-2.0.0a". The RegisteredOrganization property will be set to a value of "2"

(DMTF). The **RegisteredName** property will be set to a value of "Diagnostics Profile". The **RegisteredVersion** property will be set to a value of "2.0.0a".

Indications

An indication will be generated periodically when a serious condition exists for a particular High IOPS device. The WBEM provider currently supports six types of indications. They alert users of the SMI-S provider to conditions such as imminent wearout, degradation of writability, degradation of the flashback feature, higher temperature, and internal error states.

The indications will be instances of the **FIO_AlertIndiecation** class which simply specializes the **CIM_AlertIndication** class.

The values for the properties of the **FIO_AlertIndication** instances are under development and may change as testing proceeds and feedback is received.

FIO_AlertIndication

Property	Value
IndicationIdentifier	See below for each type
IndicationTime	Timestamp when sent
AlertingManagedElement	IoMemoryPort.DeviceID=<device ID>
AlertingElementFormat	CIMObjectPath (2)
AlertType	Device Alert (5)
PerceivedSeverity	See below for each type
ProbableCause	See below for each type
SystemCreationClassName	"FIO_AlertIndication"
SystemName	<hostname>
ProviderName	"fiosmis"
CorrelatedIndications	Not used
Description	Class description
OtherAlertType	Not used
OtherSeverity	Not used
ProbableCauseDescription	Not used
EventID	Same as IndicationIdentifier

Property	Value
OwningEntity	<vendor>
MessageID	Not used
Message	Not used
MessageArguments	Not used

Reduced Writability Indication

The High IOPS VSL software can dramatically reduce write throughput to manage device conditions such as excessive wear, high temperature, and insufficient power. The reduced writability indication is generated while the drive is in this mode. If the triggering condition is excessive wear, the **`IoMemoryPort`** health percentage will report 0% health.

Property	Value
IndicationIdentifier	<mfr>":"<hostname>":write"
PerceivedSeverity	Degraded/Warning (3)
ProbableCause	Threshold Crossed (52) Temperature Unacceptable (51) Power Problem (36)

Read-only Indication

When the drive has reached the end-of-life, it can no longer be written to and can only be read from. The read-only indication will be sent when this occurs. The IoMemoryPort health percentage will continue to report 0% health when this happens.

Property	Value
IndicationIdentifier	<mfr>":"<hostname>":read_only"
PerceivedSeverity	Degraded/Warning (3)
ProbableCause	Threshold Crossed (52)

Wearout Indication

As the drive wears out, this indication is generated as a warning when the drive health percentage drops below 10%, before write throughput is reduced.

Property	Value
IndicationIdentifier	<mfr>":"<hostname>":wearout"
PerceivedSeverity	Degraded/Warning (3)
ProbableCause	Threshold Crossed (52)

Flashback Indication

If a catastrophic part failure degrades the effectiveness of the flashback feature, this indication will be sent.

Property	Value
IndicationIdentifier	<mfr>":"<hostname>":flashback"
PerceivedSeverity	Degraded/Warning (3)
ProbableCause	Loss of Redundancy (88)

High Temperature Indication

This indication will be sent when the temperature of the card becomes excessive.

Property	Value
IndicationIdentifier	<mfr>":"<hostname>":temperature"
PerceivedSeverity	Critical (6)
ProbableCause	Temperature Unacceptable (51)

Error Indication

If the High IOPS VSL software is in an error state the error indication will be sent.

Property	Value
IndicationIdentifier	<mfr>":"<hostname>":error"
PerceivedSeverity	Major (6)
ProbableCause	Other (1)

Setting Up SNMP (Linux)

The **fio-snmp-agentx** SNMP agent is an RFC 2741-compliant AgentX sub-agent. It can work with any RFC-compliant SNMP agent, such as Net-SNMP. The master SNMP agent defers queries to **fio-snmp-agentx** for supported MIBs.

SNMP Master Agent - Linux

The **fio-snmp-agentx**, provided in the **fio-util** package, requires an already-installed SNMP master agent. The SNMP master agent must support and be configured for AgentX connections (see http://www.ietf.org/rfc/rfc2741.txt). The **fio-snmp-agentx** is tested and verified with Net-SNMP, which is the typical SNMP agent provided with most Linux distributions.

There are several agents available that support this functionality. If you choose to use Net-SNMP, then use the instructions in the following sections to configure and launch it.

Launching the SNMP Master Agent

Install the Net-SNMP package using the package manager for your version of Linux.

Red Hat

Use the following command to install Net-SNMP on Red Hat:

```
yum install net-snmp rsync
```

Other Linux Versions

Use the standard system package manager to install the Net-SNMP package on your Linux distribution. The **fio-snmp-mib** package places MIB files in **/usr/share/fio/mib.**

Configuring the Master Agent

You can configure the Net-SNMP master agent daemon to set the network communications parameters, security, and other options by using the **snmpd.conf** text file. The location of this file is system-dependent; often it is in **/etc/snmp or /usr/share/snmp.**

A simple **snmpd** configuration file might include the following:

```
# set standard SNMP variables
syslocation "Data room, third rack"
syscontact itguy@example.com
# required to enable the AgentX protocol
master agentx
agentxsocket tcp:localhost:705
#set the port that the agent listens on (defaults to 161)
agentaddress 161
# simple access control (some form of access control is required)
rocommunity public
```

Running the Master Agent

Once you install and configure the master agent, you must start or restart the **snmpd** daemon for the new parameters to take effect. You can simply run **snmpd** from its installed location (often **/usr/sbin** – see the **snmpd** man page for options). It typically needs root privileges to run properly. You can also use the snmpd startup script in **/etc/init.d or /etc/rc.d/init.d**. If you are concerned about security, use the more advanced SNMPv3 access control instead of the rocommunity and rwcommunity access control directives as outlined in the relevant man page.

SNMP AgentX Subagent - Linux

Attention!

The SNMP agent requires the **libvsl** RPM package. This should have been installed as part of the High IOPS VSL software installation.

Installing the SNMP Subagent

1. Download the IBM SNMP packages.
2. Install the package using your operating systems package manager. For instance, on Red Hat, run the following:

```
rpm -Uvh fio-snmp-*.rpm
```

The SNMP package places its MIB files in **/usr/share/fio/mib.**

Running and Configuring the Fusion SNMP Subagent

1. Configure the subagent by creating a **fio-snmp-agentx.conf** file.
2. Store this **.conf** file in the **/opt/fio/etc/snmp** directory.
3. At a minimum, set the agent network parameters in this file similar to the following:

```
# required to enable the AgentX protocol

agentxsocket tcp:localhost:705
```

This must match the AgentX network parameters in the **snmpd.conf** file for the master agent. For further AgentX configuration information, consult the man pages or visit http://www.net-snmp.org.

The **fio-snmp-agentx** startup script will launch automatically at boot time once the installation and configuration is complete.

Manually Running the SNMP Subagent

If you need to run the SNMP Subagent manually, follow these steps:

1. After the SNMP master agent is started, start the subagent by running this command:

```
/usr/bin/fio-snmp-agentx
```

   This command launches the subagent using the Net-SNMP configuration file named **fio-snmp-agentx.conf**. This file must reside in one of the **/opt/fio/etc/snmp** directory.

2. You can now view the ioMemory device management information using an SNMP MIB browser or by using a network management system accessing **FIOioDrv.mib** (in **/usr/share/fio/mib**).

The subagent can be run with the following parameters:

```
fio-snmp-agentx [options]
```

Option	Description
-f	Force the sub-agent to run in the foreground instead of as a daemon
-l	<log file> Log file to use
-s	Send errors to stderr instead of to syslog

Subagent Log File

The SNMP subagent can maintain a log file regarding its own activities. This file is separate from the MIB, as it includes entries on the subagent's communications with the master agent, including any errors or intermittent issues.

To have the subagent maintain this log file, include the **-l** parameter and a path to the log file as part of the command in running the subagent. For example, this command:

```
fio-snmp-agentx -l /usr/snmp/subagent.log
```

keeps the subagent log file as **subagent.log**, in the **/usr/snmp directory.**

The SNMP subagent is now ready to monitor your device.

Using the SNMP Sample Config Files - Linux

When you install SNMP, the following sample config files are available:

- **/usr/share/doc/fio-snmp-agentx/conf/snmpd.conf/** (master agent)
- **/usr/share/doc/fio-snmp-agentx/conf/fio-snmp-agentx.conf/** (sub-agent)

To customize and use the sample config files,

1. Rename your **snmpd.conf** file (such as to **snmpd-orig.conf**) and your **fio-snmp-agentx.conf** file (such as to **fio-snmp-agentx-orig.conf**). The **snmpd.conf** file usually reside in **/etc/snmp or /usr/share/snmp.** The **fio-snmp-agentx.conf** file resides in the **/opt/fio/etc/snmp** directory.
2. From the **/usr/share/doc/fio-snmp-agentx/conf/** directory, copy the sample **snmpd.conf** file and the sample **fio-snmp-agentx.conf** file to the appropriate directories.
3. Edit the sample files you copied and save your changes as **snmpd.conf** and **fio-snmp-agentx.conf.**

Enabling SNMP Test Mode - Linux

When the SNMP Agentx runs, it reads the **fio-snmp-agentx** config file:

```
#####################################################################
# Example config file for fio-snmp-agentx SNMP AgentX subagent.
#
# Fusion-io, Inc. #

agentxsocket tcp:localhost:705

# test_mode_enabled
# set to 1, true or yes to enable 0, false or no to disable (default:
false)
test_mode_enabled true
# traps_enabled
traps_enabled true
# testmode_file
# name of test mode file (default: testmode.ini)
testmode_file testmode.ini

# update_delay
# delay between agent polling requests in milliseconds (default: 250)
update_delay 100

# mib_select
# set to fio for FUSIONIO-IODRV-MIB or cpq for CPQIODRV-MIB (default: fio)
mib_select fio
#####################################################################
```

Conditions for test mode are described below:

1. If the Admin has set the **test_mode_enabled** parameter from TRUE to FALSE, the SNMP does not try to run test mode. Instead, it continues processing data as usual from the ioMemory VSL software, storing the data in the MIB.
2. If the CONF file says that **test_mode_enabled** is TRUE, the SNMP subagent reads the testmode.ini is read periodically by the subagent to check for any changes. A sample testmode.ini file is installed in **/usr/share/doc/fio-snmp-agentx/conf.**
3. If the **testmode.ini** file shows the test mode is set to ON, then it engages the test mode.
4. To find the SNMP index values, execute an SNMP WALK query against the OID: fusionIoDimmInfoIndex.

The SNMP subagent replaces any existing High IOPS VSL software data it may have (for the High IOPS device specified by **`TestModeIndex`**) with any populated fields in the list of parameters. If a field is not populated, Agentx retains the existing data and reports it to the MIB. If there is a value in a field, then the Agentx replaces that data and reports it to the MIB.

The subagent continues in test mode until the .INI file parameter is set to OFF. The test mode information is described in the **`testmode.ini`** file:

```
# SNMP Test Mode sample file.
# These values may be used to test the SNMP subsystem when it is in test
mode.

[SNMP Agent Test Mode]
TestMode = off
TestModeIndex = 0

# InfoState: Note that the following states may change, but current
definitions are:
# 0 = unknown
# 1 = detached
# 2 = attached
# 3 = minimal mode
# 4 = error
# 5 = detaching
# 6 = attaching
# 7 = scanning
# 8 = formatting
# 9 = updating firmware
# 10 = attach
# 11 = detach
# 12 = format
# 13 = update

InfoState = 2

InfoInternalTemp = 45
InfoAmbientTemp = 35
InfoWearoutIndicator = 2 ; 2=normal, 1=device is wearing out.
```

```
InfoWritableIndicator = 2 ; 2=normal, 1=non-writable, 0=write-reduced,
3=unknown
InfoFlashbackIndicator = 2 ; 2=normal, 1=flashback protection degraded.

ExtnTotalPhysCapacityU = 23
ExtnTotalPhysCapacityL = 215752192
ExtnUsablePhysCapacityU = 21
ExtnUsablePhysCapacityL = 7852192
ExtnUsedPhysCapacityU = 4
ExtnUsedPhysCapacityL = 782330816
ExtnTotalLogCapacityU = 18
ExtnTotalLogCapacityL = 2690588672
```

```
ExtnAvailLogCapacityU = 14
ExtnAvailLogCapacityL = 3870457856

ExtnBytesReadU = 18
ExtnBytesReadL = 3690588672
ExtnBytesWrittenU = 4
ExtnBytesWrittenL = 2578550816

InfoHealthPercentage = 95

InfoMinimalModeReason = 7 ; 0=unknown, 1=fw out of date, 2=low power, ;
3=dual plane failure, 5=internal, 6=card limit, ; 7=not in minimal mode,
8=unsupported OS, ; 9=low memory

InfoReducedWriteReason = 0 ; 0=none, 1=user requested, 2=no md blocks, ;
3=no memory, 4=failed die, 5=wearout, ; 6=adapter power, 7=internal,
8=power limit

InfoMilliVolts = 12000
InfoMilliVoltsPeak = 12100
InfoMilliVoltsMin = 11900
InfoMilliWatts = 6000
InfoMilliWattsPeak = 15000
InfoMilliAmps = 500
InfoMilliAmpsPeak = 1000

InfoAdapterExtPowerPresent = 1 ; 1=present, 2=absent

InfoPowerlossProtectDisabled = 2 ; 1=powerloss protection available but
disabled; 2=any other powerloss protection condition
```

SNMP MIB Support - Linux

For information on each MIB, including a description and variables, you may use a MIB browsing tool load one or more MIB files. The following SNMP MIB fields are supported in Linux:

- fusionIoDimmMibRevMajor
- fusionIoDimmInfoAdapterType
- fusionIoDimmMibRevMinor
- fusionIoDimmInfoAdapterPort
- fusionIoDimmMIBCondition
- fusionIoDimmInfoAdapterSerialNumber
- fusionIoDimmInfoIndex
- fusionIoDimmInfoAdapterExtPowerPresent
- fusionIoDimmInfoStatus
- fusionIoDimmInfoPowerlossProtectDisabled
- fusionIoDimmInfoName
- fusionIoDimmInfoInternalTempHigh
- fusionIoDimmInfoSerialNumber
- fusionIoDimmInfoAmbientTemp
- fusionIoDimmInfoPartNumber

- fusionIoDimmInfoPCIBandwidthCompatibility
- fusionIoDimmInfoSubVendorPartNumber
- fusionIoDimmInfoPCIPowerCompatibility
- fusionIoDimmInfoSparePartNumber
- fusionIoDimmInfoActualGoverningLevel
- fusionIoDimmInfoAssemblyNumber
- fusionIoDimmInfoLifespanGoverningLevel
- fusionIoDimmInfoFirmwareVersion
- fusionIoDimmInfoPowerGoverningLevel
- fusionIoDimmInfoDriverVersion
- fusionIoDimmInfoThermalGoverningLevel
- fusionIoDimmInfoUID
- fusionIoDimmInfoLifespanGoverningEnabled
- fusionIoDimmInfoState
- fusionIoDimmInfoLifespanGoverningTgtDate
- fusionIoDimmInfoClientDeviceName
- fusionIoDimmExtnIndex
- fusionIoDimmInfoBeacon
- fusionIoDimmExtnTotalPhysCapacityU
- fusionIoDimmInfoPCIAddress
- fusionIoDimmExtnTotalPhysCapacityL
- fusionIoDimmInfoPCIDeviceID
- fusionIoDimmExtnTotalLogCapacityU
- fusionIoDimmInfoPCISubdeviceID
- fusionIoDimmExtnTotalLogCapacityL
- fusionIoDimmInfoPCIVendorID
- fusionIoDimmExtnBytesReadU
- fusionIoDimmInfoPCISubvendorID
- fusionIoDimmExtnBytesReadL
- fusionIoDimmInfoPCISlot
- fusionIoDimmExtnBytesWrittenU
- fusionIoDimmInfoWearoutIndicator
- fusionIoDimmExtnBytesWrittenL
- fusionIoDimmInfoWritableIndicator
- fusionIoDimmExtnFormattedBlockSize
- fusionIoDimmInfoInternalTemp
- fusionIoDimmExtnCurrentRAMUsageU
- fusionIoDimmInfoHealthPercentage
- fusionIoDimmExtnCurrentRAMUsageL
- fusionIoDimmInfoMinimalModeReason
- fusionIoDimmExtnPeakRAMUsageU
- fusionIoDimmInfoReducedWriteReason
- fusionIoDimmExtnPeakRAMUsageL
- fusionIoDimmInfoMilliVolts
- fusionIoDimmWearoutTrap
- fusionIoDimmInfoMilliVoltsPeak
- fusionIoDimmNonWritableTrap
- fusionIoDimmInfoMilliVoltsMin
- fusionIoDimmTempHighTrap
- fusionIoDimmInfoMilliWatts
- fusionIoDimmTempOkTrap
- fusionIoDimmInfoMilliWattsPeak
- fusionIoDimmErrorTrap
- fusionIoDimmInfoMilliAmps

- fusionIoDimmPowerlossProtectTrap
- fusionIoDimmInfoMilliAmpsPeak

Sample SNMP Monitoring - Linux

As a data-source management tool, the SNMP agentx provides information that you can integrate into existing management applications. This example shows how an organization adapted their current management application to use certain SNMP traps and SNMP MIBs to monitor all of the ioMemory devices installed in their network.

These conditions follow the same conditions scheme as described in the Example Conditions to Monitor section earlier in this guide.

Condition	Trap or MIB	GREEN	YELLOW	RED
Device Status	fusionIoDimmInfoState: 2.1.1.1.12	attached(2)	detached(1), detaching(5), attaching(6), scanning(7), formatting(8), or updating(9)	minimal(3) or error(4)
Minimal Mode Reason	fusionIoDimmInfoMinimalModeReason: 2.1.1.1.30	7	N/A	Any other value
Power Loss Protection	fusionIoDimmInfoPowerlossProtectDisabled: 2.1.1.1.43	2	N/A	Any other value
Temperature	fusionIoDimmInfoCurrentTemp: 2.1.1.1.24	<90	90-96	97
Health Reserves	fusionIoDimmInfoPercentLifeRemaining: 2.1.1.1.25	>10	4-10	<4
Wearout Indicator	fusionIoDimmInfoWearoutIndicator: 2.1.1.1.21	2	N/A	Any other value
Writeable Indicator	fusionIoDimmInfoNonWritableIndicator: 2.1.1.1.23	2	N/A	Any other value
PCI Power Compatibility	fusionIoDimmInfoPCIPowerCompatibility: 2.1.1.1.46	2048	16	0

Setting up SNMP (Windows)

The software for High IOPS VSL 3.2.2 and later does not provide the options to install support for SNMP. Once you run the Windows Setup program, it will stop and start the Windows SNMP Service to recognize the VSL's agent.

If you did not choose to install the SNMP support at Setup, and want to do so later, rerun the Setup program. Choose to install only the SNMP support from the list of items. Once the Setup program completes the install, it will stop and restart the Windows SNMP Service.

For details on using SNMP Test Mode, see See SNMP Test Mode and MIB Support on page 72.

SNMP Test Mode and MIB Support

This section explains how you can set up a test mode with your Windows SNMP agent. This enables you to set test values in a Windows registry and force SNMP traps without having to create the actual conditions on the device.

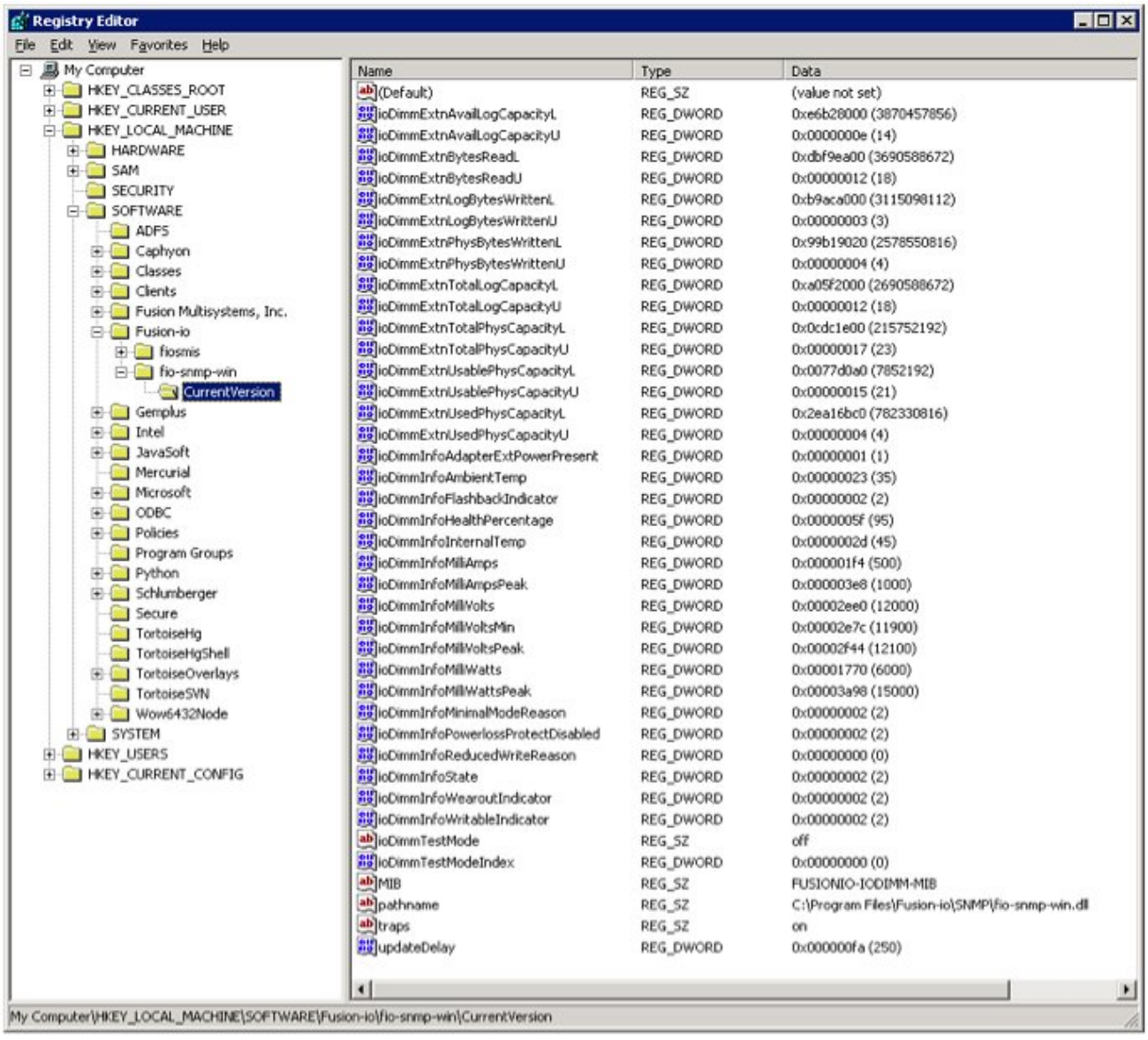
For example, you can use the SNMP test mode to change the non-writeable indicator and generate a trap, or simulate a change to the physical or logical size of the device, etc.

NOTE-

To use SNMP Test Mode, you must have installed the SNMP option with your High IOPS VSL software.

Using Test-Mode Registry Values

The screen capture below shows the registry entries included for SNMP test values.



Each of these entries is described below. Entries marked by an asterisk (*) generate SNMP traps when set to the indicated values, and the **`fusionIoDimmMIBCondition`** and **`fusionIoDimmInfoStatus`** MIB variables may be affected because of the changes.

NOTE-

All entries, except those marked by **, reflect your registry changes immediately. Entries marked by ** require a restart of the Windows SNMP agent for the changes to take effect.

SNMP Test Registry Entry	Description
ioDimmExtnAvailLogCapacityL	Lower word of the available logical capacity in bytes
ioDimmExtnAvailLogCapacityU	Upper word of the available logical capacity in bytes
ioDimmExtnBytesReadL	Lower word of the total number of bytes read since the device was formatted
ioDimmExtnBytesReadU	Upper word of the total number of bytes read since the device was formatted
ioDimmExtnLogBytesWrittenL	Lower word of the number of user data bytes written
ioDimmExtnLogBytesWrittenU	Upper word of the number of user data bytes written
ioDimmExtnPhysBytesWrittenL	Lower word of the total physical bytes written
ioDimmExtnPhysBytesWrittenU	Upper word of the total physical bytes written
ioDimmExtnTotalLogCapacityL	Lower word of the total logical capacity in bytes as formatted
ioDimmExtnTotalLogCapacityU	Upper word of the total logical capacity in bytes as formatted
ioDimmExtnTotalPhysCapacityL	Lower word of the total logical capacity in bytes as formatted
ioDimmExtnTotalPhysCapacityU	Upper word of the total logical capacity in bytes as formatted
ioDimmExtnUsablePhysCapacityL	Lower word of the useable physical capacity in bytes. This is space that is holding valid data, or is erased and ready for writing, or is waiting to be reclaimed via garbage collection.
ioDimmExtnUsablePhysCapacityU	Upper word of the useable physical capacity in bytes. This is space that is holding valid data, or is erased and ready for writing, or is waiting to be reclaimed via garbage collection.
*ioDimmInfoInternalTemp	Current internal temperature of the device in degrees Celsius. If this value is set above 78 degrees Celsius for ioDimm cards, a trap is generated. If set above 90 degrees for HP Mezzanine cards, a trap is generated.
*ioDimmInfoFlashbackIndicator	1 = flashback redundancy is degraded; 2 = false

SNMP Test Registry Entry	Description
*ioDimmInfoNonWritableIndicator	1 = device is no longer writable because it has surpassed the read-only threshold; 2 = false
ioDimmInfoPercentLifeRemaining	Upper word of the total logical capacity in bytes as formatted
*ioDimmInfoState (trap generated if state = 4)	Current state of the attached client device: unknown(0) detached(1) attached(2), minimal(3), error(4), detaching(5), attaching(6), scanning(7), formatting(8), updating(9), attach(10), detach(11), format(12), update(13)
*ioDimmInfoWearoutIndicator	Boolean: True = device has surpassed the wearout threshold
ioDimmTestMode	Set test mode on or off
ioDimmTestModeIndex	Number indicating the selected
pathname	Path to the driver, set at installation
**traps	Set trap generation on or off
**updateDelay	Number of milliseconds to wait until getting the next value from the ioMemory VSL software to generate a trap

SNMP MIB Support

The following SNMP MIB fields are supported in Windows:

- fusionIoDimmMibRevMajor
- fusionIoDimmInfoAdapterType
- fusionIoDimmMibRevMinor
- fusionIoDimmInfoAdapterPort
- fusionIoDimmMIBCondition
- fusionIoDimmInfoAdapterSerialNumber
- fusionIoDimmInfoIndex
- fusionIoDimmInfoAdapterExtPowerPresent
- fusionIoDimmInfoStatus
- fusionIoDimmInfoPowerlossProtectDisabled

- fusionIoDimmInfoName
- fusionIoDimmInfoInternalTempHigh
- fusionIoDimmInfoSerialNumber
- fusionIoDimmInfoAmbientTemp
- fusionIoDimmInfoPartNumber
- fusionIoDimmInfoPCIBandwidthCompatibility
- fusionIoDimmInfoSubVendorPartNumber
- fusionIoDimmInfoPCIPowerCompatibility
- fusionIoDimmInfoSparePartNumber
- fusionIoDimmInfoActualGoverningLevel
- fusionIoDimmInfoAssemblyNumber
- fusionIoDimmInfoLifespanGoverningLevel
- fusionIoDimmInfoFirmwareVersion
- fusionIoDimmInfoPowerGoverningLevel
- fusionIoDimmInfoDriverVersion
- fusionIoDimmInfoThermalGoverningLevel
- fusionIoDimmInfoUID
- fusionIoDimmInfoLifespanGoverningEnabled
- fusionIoDimmInfoState
- fusionIoDimmInfoLifespanGoverningTgtDate
- fusionIoDimmInfoClientDeviceName
- fusionIoDimmExtnIndex
- fusionIoDimmInfoBeacon
- fusionIoDimmExtnTotalPhysCapacityU
- fusionIoDimmInfoPCIAddress
- fusionIoDimmExtnTotalPhysCapacityL
- fusionIoDimmInfoPCIDeviceID
- fusionIoDimmExtnTotalLogCapacityU
- fusionIoDimmInfoPCISubdeviceID
- fusionIoDimmExtnTotalLogCapacityL
- fusionIoDimmInfoPCIVendorID
- fusionIoDimmExtnBytesReadU
- fusionIoDimmInfoPCISubvendorID
- fusionIoDimmExtnBytesReadL
- fusionIoDimmInfoPCISlot
- fusionIoDimmExtnBytesWrittenU
- fusionIoDimmInfoWearoutIndicator
- fusionIoDimmExtnBytesWrittenL
- fusionIoDimmInfoWritableIndicator
- fusionIoDimmExtnFormattedBlockSize
- fusionIoDimmInfoInternalTemp
- fusionIoDimmExtnCurrentRAMUsageU
- fusionIoDimmInfoHealthPercentage
- fusionIoDimmExtnCurrentRAMUsageL
- fusionIoDimmInfoMinimalModeReason
- fusionIoDimmExtnPeakRAMUsageU
- fusionIoDimmInfoReducedWriteReason
- fusionIoDimmExtnPeakRAMUsageL
- fusionIoDimmInfoMilliVolts
- fusionIoDimmWearoutTrap
- fusionIoDimmInfoMilliVoltsPeak
- fusionIoDimmNonWritableTrap
- fusionIoDimmInfoMilliVoltsMin
- fusionIoDimmTempHighTrap

- fusionIoDimmInfoMilliWatts
- fusionIoDimmTempOkTrap
- fusionIoDimmInfoMilliWattsPeak
- fusionIoDimmErrorTrap
- fusionIoDimmInfoMilliAmps
- fusionIoDimmPowerlossProtectTrap
- fusionIoDimmInfoMilliAmpsPeak

Sample SNMP Monitoring

As a data-source management tool, the SNMP agentx provides information that you can integrate into existing management applications. This example shows how an organization adapted their current management application to use certain SNMP traps and SNMP MIBs to monitor all of the ioMemory devices installed in their network.

Condition	Trap or MIB	GREEN	YELLOW	RED
Device Status	fusionIoDimmInfoState: 2.1.1.1.12	attached(2)	detached(1), detaching(5), attaching(6), scanning(7), formatting(8), or updating(9)	minimal(3) or error(4)
Minimal Mode Reason	fusionIoDimmInfoMinimalModeReason: 2.1.1.1.30	7	N/A	Any other value
Power Loss Protection	fusionIoDimmInfoPowerlossProtectDisabled: 2.1.1.1.43	2	N/A	Any other value
Temperature	fusionIoDimmInfoCurrentTemp: 2.1.1.1.24	<90	90-96	97
Health Reserves	fusionIoDimmInfoPercentLifeRemaining: 2.1.1.1.25	>10	4-10	<4
Wearout Indicator	fusionIoDimmInfoWearoutIndicator: 2.1.1.1.21	2	N/A	Any other value
Writeable Indicator	fusionIoDimmInfoNonWritableIndicator: 2.1.1.1.23	2	N/A	Any other value
PCI Power Compatibility	fusionIoDimmInfoPCIPowerCompatibility: 2.1.1.1.46	2048	16	0

Setting Up SNMP (Solaris)

The **fio-snmp-agentx** SNMP agent is an RFC 2741-compliant AgentX sub-agent. It can work with any RFC-compliant SNMP agent, such as Net-SNMP. The master SNMP agent defers queries to **fio-snmp-agentx** for supported MIBs.

SNMP AgentX Subagent - Solaris

Installing the SNMP Subagent

1. Properly install the High IOPS VSL (driver).
2. Download the SNMP packages: .
    - fio-snmp-agentx-<version>.pkg.tar.gz
    - libfio-dev-<version>.pkg.tar.gz
3. Install the **fio-snmp-agentx** package:
4. Navigate to the directory where you downloaded the files.
5. Uncompress the **fio-snmp-agentx** package.

```
$ gunzip -c fio-snmp-agentx-*.pkg.tar.gz |tar -xf -
```

6. Install the **fio-snmp-agentx** package.

```
$ yes|pfexec pkgadd -d . fio-snmp-agentx
```

7. Install the **libfio-dev** package.
8. Navigate to the directory where you downloaded the files.
9. Uncompress the **libfio-dev** package.

```
$ gunzip -c libfio-dev-*.pkg.tar.gz |tar -xf -
```

10. Install the **libfio-dev** package.

```
$ yes|pfexec pkgadd -d . libfio-dev
```

The MIB files are placed into **/usr/local/share/fio/mib.**

Running and Configuring the SNMP Subagent

NOTE-

You can use the sample .conf files that are installed in **/usr/local/share/doc/fio-snmp-agentx/conf/**, for more information see See Using the SNMP Sample Config Files - Solaris on page 82.

1. Configure the subagent by creating a **fio-snmp-agentx.conf** file.
2. Store this **.conf** file in the **/opt/fio/etc/snmp** directory.
3. At a minimum, set the agent network parameters in this file similar to the following:

```
# required to enable the AgentX protocol
agentxsocket unix:/tmp/fio-snmp-agentx-socket
```

This must match the AgentX network parameters in the **snmpd.conf** file for the master agent. For further AgentX configuration information, consult the man pages or visit http://www.net-snmp.org.

The **fio-snmp-agentx** startup script will launch automatically at boot time once the installation and configuration is complete.

SNMP Master Agent - Solaris

The **fio-snmp-agentx**, requires a SNMP master agent. The SNMP master agent must support and be configured for AgentX connections (see http://www.ietf.org/rfc/rfc2741.txt). The **fio-snmp-agentx** is tested and verified with Net-SNMP.

There are several agents available that support this functionality. If you choose to use Net-SNMP, then use the instructions in the following sections to configure and launch it.

Installing the SNMP Master Agent

Install the net-SNMP implementation of SNMP if it is not already installed on your system. Consult this document for more information: http://www.net-snmp.org/docs/README.solaris.html

1. Shutdown the Solaris **snmpdx** service

```
$ pfexec /etc/init.d/init.snmpdx stop
```

2. Shutdown the **net-snmp** that is running; it is typically misconfigured

```
$ pfexec /etc/init.d/init.sma stop
```

3. Get and build the **net-snmp** source bundle

```
$ wget
http://sourceforge.net/projects/net-snmp/files/net-snmp/5.6.1/net-snmp-
5.6.1.tar.gz/cd net-snmp-5.6.1
$ ./configure
$ make
$ make test
```

NOTE-

This fails **com2secunix** directive (**/net-snmp-5.6.1/testing/fulltests/default/T072com2secunix_simple**)

4. Install **net-snmp**.

```
$ pfexec make install
```

NOTE-

The log is **/var/log/snmpd.log**, persistent info is **/var/net-snmp**.

Attention!

Do not start **net-snmp** without a config file.

5. Get the default config file

```
$ pfexec snmpconf -g basic_setup
```

Answer all questions.

6. Copy the MIBs into place

```
$ pfexec cp /usr/local/share/fio/mib/*.mib /etc/sma/snmp/
```

7. Restart the net-snmp server

```
$ pfexec /etc/init.d/init.sma start
```

Configuring the Master Agent

NOTE-

You can use the **sample .conf** files that are installed in **/usr/local/share/doc/fio-snmp-agentx/conf/**, for more information see See Using the SNMP Sample Config Files - Solaris on page 82.

1. Back up any existing **snmpd.conf** files in **/etc/local/share/snmp**.
2. Rename the backup **snmpd.conf** files found on the system (such as to **snmpd.conf.bak**).

NOTE-

net-SNMP will look in several directories for an snmpd.conf file and will use the first one that it finds, and that might not be the one you want.

3. The net-SNMP implementation for Solaris does not support anything other than unix domain sockets for agentx transport. Add the following line to your snmpd.conf file to match the line in the subagent config file:

```
agentxsocket unix:/tmp/fio-snmp-agentx-socket
```

4. Copy the MIB file from **/usr/local/share/fio/mib/fioIoDimm.mib** to **/etc/sma/snmp/mibs**.

Running the Master Agent and Subagent

1. Start **snmpd** with this command:

```
/usr/local/sbin/snmpd
```

2. Make sure **net-snmpd** is running and has AgentX support enabled by checking the **net-snmpd** log file found in **/var/log/snmpd.log**. The log should contain an entry similar to this:

```
Turning on AgentX master support.
NET-SNMP version 5.6
```

3. Verify that the **snmpd** is working by using **snmpwalk**. It should return multiple MIB objects. Sample output:

```
snmpwalk -c public -v 1 -m ALL localhost
RFC1213-MIB::sysDescr.0 = STRING: "SunOS unknown 5.10 Generic_137138-09
i86pc"
RFC1213-MIB::sysObjectID.0 = OID: NET-SNMP-TC::solaris
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (133945) 0:22:19.45
RFC1213-MIB::sysContact.0 = STRING: "itguy@example.com"
RFC1213-MIB::sysName.0 = STRING: "unknown"
RFC1213-MIB::sysLocation.0 = STRING: "\"Data room, third rack\""
SNMPv2-MIB::sysORLastChange.0 = Timeticks: (1) 0:00:00.01
SNMPv2-MIB::sysORID.1 = OID: SNMP-MPD-MIB::snmpMPDMIBObjects.3.1.1
```

4. Start **fio-snmp-agentx** with this command (this also specifies the log location):

```
/opt/fusionio/bin/fio-snmp-agentx -l /var/log/fio-snmp-agentx.log
```

5. Verify that the subagent is running and connected by viewing the log (located in **/var/log/fio-snmp-agentx.log**. The log should contain an entry similar to this:

```
NET-SNMP version 5.6 AgentX subagent connected
fio-snmp-agentx
```

6. Use **snmpwalk** to also verify that **fio-snmp-agentx** is working. It should only return MIB objects for High IOPS devices. Sample output:

```
snmpwalk -c public -v 1 -m ALL localhost fusionio
FUSIONIO-IODIMM-MIB::fusionIoDimmExtnPeakRAMUsageU.13 = Counter32: 0
FUSIONIO-IODIMM-MIB::fusionIoDimmExtnPeakRAMUsageU.14 = Counter32: 0
FUSIONIO-IODIMM-MIB::fusionIoDimmExtnPeakRAMUsageU.19 = Counter32: 0
FUSIONIO-IODIMM-MIB::fusionIoDimmExtnPeakRAMUsageU.20 = Counter32: 0
FUSIONIO-IODIMM-MIB::fusionIoDimmExtnPeakRAMUsageL.13 = Counter32:
496850944
FUSIONIO-IODIMM-MIB::fusionIoDimmExtnPeakRAMUsageL.14 = Counter32:
```

```
108988416
FUSIONIO-IODIMM-MIB::fusionIoDimmExtnPeakRAMUsageL.19 = Counter32:
109250560
```

Using the SNMP Sample Config Files - Solaris

When you install the High IOPS VSL package, the following sample config files are available:

- /usr/local/share/doc/fio-snmp-agentx/conf/snmpd.conf (master agent)
- /usr/local/share/doc/fio-snmp-agentx/conf/fio-snmp-agentx.conf (sub-agent)

To customize and use the sample config files:

1. Rename your existing **snmpd.conf** file (such as to **snmpd-orig.conf**). The **snmpd.conf** file usually resides in **/etc/local/share/snmp**.

NOTE-

net-SNMP will look in several directories for an **snmpd.conf** file and will use the first one that it finds, so the original files need to be renamed to prevent their use.

Attention!

Instead of replacing the original **snmpd.conf** file with the sample **snmpd.conf** file, you can merge the two files together. Use the original file as the starting point, and make changes based on the sample file.

2. From the **/usr/local/share/docfio-snmp-agentx/conf/** directory, copy the sample **snmpd.conf** file and the sample **fio-snmp-agentx.conf** file to the appropriate directories.

- Copy the fio-snmp-agentx.conf file to /opt/fio/etc/snmp
- Copy the snmpd.conf file to /etc/local/share/snmp

3. Edit the sample files you copied and save your changes as **snmpd.conf** and **fio-snmp-agentx.conf**.
4. Add the following line to the **snmpd.conf** file to make it match the **fio-snmp-agentx.conf** file:

```
agentxsocket unix:/tmp/fio-snmp-agentx-socket
```

Return to one of the following sections to complete the configuration:

- See Running and Configuring the SNMP Subagent on page 78
- See Configuring the Master Agent on page 80

Enabling SNMP Test Mode - Solaris

When the SNMP Agentx runs, it reads the **fio-snmp-agentx** config file:

```
#####################################################################
# Example config file for fio-snmp-agentx SNMP AgentX subagent.
#
# Fusion-io, Inc. #

agentxsocket tcp:localhost:705

# test_mode_enabled
# set to 1, true or yes to enable 0, false or no to disable (default:
false)
test_mode_enabled true
# traps_enabled
traps_enabled true
# testmode_file
# name of test mode file (default: testmode.ini)
testmode_file testmode.ini
```

```
# update_delay
# delay between agent polling requests in milliseconds (default: 250)
update_delay 100

# mib_select
# set to fio for FUSIONIO-IODRV-MIB or cpq for CPQIODRV-MIB (default: fio)
mib_select fio
#####################################################################
```

Conditions for test mode are described below:

- If the Admin has set the **test_mode_enabled** parameter from TRUE to FALSE, the SNMP does not try to run test mode. Instead, it continues processing data as usual from the High IOPS VSL, storing the data in the MIB.
- If the CONF file says that **test_mode_enabled** is TRUE, the SNMP subagent reads the **testmode.ini** is read periodically by the subagent to check for any changes. A sample **testmode.ini** file is installed in **/usr/share/doc/fio-snmp-agentx/conf.**
- If the **testmode.ini** file shows the test mode is set to ON, then it engages the test mode.
- If test mode is ON, the SNMP Agentx reads the next line, **TestModeIndex**, to identify which High IOPS device to test. The number in this parameter is the PCIe device number shown using fio-status such as:

```
PCI:01:00.0
```

  The first two numerals identify the PCIe bus number (in this case, 01). This bus number is reported in hexadecimal, whereas the **TestModeIndex** in the **testmode.ini** file must be specified in decimal. The converted number should be entered into **testmode.ini**. The **TestModeIndex** must be a valid bus number of an High IOPS device installed in the system.

The SNMP subagent now replaces any existing High IOPS device High IOPS VSL data it may have (for the High IOPS device specified by **TestModeIndex**) with any populated fields in the list of parameters. If a

field is not populated, Agentx retains the existing data and reports it to the MIB. If there is a value in a field, then the Agentx replaces that data and reports it to the MIB.

The subagent continues in test mode until the .INI file parameter is set to OFF. The test mode information is described in the **`testmode.ini`** file :

```
# SNMP Test Mode sample file.
# These values may be used to test the SNMP subsystem when it is in test
mode.

[SNMP Agent Test Mode]
TestMode = off
TestModeIndex = 0

# InfoState: Note that the following states may change, but current
definitions are:
# 0 = unknown
# 1 = detached
# 2 = attached
# 3 = minimal mode
# 4 = error
# 5 = detaching
# 6 = attaching
# 7 = scanning
# 8 = formatting
# 9 = updating firmware
# 10 = attach
# 11 = detach
# 12 = format
# 13 = update

InfoState = 2

InfoInternalTemp = 45
InfoAmbientTemp = 35
InfoWearoutIndicator = 2 ; 2=normal, 1=device is wearing out.
```

```
InfoWritableIndicator = 2 ; 2=normal, 1=non-writable, 0=write-reduced,
3=unknown
InfoFlashbackIndicator = 2 ; 2=normal, 1=flashback protection degraded.

ExtnTotalPhysCapacityU = 23
ExtnTotalPhysCapacityL = 215752192
ExtnUsablePhysCapacityU = 21
ExtnUsablePhysCapacityL = 7852192
ExtnUsedPhysCapacityU = 4
ExtnUsedPhysCapacityL = 782330816
ExtnTotalLogCapacityU = 18
ExtnTotalLogCapacityL = 2690588672
ExtnAvailLogCapacityU = 14
ExtnAvailLogCapacityL = 3870457856
```

```
ExtnBytesReadU = 18
ExtnBytesReadL = 3690588672
ExtnBytesWrittenU = 4
ExtnBytesWrittenL = 2578550816

InfoHealthPercentage = 95
InfoMinimalModeReason = 7 ; 0=unknown, 1=fw out of date, 2=low power,;
3=dual plane failure, 5=internal,6=card limit,; 7=not in minimal
mode,8=unsupported OS,; 9=low memory

InfoReducedWriteReason = 0 ; 0=none, 1=user requested, 2=no md blocks,;
3=no memory, 4=failed die,5=wearout,; 6=adapter power, 7=internal,8=power
limit

InfoMilliVolts = 12000
InfoMilliVoltsPeak = 12100
InfoMilliVoltsMin = 11900
InfoMilliWatts = 6000
InfoMilliWattsPeak = 15000
InfoMilliAmps = 500
InfoMilliAmpsPeak = 1000

InfoAdapterExtPowerPresent = 1 ; 1=present, 2=absent

InfoPowerlossProtectDisabled = 2 ; 1=powerloss protection available but
disabled; 2=any other powerloss protection condition
```

SNMP MIB Support - Solaris

The following SNMP MIB fields are supported:

- fusionIoDimmMibRevMajor
- fusionIoDimmInfoReducedWriteReason
- fusionIoDimmMibRevMinor
- fusionIoDimmInfoMilliVolts
- fusionIoDimmMibCondition
- fusionIoDimmInfoMilliVoltsPeak
- fusionIoDimmInfoIndex
- fusionIoDimmInfoMilliVoltsMin
- fusionIoDimmInfoStatus
- fusionIoDimmInfoMilliWatts
- fusionIoDimmInfoName
- fusionIoDimmInfoMilliWattsPeak
- fusionIoDimmInfoSerialNumber
- fusionIoDimmInfoMilliAmps
- fusionIoDimmInfoPartNumber
- fusionIoDimmInfoMilliAmpsPeak
- fusionIoDimmInfoSubVendorPartNumber
- fusionIoDimmInfoAdapterType
- fusionIoDimmInfoSparePartNumber

- fusionIoDimmInfoAdapterPort
- fusionIoDimmInfoAssemblyNumber
- fusionIoDimmInfoAdapterSerialNumber
- fusionIoDimmInfoFirmwareVersion
- fusionIoDimmInfoAdapterExtPowerPresent
- fusionIoDimmInfoDriverVersion
- fusionIoDimmInfoPowerlossProtectDisabled
- fusionIoDimmInfoUID
- fusionIoDimmInfoInternalTempHigh
- fusionIoDimmInfoState
- fusionIoDimmInfoAmbientTemp
- fusionIoDimmInfoClientDeviceName
- fusionIoDimmExtnIndex
- fusionIoDimmInfoBeacon
- fusionIoDimmExtnTotalPhysCapacityU
- fusionIoDimmInfoPCIAddress
- fusionIoDimmExtnTotalPhysCapacityL
- fusionIoDimmInfoPCIBandwidthCompatibility
- fusionIoDimmExtnUsablePhysCapacityU
- fusionIoDimmInfoPCIDeviceID
- fusionIoDimmExtnUsablePhysCapacityL
- fusionIoDimmInfoPCIPowerCompatibility
- fusionIoDimmExtnUsedPhysCapacityU
- fusionIoDimmInfoPCISubdeviceID
- fusionIoDimmExtnUsedPhysCapacityL
- fusionIoDimmInfoPCIVendorID
- fusionIoDimmExtnTotalLogCapacityU
- fusionIoDimmInfoPCISubvendorID
- fusionIoDimmExtnTotalLogCapacityL
- fusionIoDimmInfoPCISlot
- fusionIoDimmExtnAvailLogCapacityU
- fusionIoDimmInfoWearoutIndicator
- fusionIoDimmExtnAvailLogCapacityL
- fusionIoDimmInfoFlashbackIndicator
- fusionIoDimmExtnBytesReadU
- fusionIoDimmInfoWritableIndicator
- fusionIoDimmExtnBytesReadL
- fusionIoDimmInfoInternalTemp
- fusionIoDimmExtnBytesWrittenU
- fusionIoDimmInfoHealthPercentage fusionIoDimmExtnBytesWrittenL
- fusionIoDimmInfoMinimalModeReason
- fusionIoDimmExtnFormattedBlockSize

Sample SNMP Monitoring - Solaris

As a data-source management tool, the SNMP agentx provides information that you can integrate into existing management applications. This example shows how an organization adapted their current management application to use certain SNMP traps and SNMP MIBs to monitor all of the ioMemory devices installed in their network.

Condition	Trap or MIB	GREEN	YELLOW	RED
Device Status	fusionIoDimmInfoState: 2.1.1.1.12	attached(2)	detached(1),	minimal(3)

Condition	Trap or MIB	GREEN	YELLOW	RED
			detaching(5), attaching(6), scanning(7), formatting(8), or updating(9)	or error(4)
Minimal Mode Reason	fusionIoDimmInfoMinimalModeReason: 2.1.1.1.30	7	N/A	Any other value
Power Loss Protection	fusionIoDimmInfoPowerlossProtectDisabled: 2.1.1.1.43	2	N/A	Any other value
Temperature	fusionIoDimmInfoCurrentTemp: 2.1.1.1.24	<90	90-96	97
Health Reserves	fusionIoDimmInfoPercentLifeRemaining: 2.1.1.1.25	>10	4-10	<4
Wearout Indicator	fusionIoDimmInfoWearoutIndicator: 2.1.1.1.21	2	N/A	Any other value
Writeable Indicator	fusionIoDimmInfoNonWritableIndicator: 2.1.1.1.23	2	N/A	Any other value
PCI Power Compatibility	fusionIoDimmInfoPCIPowerCompatibility: 2.1.1.1.46	2048	16	0
Flashback Indicatior	fusionIoDimmInfoFlashbackIndicator: 2.1.1.1.22	2	0	1

Part Number: 00AH240
Printed in the USA