Storage Management Technical Specification, Part 7 Host Elements

Version 1.7.0, Revision 5

Abstract: This SNIA Technical Position defines an interface between WBEM-capable clients and servers for the secure, extensible, and interoperable management of networked storage.

This document has been released and approved by the SNIA. The SNIA believes that the ideas, methodologies and technologies described in this document accurately represent the SNIA goals and are appropriate for widespread distribution. Suggestions for revision should be directed to http://www.snia.org/feedback/.

SNIA Technical Position

8 March, 2016

REVISION HISTORY

Revision 1

Date

8 Sept 2014

SCRs Incorporated and other changes

Memory Configuration Profile
- AddedNew profile to manage NVDIMMs (SMIS-170-Draft-SCR00005)

Persistent Memory Configuration Profile
- AddedNew profile added to manage NVDIMMs (SMIS-170-Draft-SCR00005)

Comments

Editorial notes and DRAFT material are displayed.

Revision 2

Date

18 December 2014

SCRs Incorporated and other changes

Memory Configuration Profile (SMIS-170-Draft-SCR00006)
- Updated due to class and property name changes in DMTF Multi-type Memory profile
- Introduced staged request concept
- Added use cases
- Addressed comments from internal and partner reviews

Persistent Memory Configuration Profile (SMIS-170-Draft-SCR00006)
- Added new material (most of profile content) to manage NVDIMMs (

Comments

Editorial notes and DRAFT material are displayed.

USAGE text was revised to address code.
(now included in the front matter for all SNIA specifications)

Revision 3

Date

20 May 2015

SCRs Incorporated and other changes

All recipes and associated text were removed.

Host Discovered Resources (TSG-SMIS-SCR00315.001)
- Promoted the maturity level from DRAFT to EXPERIMENTAL for update to remove SNIA_ classes and use DMTF CIM_ classes

References
- Added DMTF DSP1054 v1.2.2, Indications Profile (and changed version to 1.2.2 throughout book)

SB Multipath Management (TSG-SMIS-SCR00315.001)
- Promoted the maturity level from DRAFT to EXPERIMENTAL for update to remove SNIA_ classes and use DMTF CIM_ classes

SB Multipath Management Profile
- Added reference to SMI-S Version 1.6.1 Revision 5. No content in this deprecated profile.

Persistent Memory Configuration Profile
- Changes to diagrams approved in Core TWG 3/23/15

Memory Configuration Profile
- Changes to diagrams approved in Core TWG 3/23/15

Comments

Editorial notes and DRAFT material were hidden.

Revision 4

Date

9 September 2015

SCRs Incorporated and other changes

Multiple profiles
- Instances of *subprofile* were changed to *profile*. (TSG meeting voice vote)
- Profile versions and related text were updated. (TSG meeting voice vote)
-- *CIM/XML* was changed to *CIM-XM*L (Response to ballot comments)
- Removed instances of *Experimnental* within profiles already labeled as *Experimental* to avoid confusion and redundancy. (Editorial change)

FC HBA Profile
- Removed the profile content and added text re it being Deprecated. (CORE-SMIS-SCR-000)

iSCSI Initiator Profile
- Removed all occurrences of “(Host Hardware RAID Controller)”. (CORE-SMIS-SCR-000)

Referemces
- Updated DSP 1071, Multi-type System Memory Profile reference here and in profiles.
- Add reference to DSP 1119, 1.0.0b Diagnostics Job Control Profile

Annex A (informative) SMI-S Information Model
- DMTF’s CIM schema version changed to 2.45.0. (TSG meeting voice vote)

Comments

- Editorial notes and DRAFT material were hidden.

Revision 5

Date

9 October 2015

SCRs Incorporated and other changes

Multiple profiles: Addressed SMI-S 1.7.0 Revision 4 TSG ballot comments that were strictly editorial and were approved by voice vote of the TSG.

References
- Removed reference to DSP 1119
- Removed 2.3.

Comments

- Editorial notes were hidden.

Suggestion for changes or modifications to this document should be sent to the SNIA Storage Management Initiative Technical Steering Group (SMI-TSG) at http://www.snia.org/feedback/

USAGE

The SNIA hereby grants permission for individuals to use this document for personal use only, and for corporations and other business entities to use this document for internal use only (including internal copying, distribution, and display) provided that:

1) Any text, diagram, chart, table or definition reproduced shall be reproduced in its entirety with no alteration, and,

2) Any document, printed or electronic, in which material from this document (or any portion hereof) is reproduced shall acknowledge the SNIA copyright on that material, and shall credit the SNIA for granting permission for its reuse.

Other than as explicitly provided above, you may not make any commercial use of this document, sell any or this entire document, or distribute this document to third parties. All rights not explicitly granted are expressly reserved to SNIA.

Permission to use this document for purposes other than those enumerated above may be requested by e-mailing tcmd@snia.org. Please include the identity of the requesting individual and/or company and a brief description of the purpose, nature, and scope of the requested use.

All code fragments, scripts, data tables, and sample code in this SNIA document are made available under the following license:

BSD 3-Clause Software License

Copyright (c) 2016, The Storage Networking Industry Association.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of The Storage Networking Industry Association (SNIA) nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

DISCLAIMER

The information contained in this publication is subject to change without notice. The SNIA makes no warranty of any kind with regard to this specification, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The SNIA shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this specification.

Suggestions for revisions should be directed to http://www.snia.org/feedback/.

Portions of the CIM Schema are used in this document with the permission of the Distributed Management Task Force (DMTF). The CIM classes that are documented have been developed and reviewed by both the SNIA and DMTF Technical Working Groups. However, the schema is still in development and review in the DMTF Working Groups and Technical Committee, and subject to change.

INTENDED AUDIENCE

This document is intended for use by individuals and companies engaged in developing, deploying, and promoting interoperable multi-vendor SANs through the Storage Networking Industry Association (SNIA) organization.

CHANGES TO THE SPECIFICATION

Each publication of this specification is uniquely identified by a three-level identifier, comprised of a version number, a release number and an update number. The current identifier for this specification is version 1.7.0. Future publications of this specification are subject to specific constraints on the scope of change that is permissible from one publication to the next and the degree of interoperability and backward compatibility that should be assumed between products designed to different publications of this standard. The SNIA has defined three levels of change to a specification:

- Major Revision: A major revision of the specification represents a substantial change to the underlying scope or architecture of the SMI-S API. A major revision results in an increase in the version number of the version identifier (e.g., from version 1.x.x to version 2.x.x). There is no assurance of interoperability or backward compatibility between releases with different version numbers.
- Minor Revision: A minor revision of the specification represents a technical change to existing content or an adjustment to the scope of the SMI-S API. A minor revision results in an increase in the release number of the specification's identifier (e.g., from x.1.x to x.2.x). Minor revisions with the same version number preserve interoperability and backward compatibility.
- Update: An update to the specification is limited to minor corrections or clarifications of existing specification content. An update will result in an increase in the third component of the release identifier (e.g., from x.x.1 to x.x.2). Updates with the same version and minor release levels preserve interoperability and backward compatibility.

TYPOGRAPHICAL CONVENTIONS

Maturity Level

In addition to informative and normative content, this specification includes guidance about the maturity of emerging material that has completed a rigorous design review but has limited implementation in commercial products. This material is clearly delineated as described in the following sections. The typographical convention is intended to provide a sense of the maturity of the affected material, without altering its normative content. By recognizing the relative maturity of different sections of the standard, an implementer should be able to make more informed decisions about the adoption and deployment of different portions of the standard in a commercial product.

This specification has been structured to convey both the formal requirements and assumptions of the SMI-S API and its emerging implementation and deployment lifecycle. Over time, the intent is that all content in the specification will represent a mature and stable design, be verified by extensive implementation experience, assure consistent support for backward compatibility, and rely solely on content material that has reached a similar level of maturity. Unless explicitly labeled with one of the subordinate maturity levels defined for this specification, content is assumed to satisfy these requirements and is referred to as "Finalized". Since much of the evolving specification

content in any given release will not have matured to that level, this specification defines three subordinate levels of implementation maturity that identify important aspects of the content's increasing maturity and stability. Each subordinate maturity level is defined by its level of implementation experience, its stability and its reliance on other emerging standards. Each subordinate maturity level is identified by a unique typographical tagging convention that clearly distinguishes content at one maturity model from content at another level.

Experimental Maturity Level

No material is included in this specification unless its initial architecture has been completed and reviewed. Some content included in this specification has complete and reviewed design, but lacks implementation experience and the maturity gained through implementation experience. This content is included in order to gain wider review and to gain implementation experience. This material is referred to as “Experimental”. It is presented here as an aid to implementers who are interested in likely future developments within the SMI specification. The contents of an Experimental profile may change as implementation experience is gained. There is a high likelihood that the changed content will be included in an upcoming revision of the specification. Experimental material can advance to a higher maturity level as soon as implementations are available. Figure 1 is a sample of the typographical convention for Experimental content.

EXPERIMENTAL

Experimental content appears here.

EXPERIMENTAL

Figure 1 - Experimental Maturity Level Tag

Implemented Maturity Level

Profiles for which initial implementations have been completed are classified as “Implemented”. This indicates that at least two different vendors have implemented the profile, including at least one provider implementation. At this maturity level, the underlying architecture and modeling are stable, and changes in future revisions will be limited to the correction of deficiencies identified through additional implementation experience. Should the material become obsolete in the future, it must be deprecated in a minor revision of the specification prior to its removal from subsequent releases. Figure 2 is a sample of the typographical convention for Implemented content.

IMPLEMENTED

Implemented content appears here.

IMPLEMENTED

Figure 2 - Implemented Maturity Level Tag

Stable Maturity Level

Once content at the Implemented maturity level has garnered additional implementation experience, it can be tagged at the Stable maturity level. Material at this maturity level has been implemented by three different vendors, including both a provider and a client. Should material that has reached this maturity level become obsolete, it may only be deprecated as part of a minor revision to the specification. Material at this maturity level that has been deprecated may only be removed from the specification as part of a major revision. A profile that has reached this maturity level is guaranteed to preserve backward compatibility from one minor specification revision to the next. As a result, Profiles at or above the Stable

maturity level shall not rely on any content that is Experimental. Figure 3 is a sample of the typographical convention for Implemented content.

STABLE

Stable content appears here.

STABLE

Figure 3 - Stable Maturity Level Tag

Finalized Maturity Level

Content that has reached the highest maturity level is referred to as “Finalized.” In addition to satisfying the requirements for the Stable maturity level, content at the Finalized maturity level must solely depend upon or refine material that has also reached the Finalized level. If specification content depends upon material that is not under the control of the SNIA, and therefore not subject to its maturity level definitions, then the external content is evaluated by the SNIA to assure that it has achieved a comparable level of completion, stability, and implementation experience. Should material that has reached this maturity level become obsolete, it may only be deprecated as part of a major revision to the specification. A profile that has reached this maturity level is guaranteed to preserve backward compatibility from one minor specification revision to the next. Over time, it is hoped that all specification content will attain this maturity level. Accordingly, there is no special typographical convention, as there is with the other, subordinate maturity levels. Unless content in the specification is marked with one of the typographical conventions defined for the subordinate maturity levels, it should be assumed to have reached the Finalized maturity level.

Deprecated Material

Non-Experimental material can be deprecated in a subsequent revision of the specification. Sections identified as “Deprecated” contain material that is obsolete and not recommended for use in new development efforts. Existing and new implementations may still use this material, but shall move to the newer approach as soon as possible. The maturity level of the material being deprecated determines how long it will continue to appear in the specification. Implemented content shall be retained at least until the next revision of the specialization, while Stable and Finalized material shall be retained until the next major revision of the specification. Providers shall implement the deprecated elements as long as it appears in the specification in order to achieve backward compatibility. Clients may rely on deprecated elements, but are encouraged to use non-deprecated alternatives when possible.

Deprecated sections are documented with a reference to the last published version to include the deprecated section as normative material and to the section in the current specification with the replacement. Figure 4 contains a sample of the typographical convention for deprecated content.

DEPRECATED

Content that has been deprecated appears here.

DEPRECATED

Figure 4 - Deprecated Tag

Contents

LIST OF FIGURES

LIST OF TABLES

FOREWORD

The host-based storage portion of the Storage Management Technical Specification contains profiles and other clauses for management of host-based storage devices. Host-based storage devices provide storage capabilities to a host computer system. Examples of these devices include Fiber Channel Host Bus Adapters, Serial Attached SCSI Host Bus Adapters, RAID Controllers, JBODs (Just-A-Bunch-Of-Disks) and Operating System-discovered storage resources. The host-based profiles describe the manageability required for each device and the connectivity to the host computer system. The host-based profiles leverage existing profiles within this specification, as well as other profiles from the Distributed Management Task Force, where applicable, to create a comprehensive management model.

Parts of this Standard

This standard is subdivided in the following parts:

- *Storage Management Technical Specification, Part 1 Overview, 1.7.0 Rev 5*
- *Storage Management Technical Specification, Part 2 Common Architecture, 1.7.0 Rev 5*
- *Storage Management Technical Specification, Part 3 Common Profiles, 1.7.0 Rev 5*
- *Storage Management Technical Specification, Part 4 Block Devices, 1.7.0 Rev 5*
- *Storage Management Technical Specification, Part 5 Filesystems, 1.7.0 Rev 5*
- *Storage Management Technical Specification, Part 6 Fabric, 1.7.0 Rev 5*
- *Storage Management Technical Specification, Part 7 Host Elements, 1.7.0 Rev 5*
- *Storage Management Technical Specification, Part 8 Media Libraries, 1.7.0 Rev 5*

SNIA Web Site

Current SNIA practice is to make updates and other information available through their web site at http://www.snia.org

SNIA Address

Requests for interpretation, suggestions for improvement and addenda, or defect reports are welcome. They should be sent via the SNIA Feedback Portal at http://www.snia.org/feedback/ or by mail to the Storage Networking Industry Association, 4360 ArrowsWest Drive, Colorado Springs, Colorado 80907, U.S.A.

1 Scope

The host-base storage portion of the Storage Management Technical Specification defines management profiles for autonomous, component and abstract profiles for management of host-based storage devices. The autonomous profiles describe the management of a stand-alone host-based storage entity. The component profiles describe management of aspects of host-based storage entities that may be used by other autonomous profiles. Finally, this section describes abstract profiles that may be used as a basis for creating additional Host-based autonomous profiles.

This version of the Host-based Storage portion of the Storage Management Technical Specification includes autonomous profiles:

- "The Host Discovered Resources Profile

  This profile defines the model for the storage devices presented to an operating system running on a host computer system. In addition, this profile describes the map of storage associated to a host-computer system that a client application can discover.

- "The Fibre Channel HBA Profile

  This profile defines the model and functions of a Fibre Channel HBA that exports block storage to a host computer system from a SAN device (Fibre Channel switch, array, tape library, etc.).

- iSCSI Initiator Profile

  This profile defines the model and functions necessary to manage an iSCSI initiator.

Component profiles used by autonomous profiles to describe aspects of host-based storage elements and services. The component profiles defined in this version of the specification include:

- Host Hardware RAID Controller Profile

  This profile defines the model and functions of a host-based RAID controller that exports block storage to a host computer system from locally attached storage devices (internal hard drives, JBODs, etc.)

- Storage HBA Profile

  This profile defines the model and functions of a SAS, SATA, SPI, or Fibre Channel HBA that exports storage to a host computer system from a SAN device (Fibre Channel switch, array, tape library, etc.).

- Disk Partition Profile

  The Disk Partition profile models partition (or slice) configuration services provided by operating systems on some platforms.

- SB Multipath Management Profile

  The SB Multipath Management Profile models paths (connections between host controllers and device ports) for environments supporting the SB (Single Byte) command protocol.

- SCSI Multipath Management Profile

  The SCSI Multipath Management profile models paths (connections between host controllers, device ports, and logical units) for environments supporting the SCSI command protocol.

2 Normative References

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

2.1 Approved references

Systems Management: Data Storage Management (XDSM) API - ISBN: 1-85912-190-X

DMTF DSP1054 Indications Profile 1.2.2
http://www.dmtf.org/sites/default/files/standards/documents/DSP1054_1.2.2.pdf

2.2 References under development

EXPERIMENTAL

DMTF documents that are works in progress.

DMTF DSP1002, Diagnostics Profile 2.1.0
http://dmtf.org/sites/default/files/standards/documents/DSP1002_2.1.0a.pdf

DMTF DSP 1071, Multi-type System Memory Profile 1.0.0a
http://www.dmtf.org/sites/default/files/standards/documents/DSP1071_1.0.0a.pdf

DMTF DSP1104, FC HBA Diagnostics Profile 1.1.0a
http://dmtf.org/sites/default/files/standards/documents/DSP1104_1.1.0a.pdf

DMTF DSP1113, FC HBA Diagnostics Profile 1.1.0
http://dmtf.org/sites/default/files/standards/documents/DSP1104_1.1.0a.pdf

EXPERIMENTAL

Storage Management Technical Specification, Part 2 Common Architecture, 1.7.0 Rev 5

Storage Management Technical Specification, Part 3 Common Profiles, 1.7.0 Rev 5

3 Terms, Definitions, Symbols, Abbreviations, and Conventions

3.1 General

For the purposes of this document, the terms, definitions, symbols, abbreviations, and conventions given in *Storage Management Technical Specification, Part 2 Common Architecture, 1.7.0 Rev 5* and the following apply.

3.2 Terms and Definitions

3.2.1
Fibre Channel
a serial I/O bus capable of supporting multiple protocols, including access to open system storage (FCP protocol), access to mainframe storage (**FICON™**[1] protocol), and IP

3.2.2
Host Bus Adapter (HBA)
an I/O adapter that connects a host I/O bus to a computer's memory system

3.2.3
host computer system
any computer system to which disks, disk subsystems, or file servers are attached and accessible for data storage and I/O.

3.2.4
JBOD
an acronym for "Just a Bunch Of Disks." A cabinet or enclosure of disks

3.2.5
logical sisk
block storage on which file systems are built

A logical disk would be formatted for a particular file system.

3.2.6
Operating System (OS)
software that manages the resources of a host computer system.

3.2.7
RAID
an Acronym for Redundant Array of Independent Disks, a family of techniques for managing multiple disks to deliver desirable cost, data availability, and performance characteristics to host environments

3.2.8
storage device enclosure
a cabinet or enclosure of storage media devices.

Note 1 to entry: Example: JBOD

3.2.9
storage volume
unit of capacity served from a block storage device

1.FICON**™** is an example of a suitable product available commercially. This information is given for the convenience of users of this standard and does not constitute an endorsement of this product by SNIA or any standards organization.

EXPERIMENTAL

4 Disk Partition Profile

4.1 Description

4.1.1 Synopsis

Profile Name: Disk Partition (Component Profile)

Version: 1.6.0

Organization: SNIA

Central Class: DiskPartitionConfigurationService

Scoping Class: Base Server ComputerSystem

Related Profiles: Not defined in this standard.

4.1.2 Overview

This profile models partition (or slice) configuration services provided by operating systems on some platforms. Some operating systems do not use this type of partitioning. On the operating systems that do, the operating system disk drivers treat partitions as virtual disks. The types of valid partitions are determined by the operating system and the partitioning tools.

We need to consider several operating system variants related to operating system partitions

- On some platforms (e.g., Solaris, Windows), a raw disk volume needs to be partitioned before an application (i.e., a filesystem) uses it. There may be just a single partition on the volume. In these platforms, there is not a name that represents an entire disk volume if that disk volume has multiple partitions.
- On other platforms (e.g., Linux), an application resides on a partition or on the entire disk volume.
- Different operating systems have incompatible partitioning approaches and on-disk data structures (e.g disk labels or partition tables). This specification refers to these approaches as styles. Each style may be supported by multiple operating systems, and most operating systems support multiple styles. The styles supported in this profile are MBR (used on all operating systems running on X86 hardware), vtoc (Solaris and other operating systems with a BSD heritage), and GPT (an emerging style that supports multi-terabyte disk volumes).
- Some styles support multiple tiers of partitions - a partition at one tier may have sub-partitions. On Windows, extended partitions are also a second tier with MBR partitions at each tier.
- Some operating systems utilize two tiers of partitions with different styles at different tiers. For example, BSD-derived Unix variants running on X86 platforms: the lower tier is the X86 BIOS-supported MBR partitions; BSD-style slices can be installed on one of the MBR partitions.
- Some operating systems (AIX, HP_UX) have no equivalent to partitioning.
- Some partition styles have a fixed number of partitions (dependent on the partition type); the user can’t create or delete partitions, just adjust the properties of one of the pre-defined partitions.

A partitioned disk volume has an associated partition table. The partition table contains information about the partitions on the disk volume – the starting address, length, and (in some cases) the type of the partition. In certain cases, a partition table can be associated with a partition; allowing multiple tiers of partitions.

In order for storage applications (e.g., logical volume managers, filesystems, databases) to use a disk volume, the operating system provides a name for the volume. These names appear to be filenames but are part of one (or a few) special namespaces managed by the operating system. Windows drive letters and Unix /dev/ directories are examples of the special namespaces. Any extent that is consumable by storage applications is modeled a LogicalDisk; the LogicalDisk.Name property provides this special filename. The exported extent resulting from a partition is a LogicalDisk; on systems that do not require partitions, each usable disk volume has a LogicalDisk instance that models the operating system name. Extents that are not available for storage applications are modeled as StorageExtent (or StorageExtent subclasses other than LogicalDisk) instances and have a name derived from the underlying hardware and partition number.

Operating systems may have different partition styles. The most common style is the MBR (Master Boot Record) style used on x86 PCs. This style supports four primary partitions on a disk volume with an optional second-tier (extended/logical partitions). Solaris uses a style called VTOC that is derived from and similar to BSD partitions. VTOC supports eight partitions. On Solaris X86, VTOC is installed in one X86 MBR primary partition for compatibility with other x86 operating systems. GPT is a new set of interfaces for x86 64-bit environments and includes a partitioning style. Of particular note is that GPT partitions can exceed the two-terabyte limit associated with other partition styles. So many vendors are migrating towards GPT as an option for supporting larger volumes. This profile includes separate specialized subclasses for MBR, VTOC, and GPT partitions. Their relationship is summarized in Figure 5.

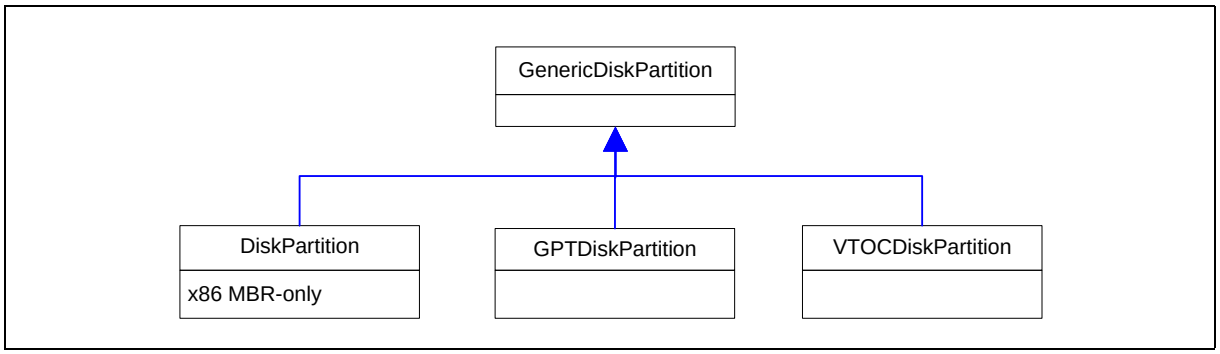

Figure 5 - Disk Partition Class Hierarchy

This profile includes a partition configuration service class that allows a client to create partition tables and modify partitions. It also includes a partition configuration capabilities class that describes the partition configuration capabilities of the system. Separate capabilities instances describe each partition style supported on the system. There shall be one instance of DiskPartitionConfigurationService, as shown in Figure 6.

All references to ComputerSystem in the Disk Partition Profile implies a single instance for a customer server or storage system as defined in the Base Server Profile. See Annex A: (Informative) Host Profile Deployment Guidelines, 1.6.0 Rev 3 for information on the use of host profiles with Base Server profile.

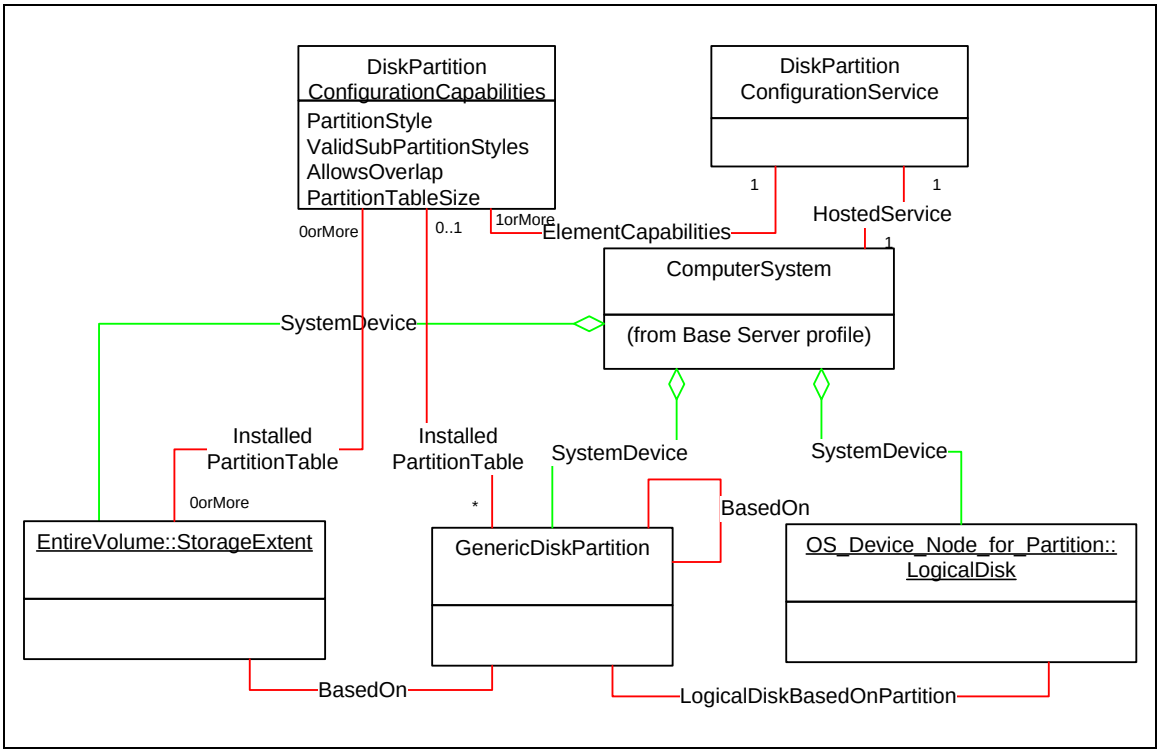

Figure 6 - Disk Partition Class Diagram

4.1.3 Background on X86 MBR Partitions

The terminology used in X86 partition applications is somewhat confusing and masks the actual configurations. The MBR style supports two tiers of partitions; up to four partitions at the entire disk volume tier and up to four partitions within each of these top-tier partitions. An MBR *primary* partition is a top-tier partition that is not sub-partitioned. An MBR *extended* partition is a top-tier partition that is sub-partitioned. An MBR *logical* partition is a sub-partition of an extended partition.

Figure 7 represents the actual layout of an MBR drive with three usable partitions – with Windows/DOS driver letter names.

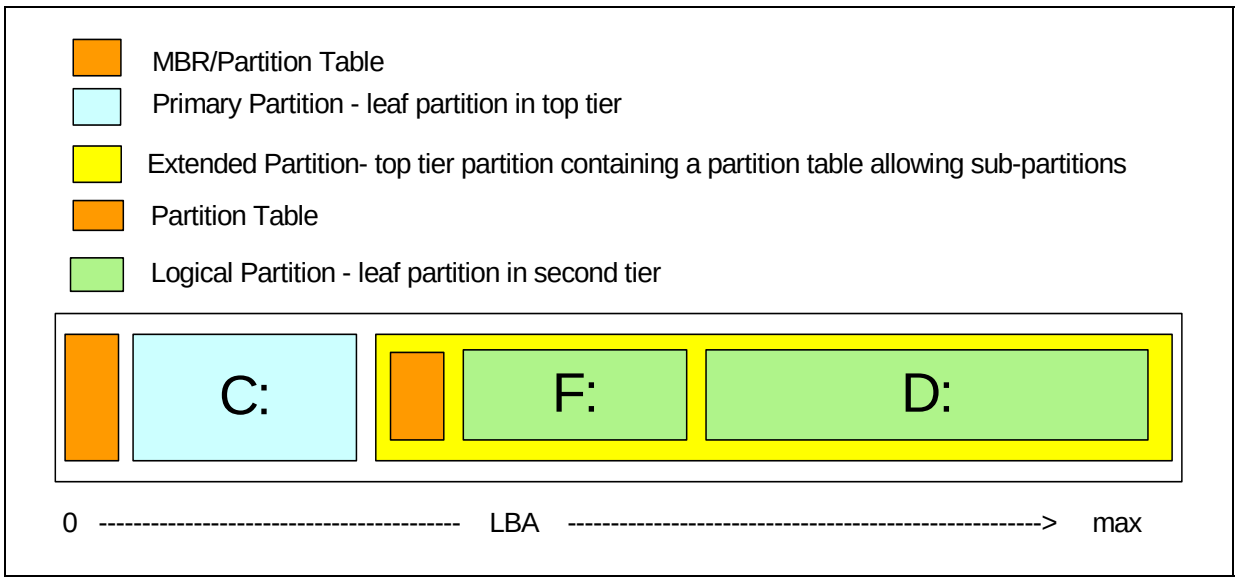

Figure 7 - Disk MBR Partition Example

C: is a primary partition and F: and D: are logical partitions that share an extended partition. Note that the partitions drive letters (C:, F:, and D:) are not in alphabetical order; the assignment of drive letters under Windows/DOS is decoupled from the partitioning logic.

Figure 8 is an instance diagram of the SMI-S classes describing this configuration. Technically, the MBR/ Partition tables could be considered to be small partitions. operating systems generally hide these sectors and treat the effective disk volume as starting just after the MBR. Rather than complicate the SMI-S model, these MBR areas are just ignored and the consumable block size is reduced by the appropriate value (the PartitionTableSize property of DiskPartitionConfigurationCapabilities). In the SMI-S model, the InstalledPartitionTable association to the containing extent indicates the presence of a disk label and/or partition table. In Figure 8, the extent representing the entire disk volume (on the lower left) and the top-tier partition to the right each contain a partition table and are each associated to DiskPartitionConfigurationCapabilities via an InstalledPartitionTable association.

In Figure 8 the StorageExtent at the lower left represents the entire disk volume and the two “top-tier” partitions are based on this extent. The LogicalDisk instances at the top represent the consumable partitions C:, F:, and D:.

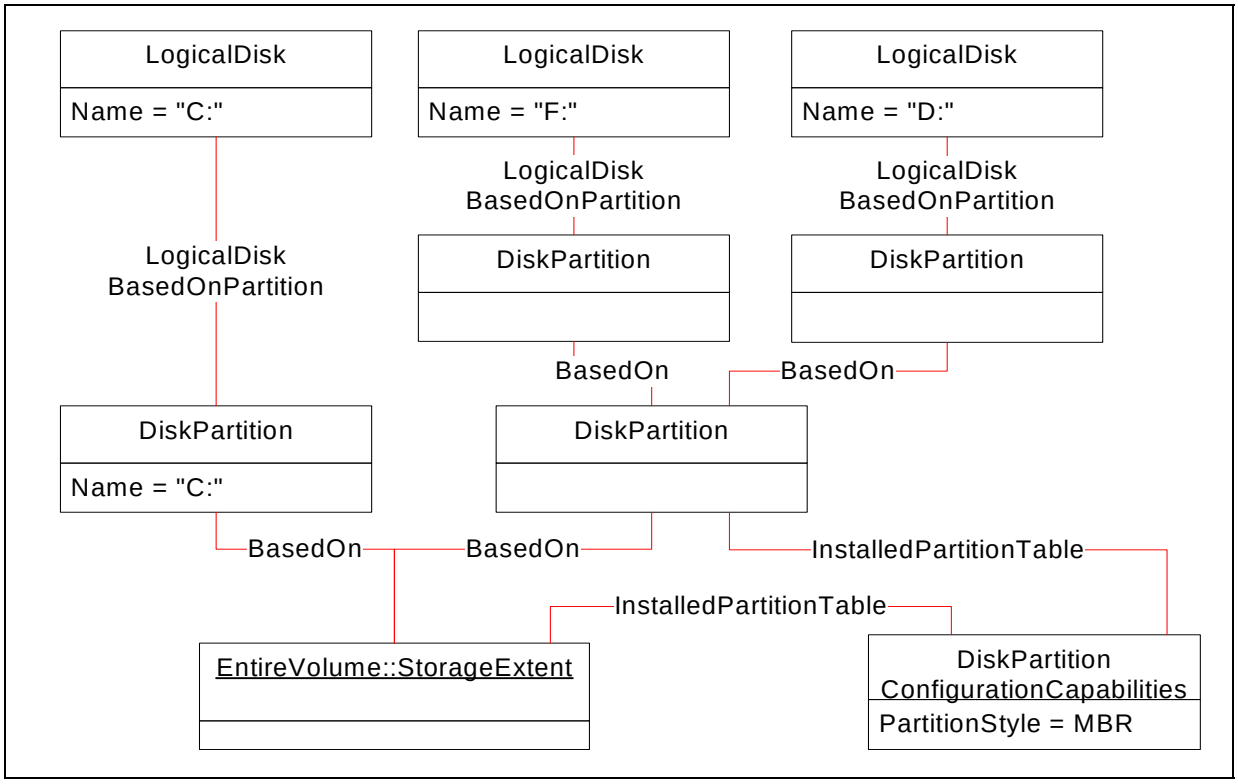

Figure 8 - MBR Partition Instance Diagram

Figure 9 models a similar configuration where the one top-tier partition contains a Solaris X86 installation. In this case, the instrumentation instantiates two instances of DiskPartitionConfigurationCapabilities, one for the top-tier MBR partition table and one for the vtoc partition table.

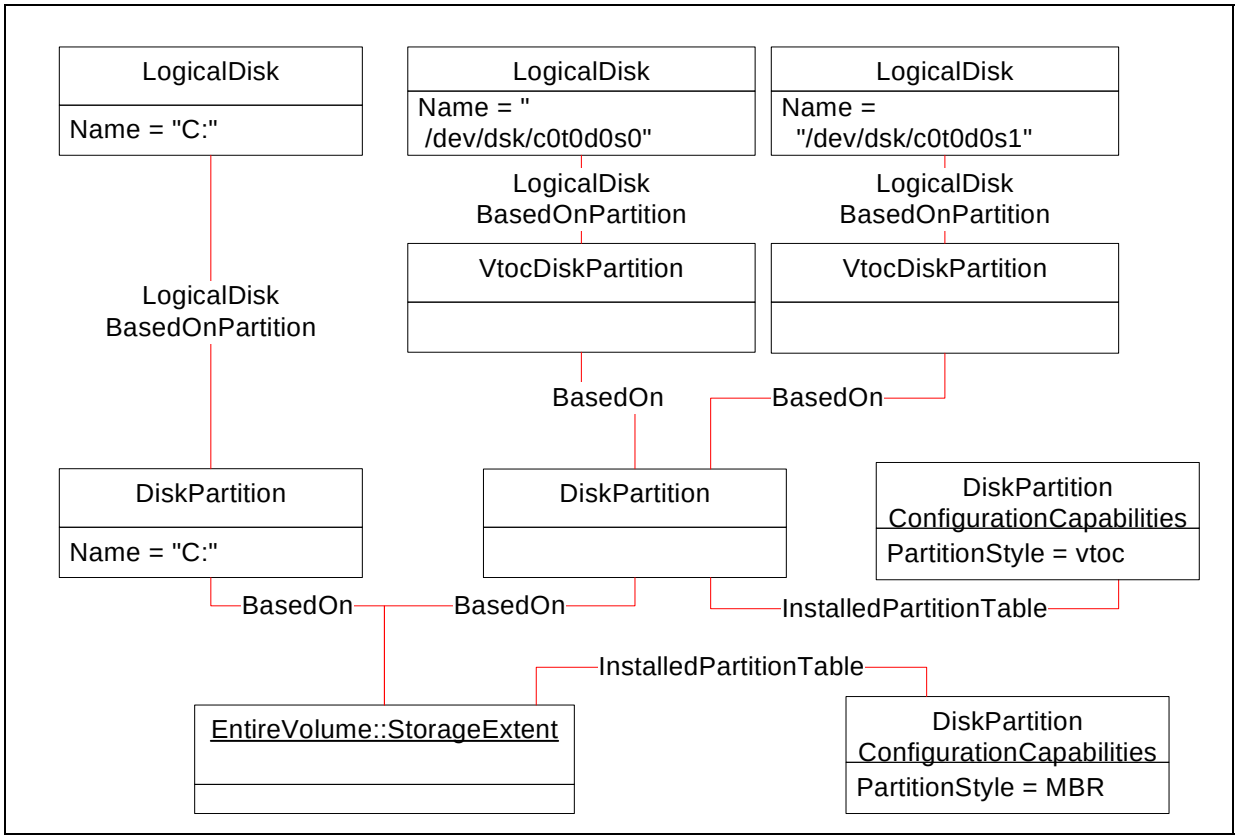

Figure 9 - MBR and VTOC Partition Instance Diagram

Table 1 summarizes likely values for capabilities properties and suggested Name properties on various operating systems

Table 1 - Capabilities Properties

Property	X86 MBR	vtoc	GPT			
			Win	Linux	Solaris SPARC	Solaris X86
Overlap Allowed	Depends on applications	true	false			
MaxCapacity	2 terabytes (2^32 blocks)	2 terabytes	2^64 blocks			
MaxNumberOfPartitions	4	8	128	15	127	127

The sizes and starting/ending addresses shall be consistent between the associated LogicalDisk, DiskPartition, and LogicalDisk instances. Figure 10 shows the classes with size information.

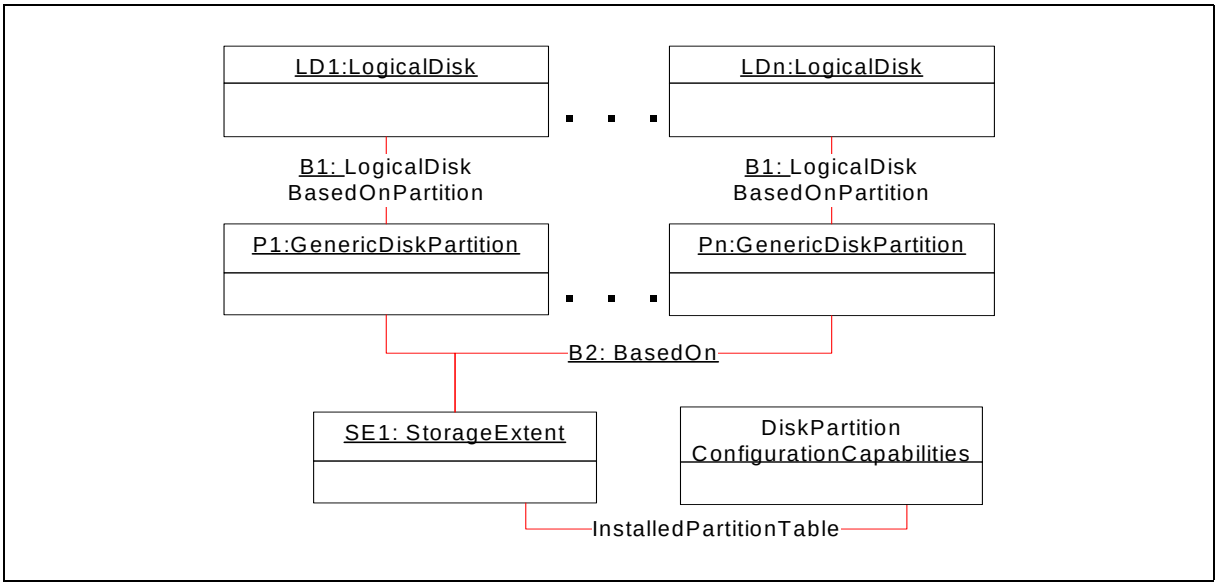

Figure 10 - Partition Instance Diagram for Size/Address Rules

In this diagram, partitions P1,... Pn are all based on the same underlying disk volume (or partition) SE1.

- The NumberOfBlocks shall be the same for a LogicalDisk and its underlying partition (for example, LD1 and P1 in the diagram).
- The StartingAddress in the LogicalDiskBasedOnPartition associations (B1 in the diagram) between a LogicalDisk and its underlying partition shall be 0. The EndingAddress in this association shall be one less than NumberOfBlocks from either the LogicalDisk or partition.
- The NumberOfBlocks for each partition (P1, ... Pn) shall be equal to the values of EndingAddress-StartingAddress+1 of the underlying BasedOn association (B2 in the diagram).
- DiskPartitionConfigurationCapabilities.PartitionTableSize shall hold the total number of blocks consumed by metadata (volume label, boot record, and partition tables) for the associated StorageExtent. For MBR and VTOC styles, this is a fixed value. For GPT, this value could in theory be larger for large extents. Separate instances of DiskPartitionConfigurationCapabilities shall be instantiated as needed to allow different values of PartitionTableSize.
- The size of maintenance tracks or cylinders shall not be included StorageExtent.NumberOfBlocks. This size may be included in DiskPartitionConfigurationCapabilities.PartitionTableSize.
- If DiskPartitionConfigurationCapabilities.OverlapAllowed is false, then the sum of the NumberOfBlocks properties for all partitions plus DiskPartitionConfigurationCapabilities.PartitionTableSize shall not exceed the value of NumberOfBlocks for the underlying StorageExtent. Other than that, there is no guaranteed relationship between StorageExtent.NumberOfBlocks and the sum of the NumberOfBlock values for partitions BasedOn the StorageExtent.

4.2 Health and Fault Management Considerations

No health information is required in LogicalDisk or partition instances. Clients should assume that the health-related properties of the underlying StorageExtent apply to all partitions and LogicalDisks based on that extent.

4.3 Supported Profiles and Packages

Not defined in this standard

4.4 Methods of the Profile

4.4.1 SetPartitionStyle

This method installs a partition table on an extent of the specified partition style, creates DiskPartition instances if SettingStyleInstantiatedPartitions is non-zero, and BasedOn associations between the underlying extent and the new partition instances. As a side effect, the usable block size of the underlying extent is reduced by the block size of the metadata reserved by the partition table and associated metadata. This size is in the PartitionTableSize property of the associated DiskPartitionConfigurationCapabilities instance.

```
uint32 SetPartitionStyle (

     [IN, Description (
         "A reference to the extent (volume or partition) where "
         "this style (partition table) will be installed.")]
   CIM_StorageExtent REF Extent,

     [IN, Description (
         "A reference to the "
         "DiskPartitionConfigurationCapabilities instance "
         "describing the desired partition style.")]
   CIM_DiskPartitionConfigurationCapabilities REF PartitionStyle );
```

4.4.2 CreateOrModifyPartition

This method creates a new partition if the Partition parameter is null or modifies the partition specified. If the starting and ending address parameters are null, the resulting partition will occupy the entire underlying extent. If a the DeviceFileName parameter is non-null, a LogicalDisk instance is created and associated via LogicalDiskBasedOnPartition to the partition. The underlying extent shall be associated to a capabilities class describing the installed partition style (partition table); this association is established using

```
uint32 CreateOrModifyPartition (
       [IN, Description (
           "A reference to the underlying extent the partition is "
           "base on.")]
    CIM_StorageExtent REF extent,

       [IN, Description (
           "The starting block number.")]
    uint64 StartingAddress,

       [IN, Description (
           "The ending block number.")]
    uint64 EndingAddress,

       [IN, Description (
           "The platform-specific special file name to be assigned "
```

```
        "to the LogicalDisk instance BasedOn the new "
        "DiskPartition instance.")]
string DeviceFileName,

   [IN, OUT, Description (
        "A reference an existing partition instance to modify or "
        "null to request a new partition.")]
CIM_GenericDiskPartition REF Partition);
```

Intrinsic delete operation will delete the disk partition and all storage extents that are Dependent on the disk partition.

4.5 Client Considerations and Recipes

4.5.1 Client Considerations

A client discovers partition configuration support by looking for instances of DiskPartitionConfigurationService. If no service instances are available, then this operating system does not support disk partitions and the client can assume that any LogicalDisk instance is consumable by applications (such as volume managers or filesystems). For operating systems that do support partitioning, the client can discover whether a particular extent is partitioned by looking for a InstalledPartitionTable instance associated with the extent. The client can discover the existing partition configuration by following BasedOn associations between the extent and GenericDiskPartition instances.

For each discovered service, there shall be one or more instances of DiskPartitionConfigurationCapabilities. There is exactly one capabilities instance per Partition Style. If multiple capabilities instances are discovered, the client should look at the PartitionStyle property to determine the services that apply to entire disk volumes and those that apply to partitions.

4.5.2 Recipes

No recipes are included in this version of the standard.

4.6 CIM Elements

Table 2 describes the CIM elements for Disk Partition.

Table 2 - CIM Elements for Disk Partition

Element Name	Requirement	Description
4.6.1 CIM_BasedOn (Partition to Extent)	Mandatory	The disk partitions that are dependent on a storage extent.
4.6.2 CIM_BasedOn (Partition to Partition)	Mandatory	The disk partitions that are dependent on a disk partition.
4.6.3 CIM_DiskPartition	Optional	
4.6.4 CIM_DiskPartitionConfigurationCapabilities	Mandatory	
4.6.5 CIM_DiskPartitionConfigurationService	Mandatory	
4.6.6 CIM_ElementCapabilities	Mandatory	
4.6.7 CIM_ElementConformsToProfile (DiskPartitionConfigurationService to Disk Partition RegisteredProfile)	Mandatory	Ties the DiskPartitionConfigurationService to the registered profile for Disk Partition.
4.6.8 CIM_GPTDiskPartition	Optional	

Table 2 - CIM Elements for Disk Partition

Element Name	Requirement	Description
4.6.9 CIM_GenericDiskPartition	Mandatory	
4.6.10 CIM_HostedService	Mandatory	
4.6.11 CIM_InstalledPartitionTable (Capabilities to Extent)	Mandatory	The disk partition capabilities for disk partitions that are dependent on a storage extent.
4.6.12 CIM_InstalledPartitionTable (Capabilities to Partition)	Mandatory	The disk partition capabilities for disk partitions that are dependent on a storage extent.
4.6.13 CIM_LogicalDisk	Optional	
4.6.14 CIM_LogicalDiskBasedOnPartition (LogicalDisk to Partition)	Mandatory	The logical disks that are dependent on a disk partition.
4.6.15 CIM_StorageExtent	Mandatory	
4.6.16 CIM_SystemDevice (System to Extent)	Mandatory	The storage extent managed by a Base Server profile computer system.
4.6.17 CIM_SystemDevice (System to LogicalDisk)	Mandatory	The logical disk that is managed by a Base Server profile computer system.
4.6.18 CIM_SystemDevice (System to Partition)	Mandatory	The disk partition managed by a Base Server profile computer system.
4.6.19 CIM_VTOCDiskPartition	Optional	
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_GenericDiskPartition	Mandatory	Partition Creation.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_GenericDiskPartition	Mandatory	Partition Deletion.

4.6.1 CIM_BasedOn (Partition to Extent)

Created By: Extrinsic: CreateOrModifyPartition

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 3 describes class CIM_BasedOn (Partition to Extent).

Table 3 - SMI Referenced Properties/Methods for CIM_BasedOn (Partition to Extent)

Properties	Flags	Requirement	Description & Notes
StartingAddress		Mandatory	
EndingAddress		Mandatory	
Dependent		Mandatory	
Antecedent		Mandatory	

4.6.2 CIM_BasedOn (Partition to Partition)

Created By: Extrinsic: CreateOrModifyPartition

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 4 describes class CIM_BasedOn (Partition to Partition).

Table 4 - SMI Referenced Properties/Methods for CIM_BasedOn (Partition to Partition)

Properties	Flags	Requirement	Description & Notes
StartingAddress		Mandatory	
EndingAddress		Mandatory	
Antecedent		Mandatory	
Dependent		Mandatory	

4.6.3 CIM_DiskPartition

CIM_DiskPartition (MBR) is subclassed from CIM_GenericDiskPartition.

Created By: Extrinsic: CreateOrModifyPartition
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 5 describes class CIM_DiskPartition.

Table 5 - SMI Referenced Properties/Methods for CIM_DiskPartition

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	
Name		Mandatory	
NumberOfBlocks		Optional	
NameFormat		Optional	
NameNamespace		Optional	
PartitionSubtype		Optional	
PartitionType		Optional	
PrimaryPartition		Optional	

4.6.4 CIM_DiskPartitionConfigurationCapabilities

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 6 describes class CIM_DiskPartitionConfigurationCapabilities.

Table 6 - SMI Referenced Properties/Methods for CIM_DiskPartitionConfigurationCapabilities

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	Shall specify the unique identifier for an instance of this class within the Implementation namespace.
ElementName		Mandatory	User friendly name.
PartitionStyle		Mandatory	The partition style (i.e partition table type) associated with this capabilities instance.
ValidSubPartitionStyles		Optional	Some partitions can act as a container for other partitions.
MaxNumberOfPartitions		Mandatory	The maximum number of partitions that can be BasedOn the Underlying extent.
MaxCapacity		Mandatory	The largest partition size (in blocks) of this style supported on this platform.
OverlapAllowed		Mandatory	The platform supports partitions with overlapping address ranges.
PartitionTableSize		Mandatory	The number of blocks occupied by the partition table and other metadata.

4.6.5 CIM_DiskPartitionConfigurationService

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 7 describes class CIM_DiskPartitionConfigurationService.

Table 7 - SMI Referenced Properties/Methods for CIM_DiskPartitionConfigurationService

Properties	Flags	Requirement	Description & Notes
CreationClassName		Mandatory	
Name		Mandatory	
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
PartitioningSchemes		Optional	Describes the partitioning schemes supported by the platform.
SetPartitionStyle()		Mandatory	
CreateOrModifyPartition()		Mandatory	

4.6.6 CIM_ElementCapabilities

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 8 describes class CIM_ElementCapabilities.

Table 8 - SMI Referenced Properties/Methods for CIM_ElementCapabilities

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	Reference to the DiskPartitionConfigurationService.
Capabilities		Mandatory	Reference to the DiskPartitionConfigurationCapabilities.

4.6.7 CIM_ElementConformsToProfile (DiskPartitionConfigurationService to Disk Partition RegisteredProfile)

The CIM_ElementConformsToProfile ties DiskPartitionConfigurationService to the registered profile for Disk Partition.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 9 describes class CIM_ElementConformsToProfile (DiskPartitionConfigurationService to Disk Partition RegisteredProfile).

Table 9 - SMI Referenced Properties/Methods for CIM_ElementConformsToProfile (DiskPartitionConfigurationService to Disk Partition RegisteredProfile)

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	A DiskPartitionConfigurationService instance that represents the Disk Partition.
ConformantStandard		Mandatory	RegisteredProfile instance describing the Disk Partition profile.

4.6.8 CIM_GPTDiskPartition

CIM_GPTDiskPartition (GPT) is subclassed from CIM_GenericDiskPartition.

Created By: Extrinsic: CreateOrModifyPartition

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 10 describes class CIM_GPTDiskPartition.

Table 10 - SMI Referenced Properties/Methods for CIM_GPTDiskPartition

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	
Name		Mandatory	

Table 10 - SMI Referenced Properties/Methods for CIM_GPTDiskPartition

Properties	Flags	Requirement	Description & Notes
NumberOfBlocks		Optional	
PartitionType		Optional	

4.6.9 CIM_GenericDiskPartition

GenericDiskPartition is used for subclassing the various partition styles, including MBR, GPT, and VTOC.

Created By: Extrinsic: CreateOrModifyPartition
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 11 describes class CIM_GenericDiskPartition.

Table 11 - SMI Referenced Properties/Methods for CIM_GenericDiskPartition

Properties	Flags	Requirement	Description & Notes
SystemCreationClassNam e		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	
Name		Mandatory	
NumberOfBlocks		Optional	

4.6.10 CIM_HostedService

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 12 describes class CIM_HostedService.

Table 12 - SMI Referenced Properties/Methods for CIM_HostedService

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	Reference to the ComputerSystem in Base Server.
Dependent		Mandatory	Reference to the DiskPartitionConfigurationService.

4.6.11 CIM_InstalledPartitionTable (Capabilities to Extent)

Created By: Static
Modified By: Static
Deleted By: Static

Requirement: Mandatory

Table 13 describes class CIM_InstalledPartitionTable (Capabilities to Extent).

Table 13 - SMI Referenced Properties/Methods for CIM_InstalledPartitionTable (Capabilities to Extent)

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

4.6.12 CIM_InstalledPartitionTable (Capabilities to Partition)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 14 describes class CIM_InstalledPartitionTable (Capabilities to Partition).

Table 14 - SMI Referenced Properties/Methods for CIM_InstalledPartitionTable (Capabilities to Partition)

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

4.6.13 CIM_LogicalDisk

Created By: Extrinsic: CreateOrModifyPartition
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 15 describes class CIM_LogicalDisk.

Table 15 - SMI Referenced Properties/Methods for CIM_LogicalDisk

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	
Name		Mandatory	
NameFormat		Mandatory	OS Device Name.
NameNamespace		Mandatory	OS Device Namespace.

Table 15 - SMI Referenced Properties/Methods for CIM_LogicalDisk

Properties	Flags	Requirement	Description & Notes
OperationalStatus		Mandatory	
NumberOfBlocks		Optional	

4.6.14 CIM_LogicalDiskBasedOnPartition (LogicalDisk to Partition)

Created By: Extrinsic: CreateOrModifyPartition
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 16 describes class CIM_LogicalDiskBasedOnPartition (LogicalDisk to Partition).

Table 16 - SMI Referenced Properties/Methods for CIM_LogicalDiskBasedOnPartition (LogicalDisk to Partition)

Properties	Flags	Requirement	Description & Notes
StartingAddress		Mandatory	Shall be 0.
EndingAddress		Mandatory	Shall be one less than NumberOfBlocks from either the LogicalDisk or Partition.
Dependent		Mandatory	
Antecedent		Mandatory	

4.6.15 CIM_StorageExtent

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 17 describes class CIM_StorageExtent.

Table 17 - SMI Referenced Properties/Methods for CIM_StorageExtent

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	
Name		Mandatory	
OperationalStatus		Mandatory	
NumberOfBlocks		Optional	

4.6.16 CIM_SystemDevice (System to Extent)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 18 describes class CIM_SystemDevice (System to Extent).

Table 18 - SMI Referenced Properties/Methods for CIM_SystemDevice (System to Extent)

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	Reference to the ComputerSystem in Base Server.
PartComponent		Mandatory	

4.6.17 CIM_SystemDevice (System to LogicalDisk)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 19 describes class CIM_SystemDevice (System to LogicalDisk).

Table 19 - SMI Referenced Properties/Methods for CIM_SystemDevice (System to LogicalDisk)

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	Reference to the ComputerSystem in Base Server.
PartComponent		Mandatory	

4.6.18 CIM_SystemDevice (System to Partition)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 20 describes class CIM_SystemDevice (System to Partition).

Table 20 - SMI Referenced Properties/Methods for CIM_SystemDevice (System to Partition)

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	Reference to the ComputerSystem in Base Server.
PartComponent		Mandatory	

4.6.19 CIM_VTOCDiskPartition

CIM_VTOCDiskPartition (VTOC) is subclassed from CIM_GenericDiskPartition.

Created By: Extrinsic: CreateOrModifyPartition

Modified By: Static
Deleted By: Static
Requirement: Optional

Table 21 describes class CIM_VTOCDiskPartition.

Table 21 - SMI Referenced Properties/Methods for CIM_VTOCDiskPartition

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	
Name		Mandatory	
NumberOfBlocks		Optional	
AsciiLabel		Optional	
Flags		Optional	
Tag		Optional	

EXPERIMENTAL

DEPRECATED

5 FC HBA Profile

The functionality of the FC HBA profile has been subsumed by 6 Storage HBA Profile.

See the latest version of this profile in SMI-S Version 1.6.1 Revision 5

DEPRECATED

EXPERIMENTAL

6 Storage HBA Profile

6.1 Synopsis

Profile Name: Storage HBA (Component Profile)

Version: 1.6.1

Organization: SNIA

Central Class: PortController

Scoping Class: ComputerSystem

Related Profiles: Table 22 describes the related profiles for Storage HBA.

Table 22 - Related Profiles for Storage HBA

Profile Name	Organization	Version	Requirement	Description
Software Inventory	SNIA	1.0.0	Mandatory	
Software Update	DMTF	1.0.0	Optional	
Physical Asset	DMTF	1.0.0a	Optional	
FC HBA Diagnostics	DMTF	1.1.0a	Optional	See DSP1104, version 1.1.0a
Diagnostics Job Control	DMTF	1.0.0b	Conditional	Conditional requirement: Required if the FC HBA Diagnostics profile is supported. See DSP1119, version 1.1.0b
Indications	DMTF	1.2.2	Mandatory	See DSP1054, version 1.2.2
SAS Initiator Ports	SNIA	1.4.0	Support for at least one is mandatory.	Conditional requirement: Required if the FC HBA Diagnostics profile is supported.
FC Initiator Ports	SNIA	1.7.0		Conditional requirement: Required if the FC HBA Diagnostics profile is supported.
FCoE Initiator Ports	SNIA	1.7.0		Conditional requirement: Required if the FC HBA Diagnostics profile is supported.

6.2 Description

The Storage HBA Profile represents the manageable elements of an HBA and optionally, the storage connected to it. An HBA can be connected to disks contained within a server's internal drive cage or an external drive enclosure or array. The profile does not include enclosure management of storage devices connected to the HBA. Storage device enclosure management is performed via the Storage Enclosure Profile.

This profile further describes how the classes are to be used to satisfy various use cases and offers suggestions to agent implementers and client application developers. Only the classes unique to a HBA are described by this profile. Other classes that are common to other profiles, reference to by the Storage HBA Profile, may be found in their respective section within this standard, DMTF specifications, or the DMTF CIM schema specification.

6.3 Implementation

6.3.1 Health and Fault Management Consideration

Not defined in this standard

6.3.2 Cascading Considerations

Not defined in this standard

6.3.3 Storage HBA Model Overview

The PortController class is the central class of the Storage HBA Profile. It represents an instance of an HBA. The PortController shall be associated to one or more instances of LogicalPort (defined in initiator port profiles) using the ControlledBy association. The PortController shall be associated to the ComputerSystem (from a referencing profile) using the SystemDevice association. The PortController shall also be associated to Product using the ProductElementComponent association; properties of Product provide information about the HBAs manufacturer and model.

6.3.3.1 PhysicalPackage Requirements

When the instrumentation is representing a physical HBA (as opposed to virtual HBA in a guest virtual machine), PortController shall be associated to PhyscialPackge (or a subclass such as Card), as shown in Figure 11. Physical Package shall be associated to PortController via Realizes and Product via ProductPhyscialComponent.If the instrumentation is running in a guest virtual machine and representing a virtual HBA, PhyscialPackge, Realizes, and ProductPhysicalComponent shall not be instantiated. See 6.3.12 for additional virtual system considerations.

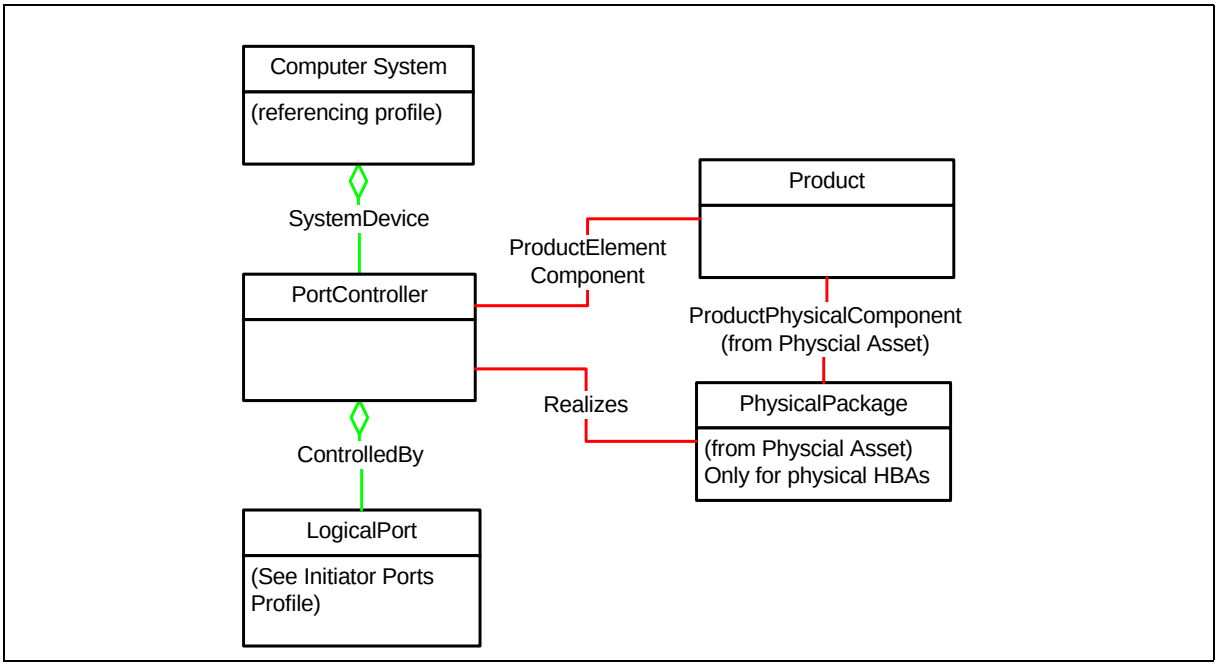

Figure 11 - Model Overview

6.3.4 CIM_ComputerSystem

In Storage HBA, the ComputerSystem Class in the diagrams represents the host containing the HBA and is not defined as part of this profile. Typically, this ComputerSystem will be defined as part of a shim profile or the Base Server Profile. Many of the other classes in the Storage HBA Profile are associated to the instance of ComputerSystem that represents the Host. This includes drives, logical ports, and physical cards.

6.3.5 Profile Registration Profile

For the Storage HBA Profile, the scoping class methodology of profile registration shall be used, as required by the Server Profile. The scoping class is the ComputerSystem in the referencing profile.

However, an implementation may use the central class methodology of profile registration from the Server Profile. The central class of the Storage HBA Profile is the instance of CIM_PortController. The instance of RegisteredProfile in the Interop namespace shall have an association to the instance of CIM_PortController in the implementation namespace, as shown in Figure 12.

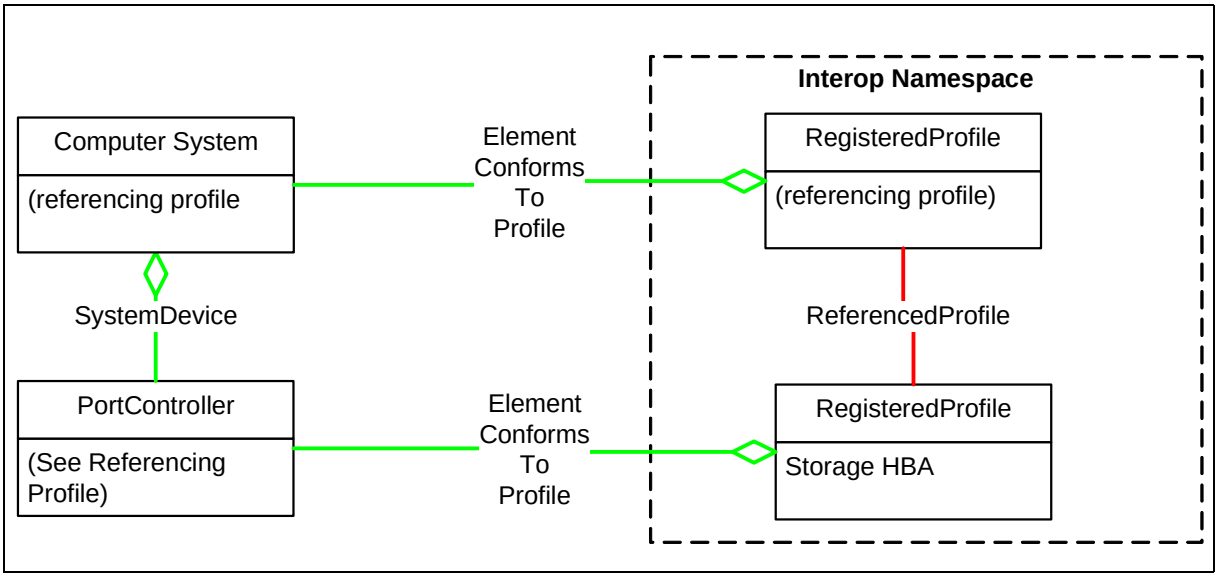

Figure 12 - Profile Registration Profile

6.3.6 Generic Initiator Ports Profile

The Storage HBA Profile utilizes specializations of the Generic Initiator Ports Profile to model the back-end ports of the HBA that are connected to the storage managed by the HBA.

6.3.6.1 CIM_LogicalPort

CIM_LogicalPort is defined in initiator port profiles and represents the logical transport port on the back-end of the HBA that is connected to the storage. This storage could be a drive cage housed inside the host or a storage device enclosure, like a JBOD. The LogicalPort class is intended to model the transport for storage commands in an abstract and agnostic manner. For example, the LogicalPort could represent a Parallel SCSI, SAS, SATA, PATA, or FC port depending on the controller implementation. Thus, the instance of this class shall be sub-classed to SPIPort, SASPort, FCPort or ATAPort depending on the subclass that best represents the transport type the HBA supports for the backend port. The implementation shall not instantiate LogicalPort.

6.3.6.2 CIM_ProtocolEndpoint

The ProtocolEndpoint class represents both ends of the logical data path between the HBA and storage where the storage protocol is transmitted. ProtocolEndpoints for the interface to disks are part of initiator ports profiles.

Like LogicalPort, the ProtocolEndpoint class is intended to model the management of storage protocol in an abstract manner. For example, the ProtocolEndpoint could represent a SCSI, ATA or SB protocol. Thus, the instance of the ProtocolEndpoint shall be subclassed to SCSIProtocolEndpoint, ATAProtocolEndpoint or SBProtocolEndpoint.

6.3.7 Software Inventory Profile

For the Storage HBA Profile, the SoftwareIdentity class from the Software Inventory Profile (DMTF) is required to model various software and firmware entities related to an HBA. Each SoftwareIdentity instance shall be associated via InstalledSOftwareIdentity to the referencing ComputerSystem and via ElementSoftwareIdentity to the PortController.

6.3.7.1 Physical HBA Considerations

The implementation shall use the Software Inventory Profile to model the firmware for the HBA. The SoftwareIdentity instance representing the driver software shall include 10 (Firmware) in the Classifications property. In the context of this profile, “Firmware” refers to firmware installed in and running on the HBA itself.

If the HBA has a separate entity for the BIOS or FCode from the firmware, the implementation may use the Software Inventory Profile to represent the BIOS/FCode. The SoftwareIdentity instance representing the driver software shall include 11 (BIOS/FCode) in the Classifications property. In the context of this profile, “BIOS/FCode” refers to BIOS/FCode extensions installed in and running on the HBA itself. Typically, these extensions are related to boot support.

The implementation shall use the Software Inventory Profile to model the driver software that interfaces with the HBA. As used here, the term “Driver” applies to software (or firmware) installed in the system OS that enables OS support for the HBA.

- If the HBA in installed in a system (a hypervisor or a general-purpose system) following a typical open-system architecture, drivers are components the OS running on the server and provide support for various devices. The Driver SoftwareIdentity in this profile models the driver(s) which enable support for the HBA.
- If the HBA is installed in a general purpose system that does not have something called drivers (for example, some mainframe OSes), then the Driver SoftwareIdentity in this profile models the OS components which enable support for the HBA.
- If the HBA is installed in a hypervisor and the hypervisor implementation is considered firmware rather than an OS, then the Driver SoftwareIdentity represents the subset of that firmware which enables support for the HBA.

In some cases, the driver is an embedded component of the OS (or RTOS or firmware) and is not independently versioned. In this case, the implementation shall populate SoftwareIdentity version information with the version information from the OS.

The use-case for requiring driver version numbers is that there may be dependencies between version numbers of the driver component of the hypervisor and the driver component running in the guest VM. Requiring each allows a client application to track these versions and report inconsistencies.

6.3.7.2 Virtual HBA Considerations

The implementation shall use the Software Inventory Profile, shown in Figure 13, to model the driver software that interfaces with the HBA. As used here, the term “Driver” applies to software installed in the virtual system OS, enabling the OS to interface to the virtual HBAs.

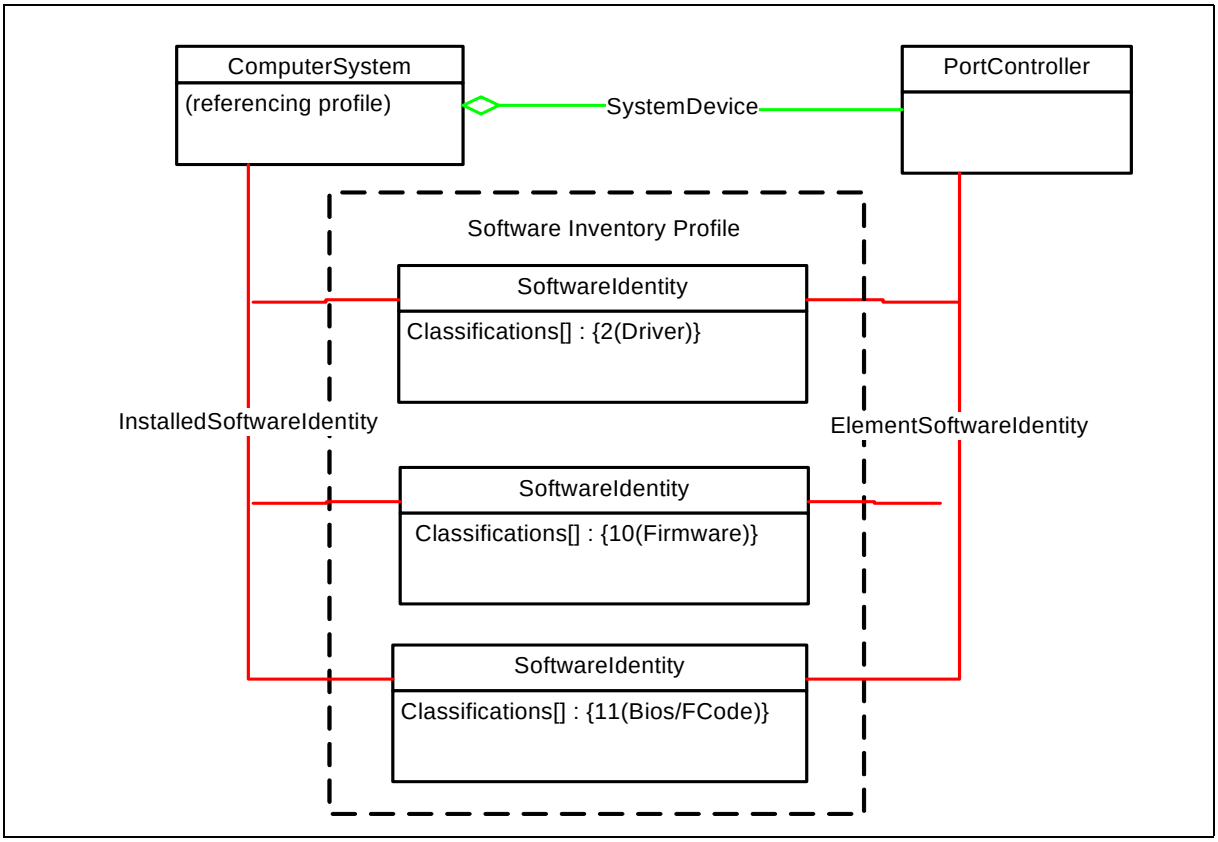

Figure 13 - Software Inventory Profile in Storage HBA

6.3.8 Software Update Profile

The implementation may optionally use the DMTF Software Update Profile to provide an interface for updating HBA software or firmware.

6.3.9 HBA Hot Swap Events

The implementation may optionally support asynchronous notification of HBA inserts and removals using the InstCreation and InstDeletion indications (see Table 25, “CIM Elements for Storage HBA,”).

6.3.10 Physical Asset Profile

The physical representation of the controller is optional. The Physical Asset Profile defines the set of classes and subclasses for describing the physical assets of a managed component. Most HBAs can be described as a physical card or chip on a motherboard. Therefore, at a minimum, the implementation shall include an instance of a subclass of PhysicalComponent or PhysicalPackage. For example, the CIM_Card class is a subclass of CIM_PhysicalPackage. The implementation may choose CIM_Card to represent a physical RAID controller card. In this case, the instance of CIM_Card is associated to the top-level controller CIM_PortController via the CIM_Realizes association.

Other physical classes such as Slots are optional, provided the sub-elements are consistent with the Physical Asset Profile. Figure 14 illustrates an HBA Card with Physical Classes.

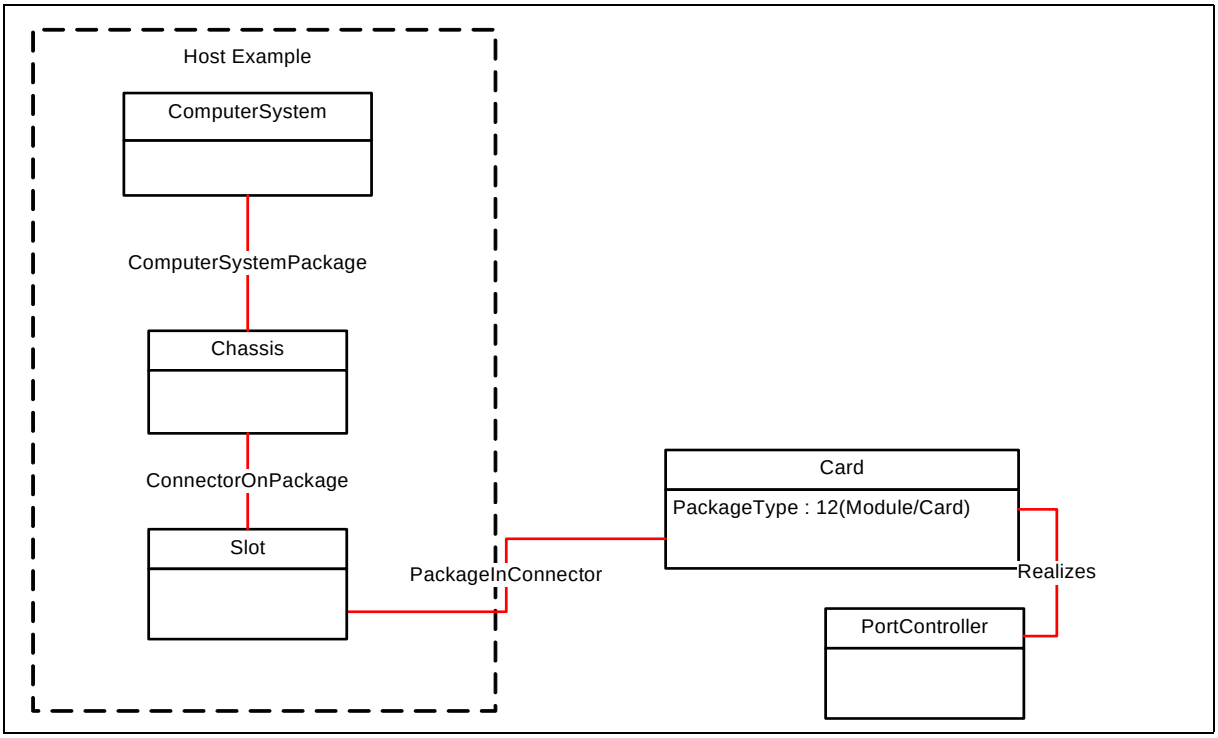

Figure 14 - HBA Card with Physical Classes

6.3.11 Modeling Attached Disk, Tape, and Optical Drives

Attached devices may be modeled using the optional remote elements model in 13.3.1 Remote Device Models.

6.3.12 Virtual System Considerations

In the “usual” configuration without any system virtualization, the instrumentation is directly working with physical HBAs which are not shared with guest virtual machines (VMs). In this case, PhysicalPackage (or a subclass) is mandatory as described in 6.3.3.1.

In a virtual system configuration, the requirements differ for the hypervisor and guest VMs.

- In a hypervisor context, PhysicalPackage (or a subclass) is mandatory as described in 6.3.3.1
- In a guest VM, PhysicalPackage shall not be instantiated.

A future version of this profile will reference emerging profiles that define how virtual HBAs are allocated and mapped to physical HBAs.

6.3.13 Fibre Channel HBAs

6.3.13.1 General FC HBA Considerations

The FC Initiator Ports Profile specialization of Generic Initiator Ports Profile is mandatory.

6.3.13.2 Physical FC HBA Considerations

In this version of the standard, implementers should use the FC HBA Profile to model physical FC HBAs, except for those used in a hypervisor context. Physical FC HBAs in a hypervisor should be modeled using this profile.

EXPERIMENTAL

6.3.14 FC HBA Diagnostics Profile

The FC HBA Diagnostics Profile is a DMTF profile that supports diagnostic tests on fibre channel HBAs. Support for the profile would only occur if the Storage HBA supports fibre channel ports and is optional in any case. The diagnostic tests supported include an Echo and Ping test for testing access to storage devices, as well as tests on the HBA itself.

When the FC HBA Diagnostics Profile is supported, there will be an instance of CIM_FCHBADiagnosticTest associated (via CIM_HostedService) to the Base Server system. That diagnostic test instance will have a CIM_AvailableDiagnosticService association to the Storage HBA PortController.

EXPERIMENTAL

6.4 Methods of the Profile

This section details the requirements for supporting intrinsic operations and extrinsic methods for the CIM elements defined by this profile.

6.4.1 Profile Conventions for Operations

Table 23 lists operations and requirements for CIM_PortController.

Table 23 - CIM_PortController

Operation	Requirement	Message
GetInstance	Mandatory	None
ModifyInstance	Unspecified	None
Associators	Unspecified	None
AssociatorNames	Unspecified	None
References	Unspecified	None
ReferenceNames	Unspecified	None
EnumerateInstances	Unspecified	None
EnumerateInstanceNames	Unspecified	None

Table 24 lists operations and requirements for CIM_SystemDevice.

Table 24 - CIM_SystemDevice

Operation	Requirement	Message
GetInstance	Mandatory	None
ModifyInstance	Unspecified	None

Table 24 - CIM_SystemDevice

Operation	Requirement	Message
Associators	Unspecified	None
AssociatorNames	Unspecified	None
References	Unspecified	None
ReferenceNames	Unspecified	None
EnumerateInstances	Unspecified	None
EnumerateInstanceNames	Unspecified	None

6.5 Use Cases

Not defined in this standard

6.6 CIM Elements

Table 25 describes the CIM elements for Storage HBA.

Table 25 - CIM Elements for Storage HBA

Element Name	Requirement	Description
6.6.1 CIM_ControlledBy	Mandatory	Associates PortController and LogicalPort.
6.6.2 CIM_PortController	Mandatory	Represents the logical aspects of the HBA. Associated to RegisteredProfile.
6.6.3 CIM_Product	Mandatory	
6.6.4 CIM_ProductElementComponent	Mandatory	
6.6.5 CIM_Realizes	Optional	
6.6.6 CIM_SystemDevice	Mandatory	Associates System to PortController.
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_PortController	Optional	PortController (HBA) Creation. See6.3.9 HBA Hot Swap Events.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_PortController	Optional	PortController (HBA) Removal. See6.3.9 HBA Hot Swap Events.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity='SNIA' and MessageID='Host1'	Optional	Controller firmware is older than required.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity='SNIA' and MessageID='Host2'	Optional	Controller firmware is older than recommended.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity='SNIA' and MessageID='Host3'	Optional	Controller is okay.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity='SNIA' and MessageID='Host4'	Optional	Controller is not okay.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity='SNIA' and MessageID='Host5'	Optional	Bus rescan complete.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity='SNIA' and MessageID='Host6'	Optional	Disk initialize failed.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity='SNIA' and MessageID='Core11'	Optional	Drive not responding.

6.6.1 CIM_ControlledBy

Associates PortController and LogicalPort.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 26 describes class CIM_ControlledBy.

Table 26 - SMI Referenced Properties/Methods for CIM_ControlledBy

Properties	Flags	Requirement	Description & Notes
DeviceNumber		Optional	The number of the port, relative to the PortController. This is sometimes referred to as the bus number.
Dependent		Mandatory	Reference to LogicalPort.
Antecedent		Mandatory	Reference to PortController.

6.6.2 CIM_PortController

Represents the logical aspects of the HBA.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Shall be associated to RegisteredProfile using ElementConformsToProfile association. The RegisteredProfile instance shall have RegisteredName set to 'Storage HBA', RegisteredOrganization set to 'SNIA', and RegisteredVersion set to '1.6.1'.

Table 27 describes class CIM_PortController.

Table 27 - SMI Referenced Properties/Methods for CIM_PortController

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	

6.6.3 CIM_Product

Requirement: Mandatory

Table 28 describes class CIM_Product.

Table 28 - SMI Referenced Properties/Methods for CIM_Product

Properties	Flags	Requirement	Description & Notes
ElementName		Mandatory	
Name		Mandatory	
IdentifyingNumber		Mandatory	
Vendor		Mandatory	
Version		Mandatory	

6.6.4 CIM_ProductElementComponent

Requirement: Mandatory

Table 29 describes class CIM_ProductElementComponent.

Table 29 - SMI Referenced Properties/Methods for CIM_ProductElementComponent

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	
PartComponent		Mandatory	

6.6.5 CIM_Realizes

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 30 describes class CIM_Realizes.

Table 30 - SMI Referenced Properties/Methods for CIM_Realizes

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

6.6.6 CIM_SystemDevice

Associates System to PortController.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 31 describes class CIM_SystemDevice.

Table 31 - SMI Referenced Properties/Methods for CIM_SystemDevice

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	
PartComponent		Mandatory	

EXPERIMENTAL

EXPERIMENTAL

7 Host Discovered Resources Profile

7.1 Description

7.1.1 Synopsis

Profile Name: Host Discovered Resources (Component Profile)

Version: 1.7.0

Organization: SNIA

Central Class: LogicalDevice

Scoping Class: Base Server ComputerSystem

Related Profiles: Not defined in this standard.

7.1.2 Overview

The Host Discovered Resources Profile allows a client application to discover the storage hardware resources attached to a host system, the logical storage resources available through the OS, and the relationship between these hardware and logical resources. The hardware resources include host adapters and storage devices. The logical resources include the OS special files that represent storage devices. In some cases, there is a one-to-one relationship between the logical and physical device. But multipath and disk partitioning introduce resource fan-in and fan-out that are also modeled in this profile.

Figure 15 depicts the relationship between the Host Discovered Resources Profile and these other profiles. The areas with the shaded background are covered by the Host Discovered Resources Profile – including partitioned and multipath storage.

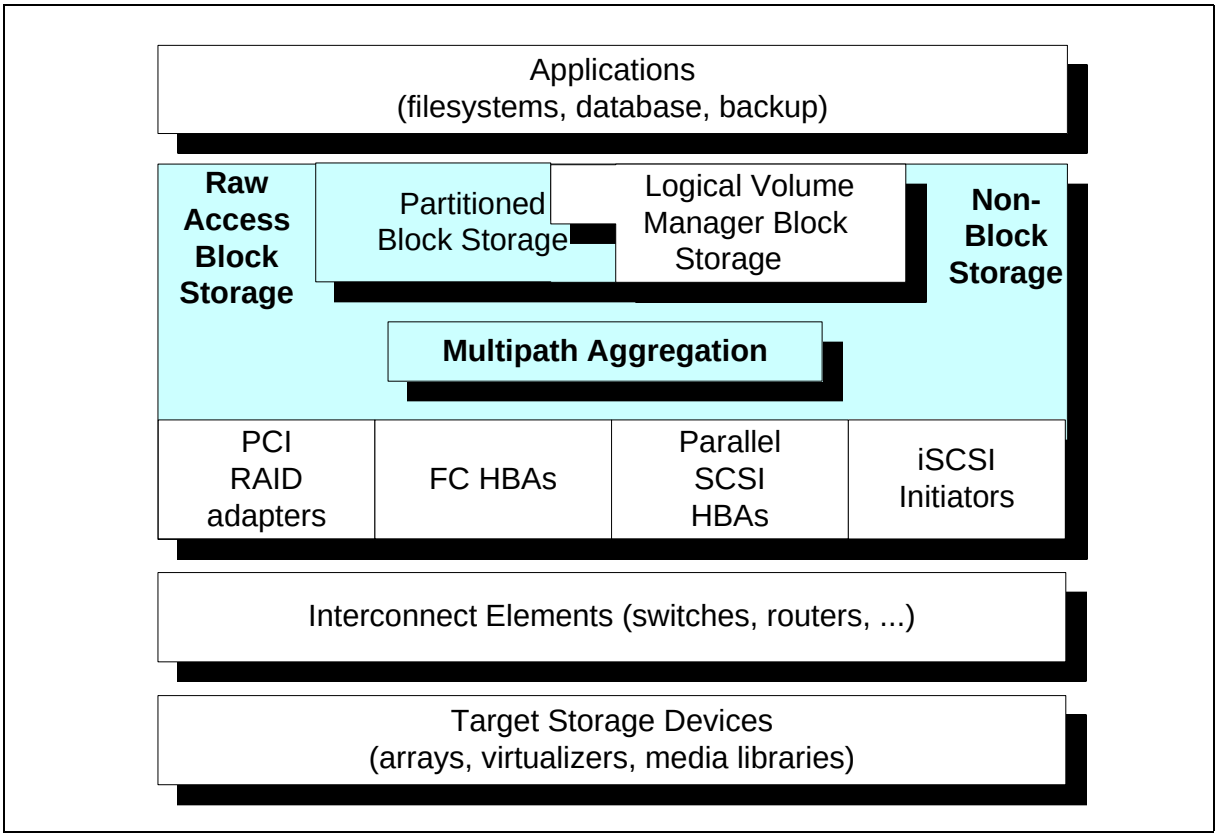

Figure 15 - Host Discovered Resources Block Diagram

Applications and Logical Volume Manager are consumers of Host Discovered Resources. The diagram depicts how an application can use Logical Volume Manager resources or use Host Discovered Resources directly. For example, a server may have some filesystems using LVM volumes and some filesystems using OS volumes.

The blocks at the bottom of the diagram represent resources (HBAs and target devices) for which the Host Discovered Resources Profile provides a host view. Note that interconnect elements between the HBAs and target devices are not part of the Host Discovered Resources Profile.

The Host Discovered Resources Profile provides a minimal amount of information about the discovered hardware resources; this includes the connectivity and correlatable IDs. The Host Discovered Resources Profile does not act as the canonical profile for any particular hardware resource; even host-resident elements like FC HBAs, iSCSI initiators, and Logical Volume Managers have separate profiles. The correlatable IDs exposed by The Host Discovered Resources Profile allow an application to associate host-discovered resources with resources from these other profiles.

For example, an array profile can describe the redundancy characteristics and performance statistics of a RAID volume. But the array profile will use a SCSI logical unit identifier as the volume's name. By combining information from the array and host-discovered resources profiles, a client can display the host special file name(s) associated with that volume. This additional name information can help the administrator (or client software) determine which applications are associated with volumes.

All references to ComputerSystem in the Host Discovered Resources Profile implies a single instance for a customer server or storage system as defined in the Base Server Profile. See Annex A: (Informative) Host Profile Deployment Guidelines, 1.6.0 Rev 3 for information on the use of host profiles with Base Server profile.

7.1.3 Host Disk Extent Class Name Conventions

The Host Discovered Resources Profile uses several different CIM classes to represent disk extents. **LogicalDisk** models an extent exposed by the OS to applications such as filesystems, databases, or logical volume managers. **GenericDiskPartition** represents a partition or slice of a disk as supported directly by the OS. **StorageVolume** represents disks or virtual volumes exported from disk arrays and virtualizers in the array or virtualizer profiles. **StorageExtent** represents disk extents that do fit these other classes; these will be intermediate extents that are neither consumed volumes nor exported logical disks.

Note that Logical Volume Managers are described in a separate profile. Logical Volume Managers may also expose partitions, but these are independent of partitions integrated into some OSes. The Host Discovered Resources Profile just addresses OS partitions.

To make it easier for clients of this profile, all consumable storage exported by this profile are modeled as instances of LogicalDisk.

The functionality of host resources discovery is broken into three areas:

- Disk partition discovery and management - see 4 Disk Partition Profile.
- Multipath Management - see 10 SCSI Multipath Management Profile.
- DIscovery of Hardware Resouces - see 7.1.4.

7.1.4 Discovered Hardware Resources

This profile presents a view of discovered resources with a common model based on the various command sets used in I/O devices - SCSI, ATA, or SB. Ports are modeled as instances of SCSIProtocolEndpoint, ATAProtocolEndpoint, or SBProtocolEndpoint - representing the use of the command set(s) used with the port.

The Host Discovered Resource Profile could be implemented using standard APIs (such as an HBA, SCSI, or iSCSI API) to create a generic model of the host-storage controllers and storage attached to those controllers. The model includes elements also exposed by HBA and storage agents; the details are included in these other profiles. A client uses correlatable IDs to equate objects from different agents.

The correlatable IDs for logical units (LogicalDisk, StorageExtent, TapeDrive) are the identifiers assigned by the hosting operating system (see *Storage Management Technical Specification, Part 2 Common Architecture, 1.7.0 Rev 5*, 7 Correlatable and Durable Names, 7.6.6 "Operating System Device Names" for the name requirements for OS names of disk logical units). An implementation of this profile shall also provide the correlatable names associated with the underlying devices. The requirements specified in *Storage Management Technical Specification, Part 2 Common Architecture, 1.7.0 Rev 5*, 7 Correlatable and Durable Names, 7.6.2 "Standard Formats for Logical Unit Names" applies, but instead of using the Name and NameNamespace properties, the information is contained in the OtherIdentifyingInfo and IdentifyingDescriptions array properties. The valid strings for IdentifyingDescriptions are exactly those described for NameNamespace in *Storage Management Technical Specification, Part 2 Common Architecture, 1.7.0 Rev 5*, 7 Correlatable and Durable Names, 7.6.2 "Standard Formats for Logical Unit Names".

This profile is restricted to discovery of I/O devices and does not include remote filesystems. The SCSI, ATA, and SB models are discussed separately.

Model for SCSI Protocol Resources

The SCSI protocol is used in several transports - Fibre Channel, iSCSI, Parallel SCSI (SPI), and Serial Attached SCSI (SAS). SCSI Protocol includes initiator and target ports, and logical units (RAID volumes, tape drives) in a many-to-many-to-many relationship - in other words, an initiator port may connect to many target ports (and vice versa), and each target device many have many logical units connected to initiator and target ports.

Figure 16 is a class diagram for Host Discovered Resources. SCSIProtocolEndpoint represents the SCSI logical port, either initiator or target. The transports type (e.g., FC, iSCSI) is specified in SCSIProtocolEndpoint ConnectionType property.

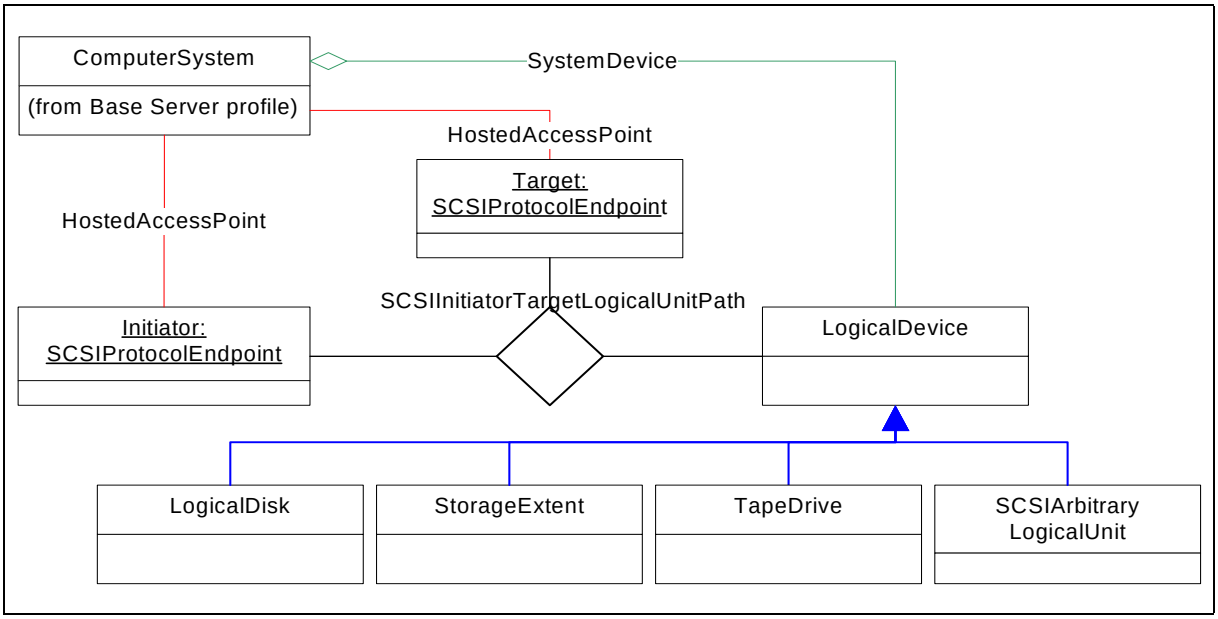

Figure 16 - Host Discovered Resources Class Diagram

The initiator ProtocolEndpoint and each target ProtocolEndpoint and LogicalDevice are associated by SCSIInitiatorTargetLogicalUnitPath.

Consider a few concrete cases. Figure 17 depicts the model for a single parallel SCSI disk. In general, Host APIs cannot differentiate a “real” disk from a virtual disk as exposed by a RAID controller, so the StorageExtent subclass of LogicalDevice is used.

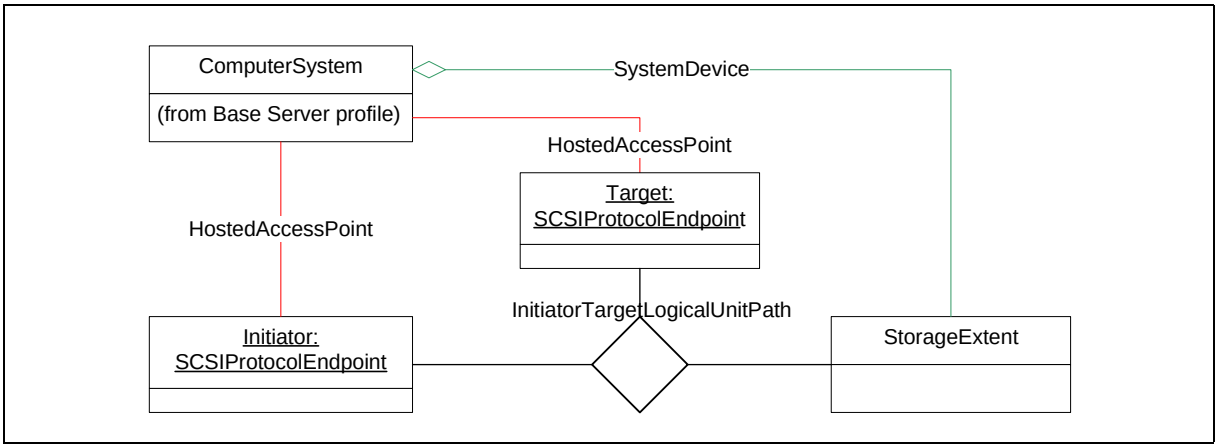

Figure 17 - Single SPI Disk Model

Figure 18 depicts a Fibre Channel RAID controller exposing three virtual disks to a single host/initiator port. There is a single initiator and target that share access to three StorageExtent instances.

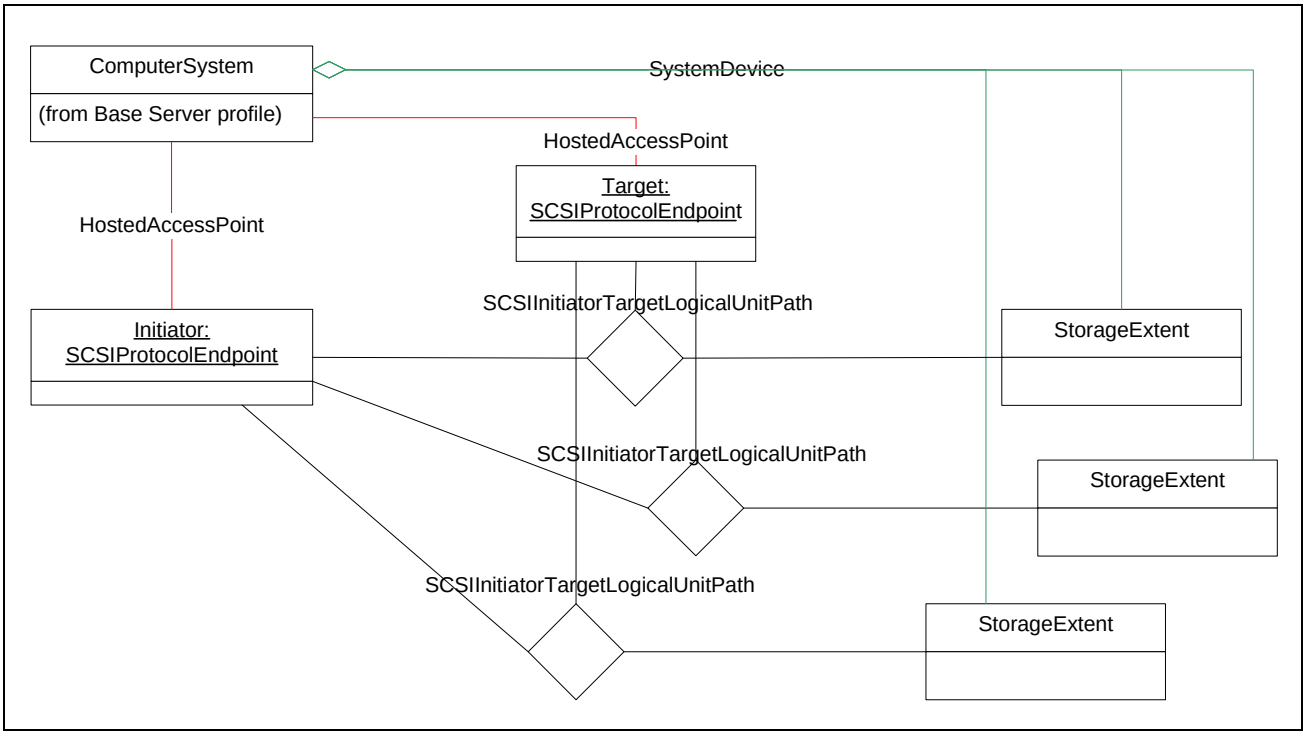

Figure 18 - Three FCP Logical Unit Instance Diagram

The Multipath Profile describes more complicated multipath configurations. See Figure 38.

Model for ATA Protocol Resources

The model for ATA, shown in Figure 19, uses a ConnectivityCollection of ATAProtocolEndpoints.

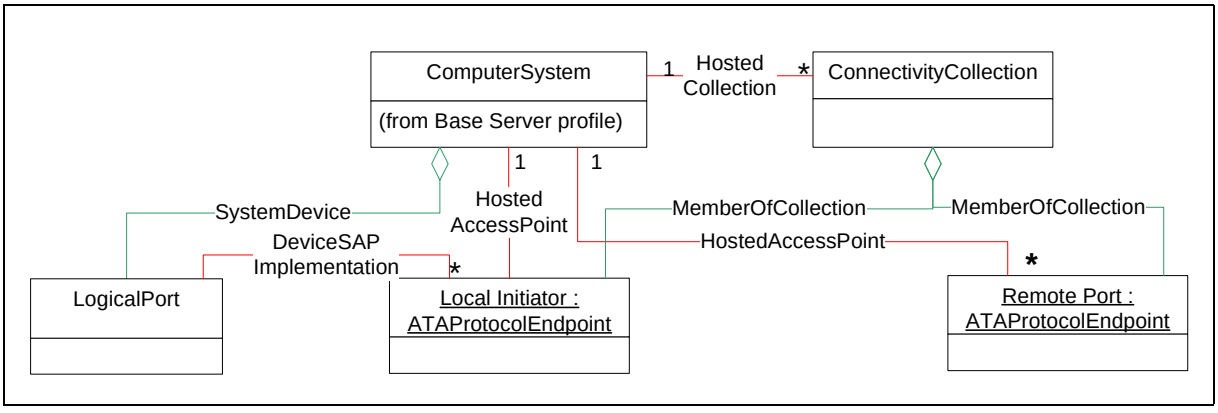

Figure 19 - ATA Discovered Resource Model

Model for SB Protocol Resources

The SB protocol is used in the Fibre Channel FC-SB-x transport. SB protocol includes initiator ("channel image") and target ports, and a logical units (ECKD volumes, tape drives) in a many-to-many-to-many relationship - in other words, an initiator port may connect to many target ports (and vice versa), and each target device may have many logical units connected to initiator and target ports. Figure 20 provides a general controller/device SB model. LogicalDevice subclasses model different types of SB logical unit,

e.g., TapeDrive. SBProtocolEndpoint represents the SB logical port, either initiator or target. The transports type (e.g., FC) is specified in SBProtocolEndpoint ConnectionType property. The initiator ProtocolEndpoint and each target ProtocolEndpoint and LogicalDevice are associated by SBInitiatorTargetLogicalUnitPath. Each SBLogicalDevice is contained within a one SBCUImage.

Figure 20 depicts SB Host Discovered Resources.

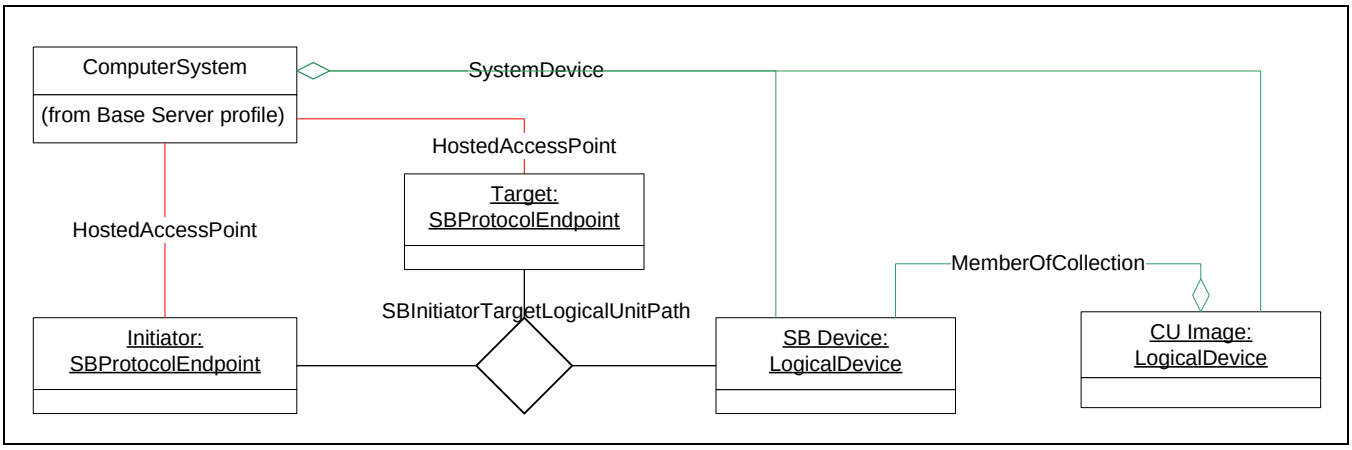

Figure 20 - SB Host Discovered Resources

Associating Hardware and OS Devices

There are two variations for disks and virtual disks - configurations with or without disk partitions.

1) With no partitions, each discovered (virtual) disk is modeled as LogicalDisk

2) With disk partitions, each partition exposed to an application or LVM is modeled as LogicalDisk. Any disk (or intermediate partition) that contains partitions is modeled as StorageExtent. DiskPartition instances are modeled between the StorageExtents and LogicalDisks. For more details, see 4 Disk Partition Profile. The requirement for disk partitions is reflected by the presence of DiskPartition-ConfigurationCapabilities.

Tape drive configurations are similar to case 1 above, with TapeDrive rather than LogicalDisk.

7.2 Health and Fault Management Considerations

Not defined in this standard

7.3 Cascading Considerations

Not defined in this standard

7.4 Extrinsic Methods of the Profile

StorageConfigurationService.ScsiScan

This method requests that the system rescan SCSI devices for changes in their configuration. If called on a general-purpose host, the changes are reflected in the list of devices available to applications (for example, the UNIX 'device tree').

This operation can be disruptive; optional parameters allow the caller to limit the scan to a single or set of SCSI device elements. All parameters are optional; if parameters other than Job are passed in as null, a full scan is invoked. If the caller specifies a connection type, the scan is limited to that connection type.

Job - a reference to a Job

ConnectionType - The type of connection (transport, such as FC or iSCSI), constrains the scan to initiator ports of this type. Only used if the Initiators parameter is null.

OtherConnectionType - The connection type if the ConnectionType parameter is Other.

Initiators - A list of references to initiators. Scanning will be limited to SCSI targets attached to these initiators. If this parameter is null and connection is specified, all initiators of that connection type are scanned. If this parameter and ConnectionType are null, all targets on all system initiators are probed.

Targets - A list of names or numbers for targets. These should be formatted to match the appropriate connection type. For example, PortWWNs would be specified for Fibre Channel targets.

LogicalUnits - A list of SCSI logical unit numbers representing logical units hosted on the targets specified in the Targets argument.

ScsiScan() support is optional. Support for ScsiScan() can be determined based on the inclusion of “SCSI Scan” in the SupportedAsynchronousActions array in StorageConfigurationCapabilities.

7.5 Use Cases

Not defined in this standard

7.6 CIM Elements

Table 32 describes the CIM elements for Host Discovered Resources.

Table 32 - CIM Elements for Host Discovered Resources

Element Name	Requirement	Description
7.6.1 CIM_ATAInitiatorTargetLogicalUnitPath	Optional	Associates initiator and target ATAProtocolEndpoints to a logical unit.
7.6.2 CIM_ATAProtocolEndpoint	Optional	
7.6.3 CIM_ElementConformsToProfile (LogicalDevice to Host Discovered Resources RegisteredProfile)	Mandatory	Ties the LogicalDevice to the registered profile for Host Discovered Resources.
7.6.4 CIM_HostedAccessPoint	Mandatory	This association links all ProtocolEndpoints to the Base Server profile computer system.
7.6.5 CIM_LogicalDevice (LogicalDevice)	Mandatory	Represents a block logical unit that is exposed to applications such as file systems without being partitioned.
7.6.6 CIM_LogicalDisk (LogicalDevice)	Optional	Represents a block logical unit that is exposed to applications such as file systems without being partitioned.
7.6.7 CIM_SCSIArbitraryLogicalUnit (LogicalDevice)	Optional	A SCSI Logical Unit that exists only for management.
7.6.8 CIM_SCSIInitiatorTargetLogicalUnitPath	Optional	Associates initiator and target SCSIProtocolEndpoints to a logical unit.
7.6.9 CIM_SCSIProtocolEndpoint	Optional	
7.6.10 CIM_StorageExtent (LogicalDevice)	Optional	Represents a block logical unit in the host that is partitioned before being exposed to applications.
7.6.11 CIM_SystemDevice	Mandatory	This association links LogicalDevices to the Base Server profile computer system.
7.6.12 CIM_TapeDrive (LogicalDevice)	Optional	Represents a tape drive logical unit in the host.
7.6.13 CIM_SBInitiatorTargetLogicalUnitPath	Optional	Associates initiator and target SBProtocolEndpoints to a logical unit.

Table 32 - CIM Elements for Host Discovered Resources

Element Name	Requirement	Description
7.6.14 CIM_SBProtocolEndpoint	Optional	
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_SCSIInitiatorTargetLogicalUnitPath	Mandatory	Path creation.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_SCSIInitiatorTargetLogicalUnitPath	Mandatory	Path deletion.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_SCSIInitiatorTargetLogicalUnitPath AND SourceInstance.CIM_SCSIInitiatorTargetLogicalUnitPath::State <> PreviousInstance.CIM_SCSIInitiatorTargetLogicalUnitPath::State	Mandatory	CQL -Path State change.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_SCSIInitiatorTargetLogicalUnitPath AND SourceInstance.CIM_SCSIInitiatorTargetLogicalUnitPath::AdministrativeWeight <> PreviousInstance.CIM_SCSIInitiatorTargetLogicalUnitPath::AdministrativeWeight	Mandatory	CQL -Path AdminsitrativeWeight change.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_SCSIInitiatorTargetLogicalUnitPath AND SourceInstance.CIM_SCSIInitiatorTargetLogicalUnitPath::AdministrativeOverride <> PreviousInstance.CIM_SCSIInitiatorTargetLogicalUnitPath::AdministrativeOverride	Mandatory	CQL -Path AdministrativeOverride change.

7.6.1 CIM_ATAInitiatorTargetLogicalUnitPath

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 33 describes class CIM_ATAInitiatorTargetLogicalUnitPath.

Table 33 - SMI Referenced Properties/Methods for CIM_ATAInitiatorTargetLogicalUnitPath

Properties	Flags	Requirement	Description & Notes
State		Mandatory	
LogicalUnit		Mandatory	A reference to a LogicalDevice.
Initiator		Mandatory	A reference to the initiator CIM_ATAProtocolEndpoint.
Target		Mandatory	A reference to the target CIM_ATAProtocolEndpoint.

7.6.2 CIM_ATAProtocolEndpoint

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 34 describes class CIM_ATAProtocolEndpoint.

Table 34 - SMI Referenced Properties/Methods for CIM_ATAProtocolEndpoint

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
Name	C	Mandatory	
ProtocolIFType		Mandatory	
ConnectionType		Mandatory	
Role		Mandatory	

7.6.3 CIM_ElementConformsToProfile (LogicalDevice to Host Discovered Resources Registered-Profile)

The CIM_ElementConformsToProfile ties LogicalDevice to the registered profile for Host Discovered Resources.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 35 describes class CIM_ElementConformsToProfile (LogicalDevice to Host Discovered Resources RegisteredProfile).

Table 35 - SMI Referenced Properties/Methods for CIM_ElementConformsToProfile (LogicalDevice to Host Discovered Resources RegisteredProfile)

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	A LogicalDevice instance that represents the Host Discovered Resources.
ConformantStandard		Mandatory	RegisteredProfile instance describing the Host Discovered Resources profile.

7.6.4 CIM_HostedAccessPoint

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 36 describes class CIM_HostedAccessPoint.

Table 36 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

7.6.5 CIM_LogicalDevice (LogicalDevice)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 37 describes class CIM_LogicalDevice (LogicalDevice).

Table 37 - SMI Referenced Properties/Methods for CIM_LogicalDevice (LogicalDevice)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	Opaque identifer.
ElementName		Mandatory	User-friendly name.
Name	C	Mandatory	OS device name.
NameFormat	C	Mandatory	Shall be 12 (OS Device Name).
NameNamespace	C	Mandatory	Shall be 8 (OS Device NameSpace).
OtherIdentifyingInfo	C	Mandatory	The correlatable ID of the underlying logical unit.
IdentifyingDescriptions	C	Mandatory	
OperationalStatus		Mandatory	

7.6.6 CIM_LogicalDisk (LogicalDevice)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 38 describes class CIM_LogicalDisk (LogicalDevice).

Table 38 - SMI Referenced Properties/Methods for CIM_LogicalDisk (LogicalDevice)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	

Table 38 - SMI Referenced Properties/Methods for CIM_LogicalDisk (LogicalDevice)

Properties	Flags	Requirement	Description & Notes
CreationClassName		Mandatory	
DeviceID		Mandatory	Opaque identifer.
ElementName		Mandatory	User-friendly name.
Name	C	Mandatory	OS device name.
OtherIdentifyingInfo	C	Mandatory	The correlatable ID of the underlying logical unit.
IdentifyingDescriptions	C	Mandatory	
OperationalStatus		Mandatory	

7.6.7 CIM_SCSIArbitraryLogicalUnit (LogicalDevice)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 39 describes class CIM_SCSIArbitraryLogicalUnit (LogicalDevice).

Table 39 - SMI Referenced Properties/Methods for CIM_SCSIArbitraryLogicalUnit (LogicalDevice)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	Opaque identifer.
ElementName		Mandatory	User-friendly name.
Name	C	Mandatory	OS device name.
OtherIdentifyingInfo	C	Mandatory	The correlatable ID of the underlying logical unit.
IdentifyingDescriptions	C	Mandatory	
OperationalStatus		Mandatory	

7.6.8 CIM_SCSIInitiatorTargetLogicalUnitPath

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 40 describes class CIM_SCSIInitiatorTargetLogicalUnitPath.

Table 40 - SMI Referenced Properties/Methods for CIM_SCSIInitiatorTargetLogicalUnitPath

Properties	Flags	Requirement	Description & Notes
OSDeviceName		Optional	
AdministrativeWeight	M	Mandatory	
State		Mandatory	
AdministrativeOverride		Mandatory	
LogicalUnit		Mandatory	A reference to a LogicalDevice.
Initiator		Mandatory	A reference to the initiator CIM_SCSIProtocolEndpoint.
Target		Mandatory	A reference to the target CIM_SCSIProtocolEndpoint.

7.6.9 CIM_SCSIProtocolEndpoint

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 41 describes class CIM_SCSIProtocolEndpoint.

Table 41 - SMI Referenced Properties/Methods for CIM_SCSIProtocolEndpoint

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
Name	C	Mandatory	
ProtocolIFType		Mandatory	Shall be 1 (Other).
ConnectionType		Mandatory	
Role		Mandatory	

7.6.10 CIM_StorageExtent (LogicalDevice)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 42 describes class CIM_StorageExtent (LogicalDevice).

Table 42 - SMI Referenced Properties/Methods for CIM_StorageExtent (LogicalDevice)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	Opaque identifer.
ElementName		Mandatory	User-friendly name.
Name	C	Mandatory	OS device name.
NameFormat	C	Mandatory	Shall be 12 (OS Device Name).
NameNamespace	C	Mandatory	Shall be 8 (OS Device NameSpace).
OtherIdentifyingInfo	C	Mandatory	The correlatable ID of the underlying logical unit.
IdentifyingDescriptions	C	Mandatory	
OperationalStatus		Mandatory	

7.6.11 CIM_SystemDevice

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 43 describes class CIM_SystemDevice.

Table 43 - SMI Referenced Properties/Methods for CIM_SystemDevice

Properties	Flags	Requirement	Description & Notes
PartComponent		Mandatory	
GroupComponent		Mandatory	

7.6.12 CIM_TapeDrive (LogicalDevice)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 44 describes class CIM_TapeDrive (LogicalDevice).

Table 44 - SMI Referenced Properties/Methods for CIM_TapeDrive (LogicalDevice)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	

Table 44 - SMI Referenced Properties/Methods for CIM_TapeDrive (LogicalDevice)

Properties	Flags	Requirement	Description & Notes
CreationClassName		Mandatory	
DeviceID		Mandatory	Opaque identifer.
ElementName		Mandatory	User-friendly name.
Name	C	Mandatory	OS device name.
OtherIdentifyingInfo	C	Mandatory	The correlatable ID of the underlying logical unit.
IdentifyingDescriptions	C	Mandatory	
OperationalStatus		Mandatory	

7.6.13 CIM_SBInitiatorTargetLogicalUnitPath

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 45 describes class CIM_SBInitiatorTargetLogicalUnitPath.

Table 45 - SMI Referenced Properties/Methods for CIM_SBInitiatorTargetLogicalUnitPath

Properties	Flags	Requirement	Description & Notes
OSDeviceName		Optional	
UsePreferredPath		Optional	Boolean indicating whether preferred path processing is required.
PreferredPath		Optional	Boolean indicating whiether this is a preferred path.
PathGroupState		Optional	One of 'Unknown, 'Path grouping not supported','Reset', 'Grouped', 'Ungrouped'.
PathGroupMode		Optional	One of 'Unknown', 'None', 'Single path', 'Multipath' (SIngle path and multipath only v alid if PathGroupState is grouped.
PathGroupID		Optional	String containing the ID from the OS, only valid if PathGroupState is Grouped.
LogicalUnit		Mandatory	A reference to a LogicalDevice.
Initiator		Mandatory	A reference to the initiator CIM_SBProtocolEndpoint.
Target		Mandatory	A reference to the target CIM_SBProtocolEndpoint.

7.6.14 CIM_SBProtocolEndpoint

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 46 describes class CIM_SBProtocolEndpoint.

Table 46 - SMI Referenced Properties/Methods for CIM_SBProtocolEndpoint

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
Name	C	Mandatory	
ProtocolIFType		Mandatory	
ConnectionType		Mandatory	
Role		Mandatory	

EXPERIMENTAL

EXPERIMENTAL

8 Host Hardware RAID Controller Profile

8.1 Synopsis

Profile Name: Host Hardware RAID Controller (Component Profile)

Version: 1.5.0

Organization: SNIA

Central Class: ComputerSystem

Scoping Class: ComputerSystem

Related Profiles: Table 47 describes the related profiles for Host Hardware RAID Controller**.**

Table 47 - Related Profiles for Host Hardware RAID Controller

Profile Name	Organization	Version	Requirement	Description
Physical Asset	DMTF	1.0.0a	Mandatory	
Block Services	SNIA	1.7.0	Mandatory	
Disk Drive Lite	SNIA	1.7.0	Optional	
Software Inventory	SNIA	1.0.0	Mandatory	Experimental.
Software Update	DMTF	1.0.0	Optional	
Extent Composition	SNIA	1.6.0	Optional	
Disk Sparing	SNIA	1.7.0	Optional	
DA Target Ports	SNIA	1.4.0	Mandatory	
Erasure	SNIA	1.7.0	Optional	Experimental.
Copy Services	SNIA	1.5.0	Optional	Deprecated
Replication Services	SNIA	1.7.0	Optional	
Storage Enclosure	SNIA	1.3.0	Optional	Model for External Disk Enclosures.
Thin Provisioning	SNIA	1.6.0	Optional	
Block Storage Views	SNIA	1.7.0	Optional	
FC Initiator Ports	SNIA	1.7.0	Support for at least one is mandatory.	
SAS Initiator Ports	SNIA	1.7.0		
Indications	DMTF	1.2.2	Mandatory	See DSP1054, version 1.2.2

The Host Hardware RAID Controller Profile describes classes, properties, and other profiles necessary to manage a host-based RAID controller. The central class and the scoping Class is the ComputerSystem representing the controller.

8.2 Description

The Host Hardware RAID Controller Profile is intended to represent the manageable elements of a host-based RAID controller and the storage it controls. A RAID controller may manage disks contained within a server's internal drive cage or an external drive enclosure.

The Host Hardware RAID Controller Profile may be used to model manageability for software-based RAID included in drivers. However, manageability for volume manager-based RAID, running on a host operating system, is out of scope for the Host Hardware RAID Controller Profile.

Figure 21, “Host Hardware RAID Controller Package Diagram” shows the relationship between the Host Hardware RAID Profile and its key component profiles.

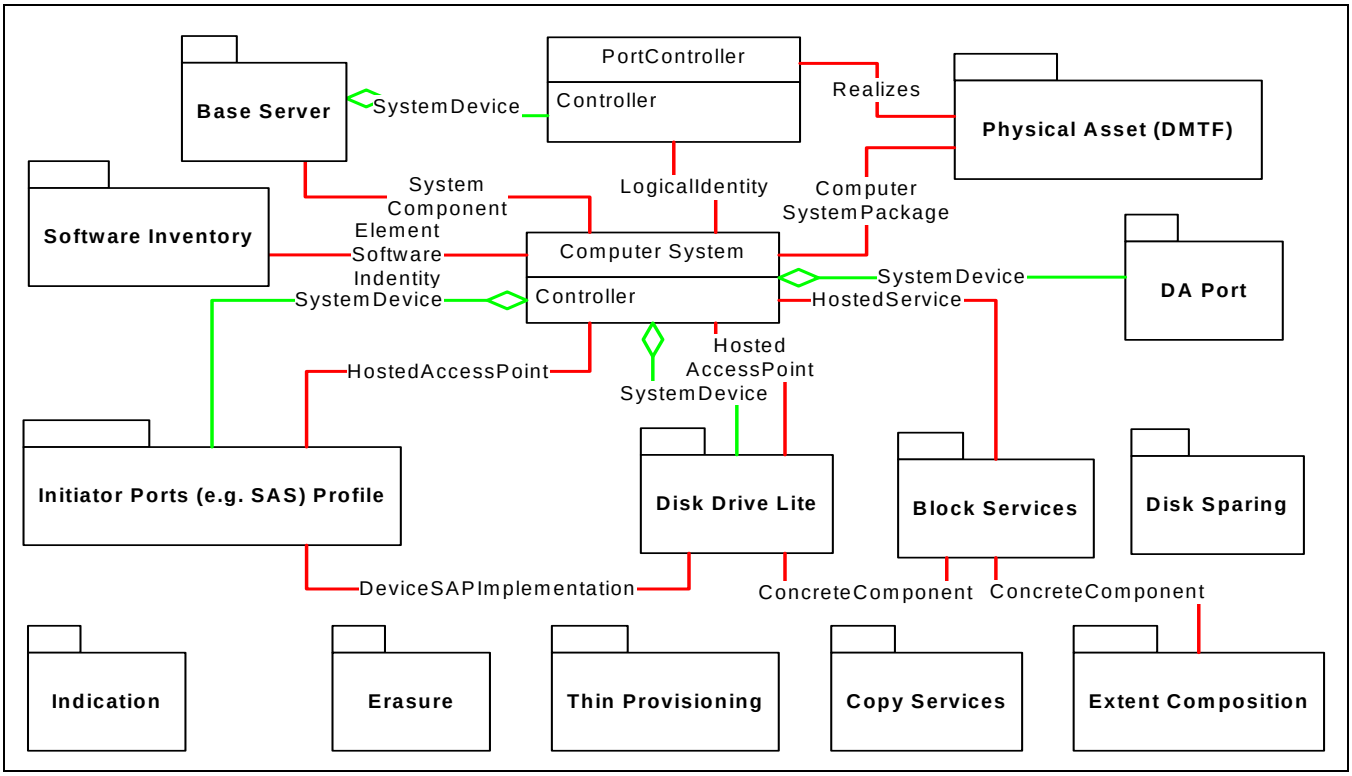

Figure 21 - Host Hardware RAID Controller Package Diagram

8.3 Implementation

8.3.1 Relationship to autonomous profiles

Although the Host Hardware RAID Controller Profile includes a ComputerSystem instance, it is a component profile, intended to be referenced by a separate profile modeling the host system that contains the RAID controller. In most cases, the Base Server Profile (see *Storage Management Technical Specification, Part 3 Common Profiles, 1.7.0 Rev 5* 30 Base Server Profile). will be the referencing profile. The ComputerSystems from the Host Hardware RAID Controller Profile and its referencing profile are associated using CIM_SystemComponent.

All references to ComputerSystem in the Host Hardware RAID Controller Profile impliy a single instance for a customer server or storage system as defined in the Base Server Profile. See Annex B (Informative) Host Profile Deployment Guidelines for information on the use of host profiles with Base Server profile.

8.3.2 CIM_PortController

The PortController class represents an instance of a RAID controller and controllers the backend port to the storage managed by this controller. An implementation of Host Hardware RAID Controller Profile shall associate PortController to the instance of ComputerSystem that represents the host in the referencing profile, using the SystemDevice association. Also, the PortController shall be associated to the ComputerSystem that represents the controller using the LogicalIdentity association. Finally, the PortController shall be associated to one or more instances of LogicalPort using the ControlledBy association.

8.3.3 CIM_ComputerSystem

In the Host Hardware RAID Controller Profile, the ComputerSystem Class within this profile represents the RAID controller itself.

The ComputerSystem that represents the RAID controller system acts as the principal class of the profile. Many of the other classes in the Host Hardware RAID Controller Profile that together act as a host-based RAID controller are scoped to the instance of ComputerSystem that represents the controller. This includes attached storage pools, volumes, drives, configuration services, logical ports, and physical cards. Any implementation shall instantiate an instance of ComputerSystem associated to PortController using the LogicalIdentity association. Also, the implementation shall include the value 30 (Host-Based RAID controller) in the Dedicated property".

Figure 22 illustrates the relationship between the Base Server ComputerSystem and the Host Hardware RAID Controller system, and the relationship of devices defined by this profile to the Host Hardware RAID Controller system.

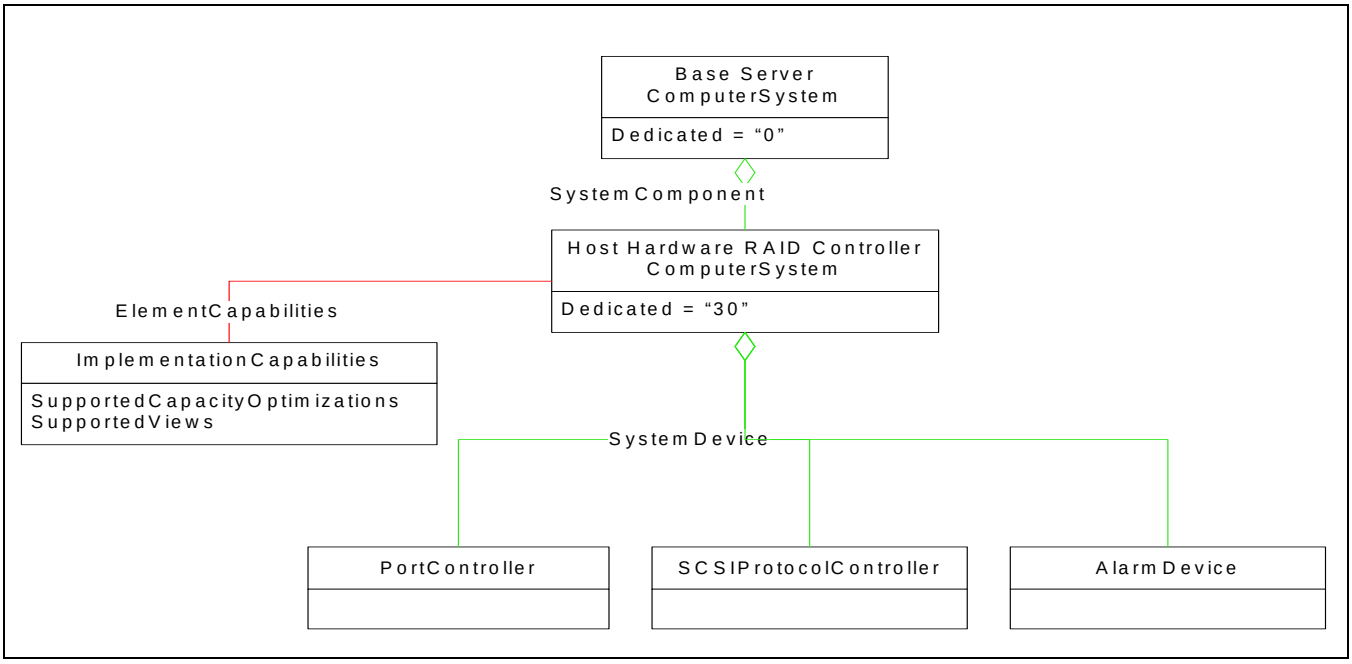

Figure 22 - Host Hardware RAID resources scoped to HHRC ComputerSystem

Figure 22 also illustrates the ImplementationCapabilities associated to the Host Hardware RAID Controller ComputerSystem. These capabilities identify the capacity optimization techniques and views supported by the implementation. Note that the Block Storage Views also calls for an instance of ViewCapabilities to identify the views supported. However, the supported views are greater than those supported in a Host Hardware RAID Controller configuration. The SupportedViews property of

ImplementationCapabilities limits the possible views supported to those that make sense in the context of a Host Hardware RAID Controller implementation.

8.3.4 CIM_AlarmDevice

Some Host-based RAID controllers have a visual or audible alarm to indicate when some event has occurred on the system, like a degraded RAID StorageVolume. CIM_AlarmDevice may be implemented to represent an alarm device on the RAID controller.

Determination: The implementation may create an instance of CIM_AlarmDevice that represents an alarm device associated to ComputerSystem that represents the controller using the SystemDevice association. Alarms may be either visual (i.e., LEDs on the controller) or audible, thus implementation shall set the value of the appropriate boolean property to TRUE in the CIM_AlarmDevice class. For example, if the alarm is an audible alarm, the implementation shall set the value of AudibleAlarm property to TRUE.

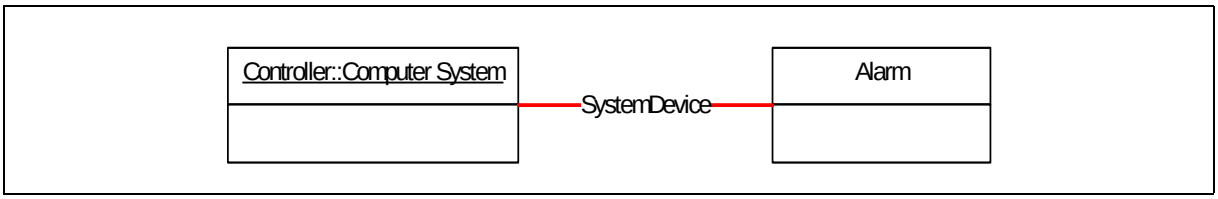

Figure 23 - Alarms in Host Hardware RAID Controller

8.3.5 Server Profile

For the Host Hardware RAID Controller Profile, the SNIA Server Profile is required. Any implementation shall follow the requirements of the SNIA Server Profile (see *Storage Management Technical Specification, Part 3 Common Profiles, 1.7.0 Rev 5* 35 Server Profile).

8.3.6 Profile Registration

For the Host Hardware RAID Controller Profile, the scoping class methodology of profile registration shall be used, as required by the server profile. However, an implementation may use the central class

methodology. The scoping class and central class of the Host Hardware RAID Controller profile is the instance of CIM_ComputerSystem that represents the controller.

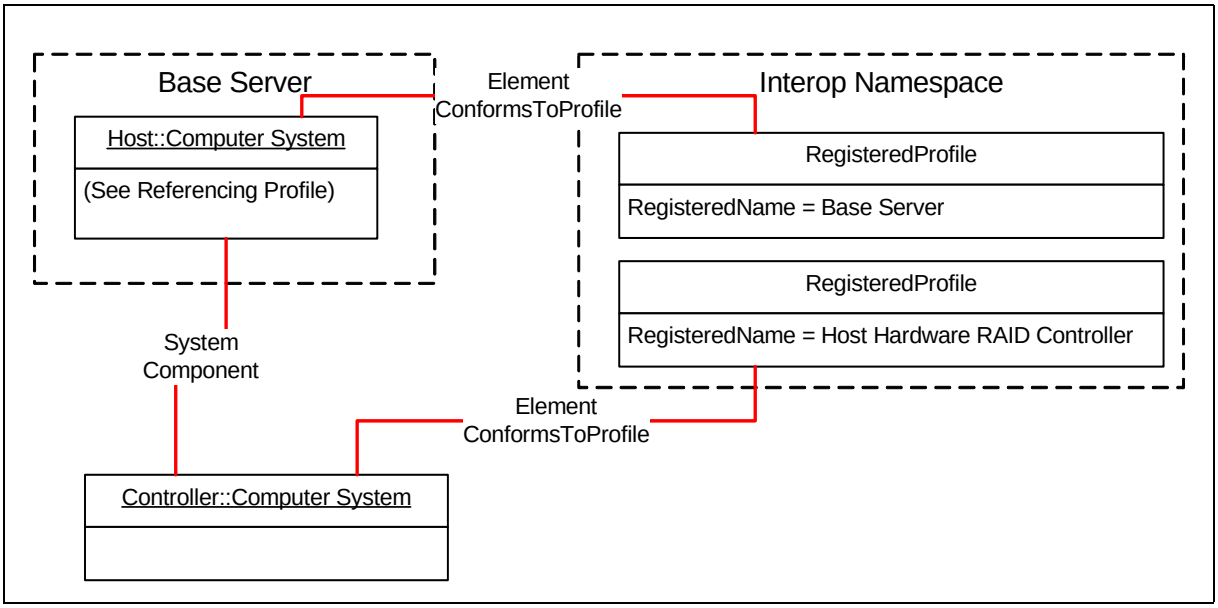

Figure 24 - Profile Registration with Host Hardware RAID Controller and Base Server Profiles

8.3.7 Profile Discovery and Advertisement

The Host Hardware RAID Controller Profile shall be advertised in SLP as “SNIA:Host Hardware RAID Controller”.

8.3.8 Physical Asset Profile

The physical representation of the controller is mandatory and realized by implementing the Physical Asset Profile. The Physical Asset Profile defines the set of classes and subclasses for describing the physical assets of a managed component. Most host-based RAID controllers can be described as a physical card or chip on a motherboard. Therefore, at a minimum, the implementation shall include an instance of a subclass of PhysicalComponent or PhysicalPackage. The PhysicalPackage or PhysicalComponent shall be associated (using Realizes) to the PortController and to the ComputerSystem representing the controller (using ComputerSystemPackage). For example, the CIM_Card class is a subclass of PhysicalPackage. The implementation may choose CIM_Card to represent a physical RAID controller card. In this case, the instance of CIM_Card is associated to the top-level controller CIM_PortController via the Realizes association.

For any instantiation of a subclass of PhysicalComponent or PhysicalPackage class (i.e., CIM_Card), the implementation shall populate the ElementName property with the name of the RAID controller model as described by the manufacturer.

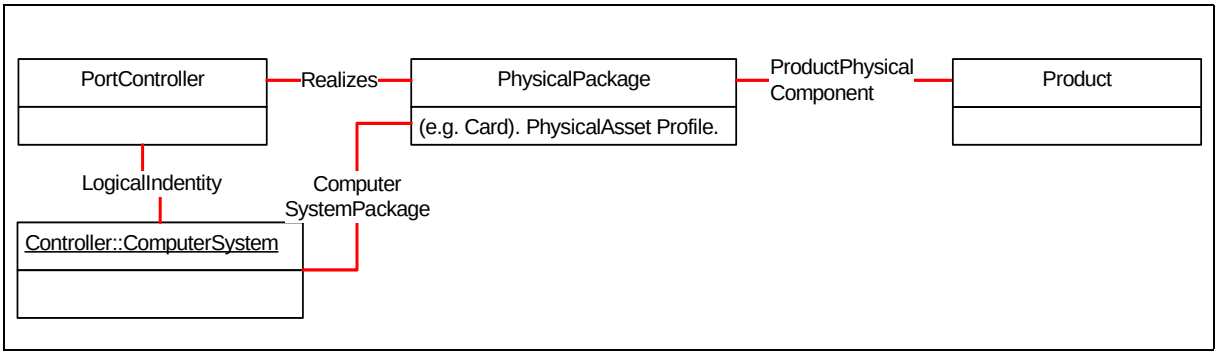

Figure 25 - Implementation of Physical Asset Profile

8.3.9 Storage Enclosure Profile

The Storage Enclosure Profile is an optional profile modeling disk enclosures dedicated to a Host Hardware RAID Controller. The associations defined in the Storage Enclosure Profile referencing ComputerSystem should reference the controller ComputerSystem defined in the Host Hardware RAID Controller Profile.

8.3.10 Implementation of Block Services Package

Figure 26 - Block Services Package in Host Hardware RAID Controller illustrates the Host Hardware RAID Controller use of Block Services. Note that most of the non-system objects must also be associated to the ComputerSystem; these associations are omitted to simplify the diagram.

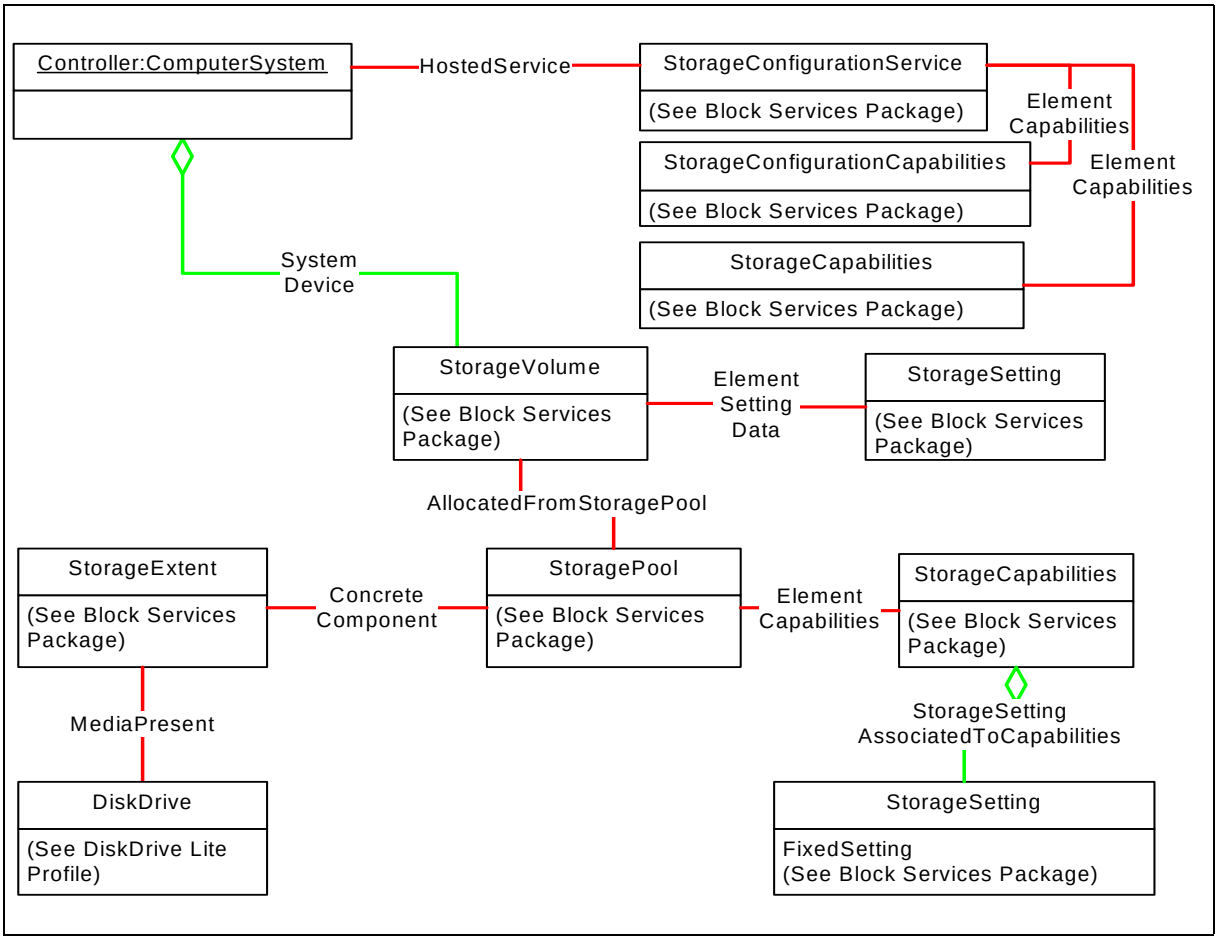

Figure 26 - Block Services Package in Host Hardware RAID Controller

8.3.10.1 Storage Pools

8.3.10.1.1 Primordial Storage Pools

As required by block services package, an implementation shall instantiate at least one primordial storage pool that represents the physical disk storage attached to the controller. However, some implementations that support disparate storage device types attached to the controller, such as a SAS/ SATA JBOD, may create multiple primordial storage pools based on the storage capabilities. For example, an implementation that supports SAS/SATA JBODs attached to the RAID controller may support one primordial pool for SAS disks and a second primordial pool for SATA disks.

Primordial StoragePools shall be associated to the principal instance ComputerSystem that represents the RAID controller using the HostedStoragePool association. Primordial storage pools may increase or decrease in size or go away but, at least one primordial storage pool shall always be instantiated, even if the RemainingManagedSpace property size is zero.

8.3.10.2 StorageConfigurationCapabilities

As required by block services, implementations shall instantiate a single instance of StorageConfigurationCapabilities associated to the StorageConfigurationService instance using the ElementCapabilities association. However, for the Host Hardware RAID Controller Profile, LogicalDisk creation and modification is not supported. Therefore the following properties shall be limited to a subset of the values defined in the DMTF MOF files for the StorageConfigurationCapabilities class:

- SupportedStorageElementTypes shall include the value 2 (StorageVolume).
- SupportedStoragePoolFeatures shall include the value 2 (InExtents) or 3 (Single InPool).
- SupportedStorageElementFeatures shall include the value 3 (StorageVolume Creation), 5 (StorageVolume Modification), 6 (Single InPool) or 10 (InElelments).

8.3.10.3 Storage Capabilities

For the initial state of Host Hardware RAID Controller Profile, implementations shall instantiate at least two instances of the StorageCapabilities class.

The first instantiation of StorageCapabilities is used to model the storage capabilities of the controller. The StorageCapabilities instance shall be associated to the StorageConfigurationService using the ElementCapabilities association. This instance allows the client to easily determine the storage capabilities of the controller. This capability is fixed and may change only when new functionality is added to the controller through a firmware change/update.

The second instantiation of StorageCapabilities is associated to a primordial StoragePool using the ElementCapabilities association. This instantiation of StorageCapabilities is required by Block Services Package and defines the range of redundancy capabilities of the primordial StoragePool.

8.3.10.4 Storage Settings

Implementations shall instantiate at least one StorageSetting class associated to the StorageCapabilities (associated to the primordial StoragePool) using the StorageSettingAssociatedToCapabilities association. The StorageSetting class further defines the redundancy capabilities of the primordial StoragePool. This is a “fixed” association, and shall not be modified by the client.

8.3.11 Implementation of DAPort and SCSIProtocolController

RAID controllers make the volumes exported by the controller appear as disks (or disk partitions) to the host operating system - which in turn makes them available to filesystems and databases running on the host. Typically the drivers supporting RAID controllers cause exported volumes to appear in the operating system’s “device tree” similarly to an external RAID array: SCSI logical units attached to SCSI target devices.

SCSIProtocolController represents the SCSI target device. StorageVolumes (from the Block Services package) represent the logical units and are associated to SCSIProtocolController with the ProtocolControllerForUnit association. This association has a DeviceNumber that holds the logical unit number for the StorageVolume. The DA Target Port Profile models the simulated SCSI port related to the SCSIProtocolController. The combination of these elements allows a application with existing support for SMI-S to additionally support the Host Hardware RAID profile with minimal changes.

The model for DA TArget Ports and SCSIProtocolCOntroller is shown in Figure 27, “DAPort Profile in Host Hardware Controller”.

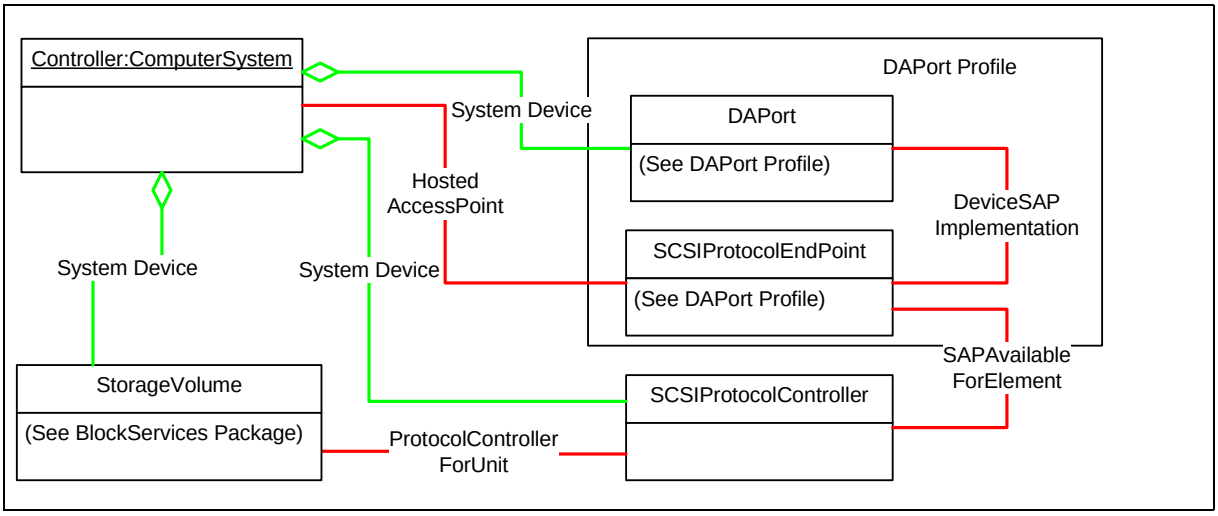

Figure 27 - DAPort Profile in Host Hardware Controller

8.3.12 Implementation of Software Inventory Profile

For the Host Hardware RAID Controller Profile, the SoftwareIdentity class from the Software Inventory Profile is required to model various software entities for a RAID controller. The implementation shall use the Software Inventory Profile to model the driver software for the RAID controller running on the Host Operating System and the firmware internal to the controller. If the RAID controller has a separate software entity for the BIOS from the firmware, the implementation may use the Software Inventory Profile to represent the BIOS.

To model the driver, firmware and BIOS software for the controller, the implementation shall instantiate an instance of SoftwareIdentity class associated to the top level ComputerSystem that represents the RAID controller, using the ElementSoftwareIdentity association.The SoftwareIdentity instances are differentiated by including the values Driver, Firmware, or FCode/BIOS in the Classifications property.

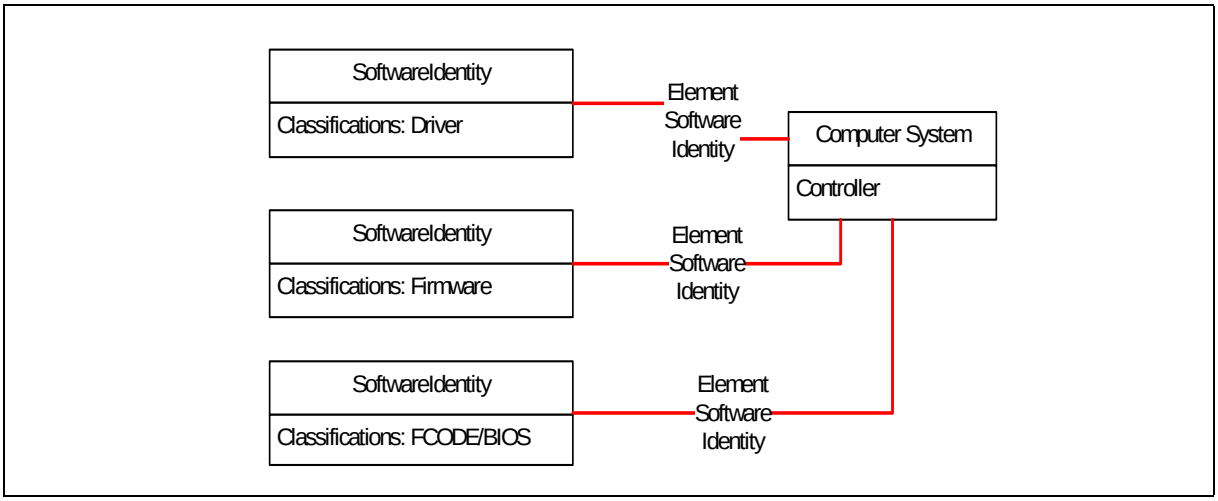

Figure 28 - Software Inventory Profile in Host Hardware RAID Controller

8.3.13 Implementation of Initiator Ports Profiles

The Host Hardware RAID Controller Profile utilizes the initiator ports profiles to model the back-end ports of the controller that are connected to the storage managed by the RAID controller.

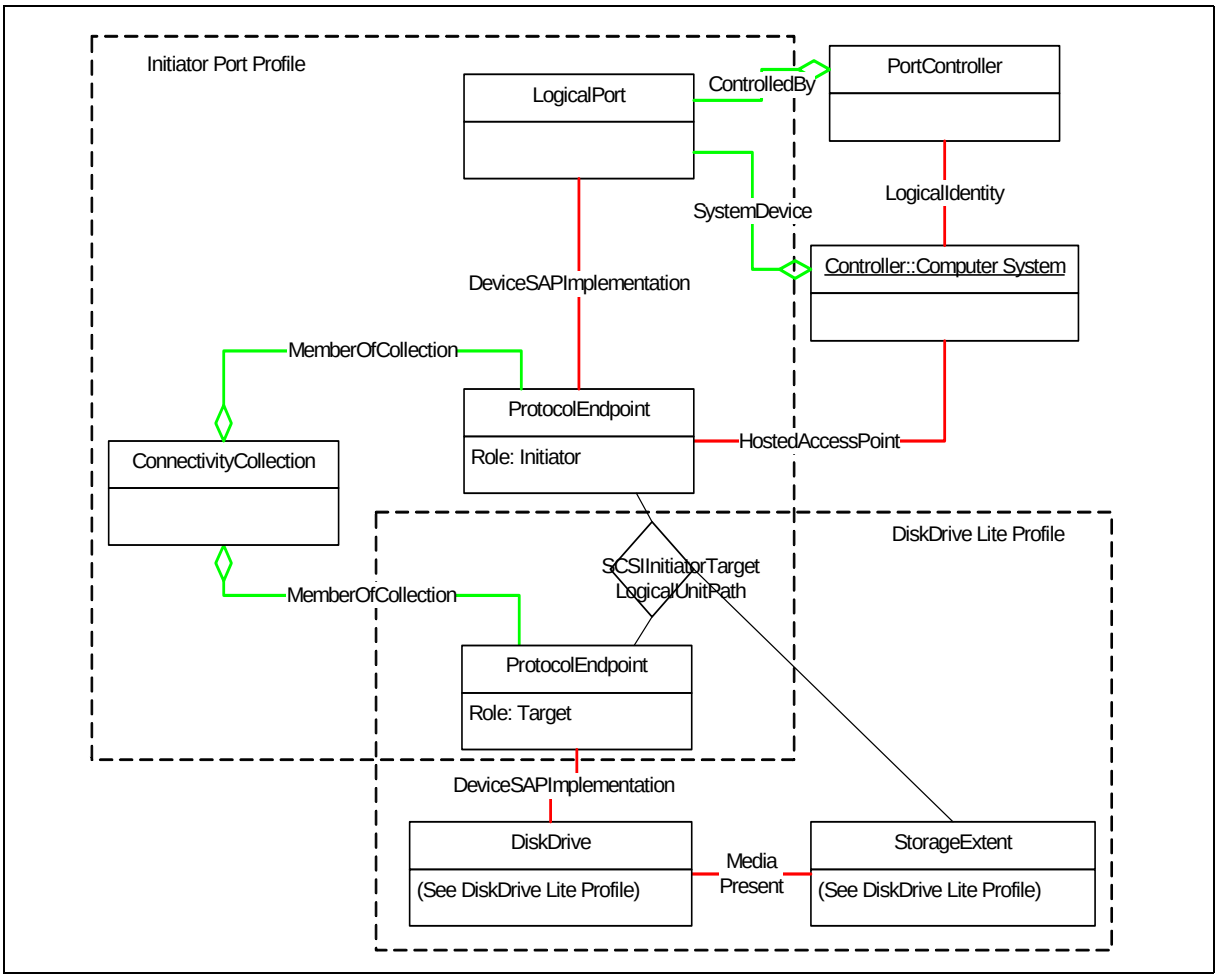

Figure 29 - Initiator Port profiles and Disk Drive Lite Profile

8.3.13.1 CIM_LogicalPort

CIM_LogicalPort represents the logical transport port on the back-end of the controller that is connected to the storage. This storage could be a drive cage housed inside the host or a storage device enclosure, like a JBOD. The LogicalPort class is intended to model the transport for storage commands in an abstract and agnostic manner. For example, the LogicalPort could represent a SCSI, SAS, ATA, or FC port depending on the controller implementation. Thus, the instance of this class shall be sub-classed to SPIPort, SASPort, FCPort or ATAPort depending on the subclass that best represents the transport type the controller supports for the backend port. The implementation shall not instantiate LogicalPort.

8.3.13.2 CIM_ProtocolEndpoint

The ProtocolEndpoint class represents the command set used between the controller and storage where the storage protocol is transmitted.

Like LogicalPort, the ProtocolEndpoint class is intended to model the storage protocol in an abstract manner. For example, the ProtocolEndpoint could represent a SCSI or ATA protocol. Thus, the instance of the ProtocolEndpoint shall be subclassed to SCSIProtocolEndpoint or ATAProtocolEndpoint.

8.3.14 Models for Imported Storage

In most cases, the storage imported by a Host Hardware Storage Controller is disk drives. It is possible that storage could be imported from another storage system (such as an Array) or non-disk devices (such as tape or optical media drives) could be attached to the RAID controller and passed through to the host.

8.3.14.1 Disks modeled with the Disk Drive Lite Profile

Individual disks are modeled using the Disk Drive Lite Profile (see *Storage Management Technical Specification, Part 4 Block Devices, 1.7.0 Rev 5* 10 Disk Drive Lite Profile).

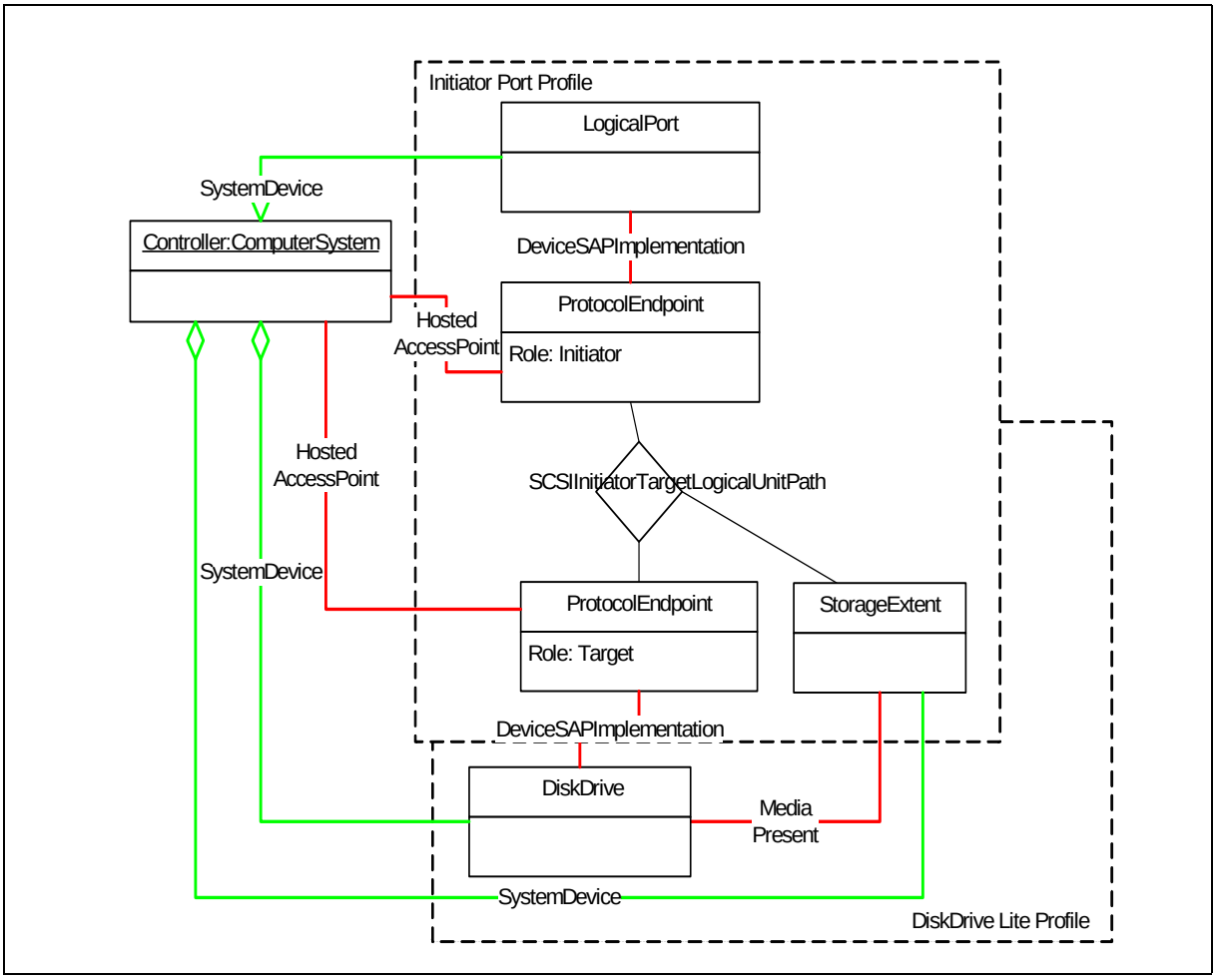

Figure 30 - Model for Imported Disks

To model one disk drive:

- The StorageExtent instance defined in Disk Drive Lite serves as the LogicalDevice defined in the initiator ports profiles
- The initiator and target ProtocolEndpoint instances (and appropriate associations) defined in the Disk Drive Lite Profile also serve as their equivalents in the Initiator Ports Profiles.

8.3.14.2 Virtual Volumes/Disks

Some host-based RAID controllers may be connected to another system that creates virtual volumes, such as a RAID Array. In this case, the implementation should model the storage from the array using a

specialization of the Generic Initiator Port Profile and instantiating instances of StorageExtent to represent the imported volumes. Figure 31, “Imported Virtual Volumes” shows this model.

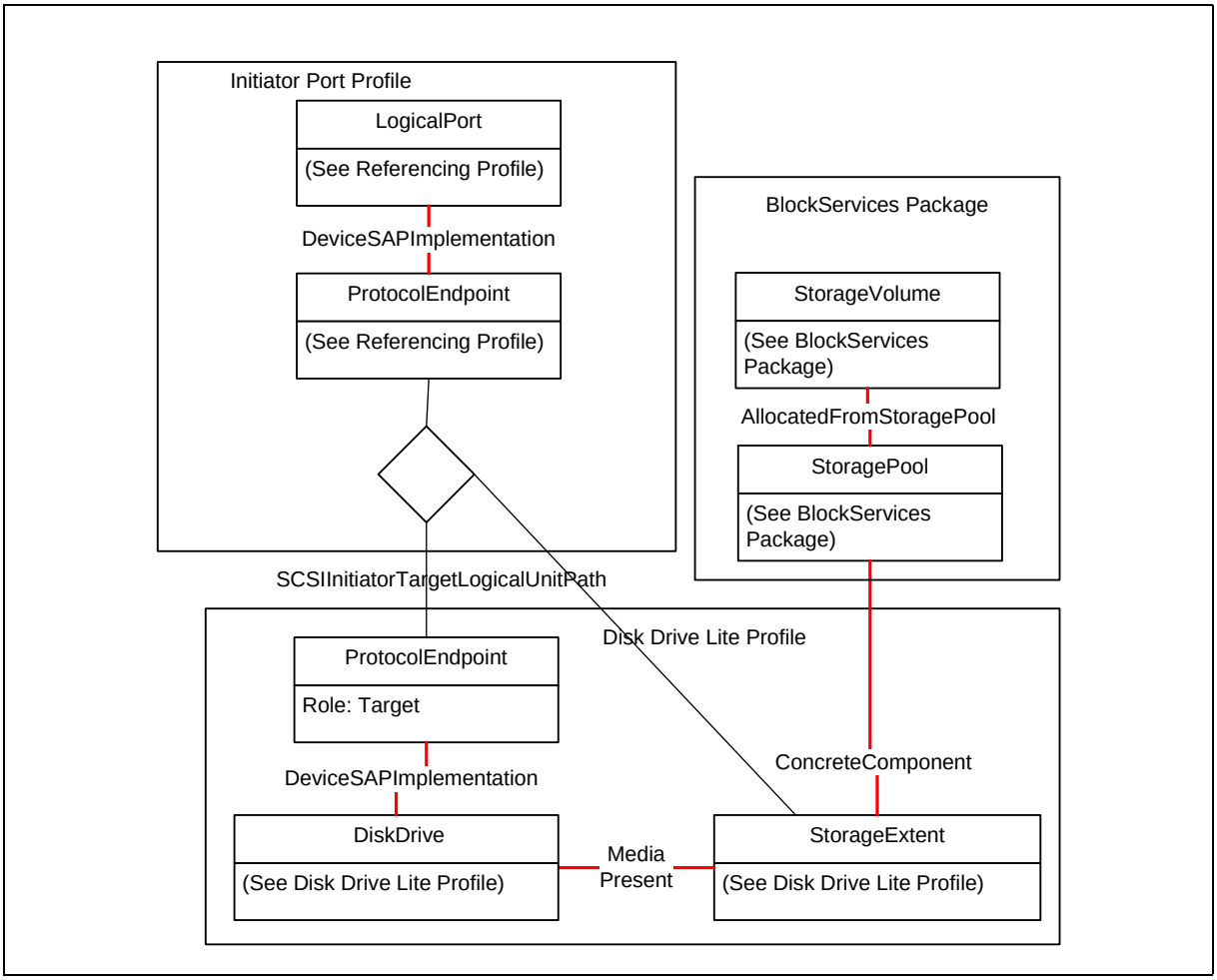

Figure 31 - Imported Virtual Volumes

8.3.14.3 Non-block Devices “Passed Through” with no Block Virtualization

A RAID controller may support attaching non-disk devices (tape or optical dries) and providing support to “pass through” these devices with no additional block virtualization. An instance of a LogicalDevice subclass (such as MediaAccessDevice) models these devices. Since these devices are exposed to the

host OS, this LogicalDevice is associated to the SCSIProtocolController (see 8.3.11 Implementation of DAPort and SCSIProtocolController). Figure 32, “Device “Pass Through” Example” shows this model.

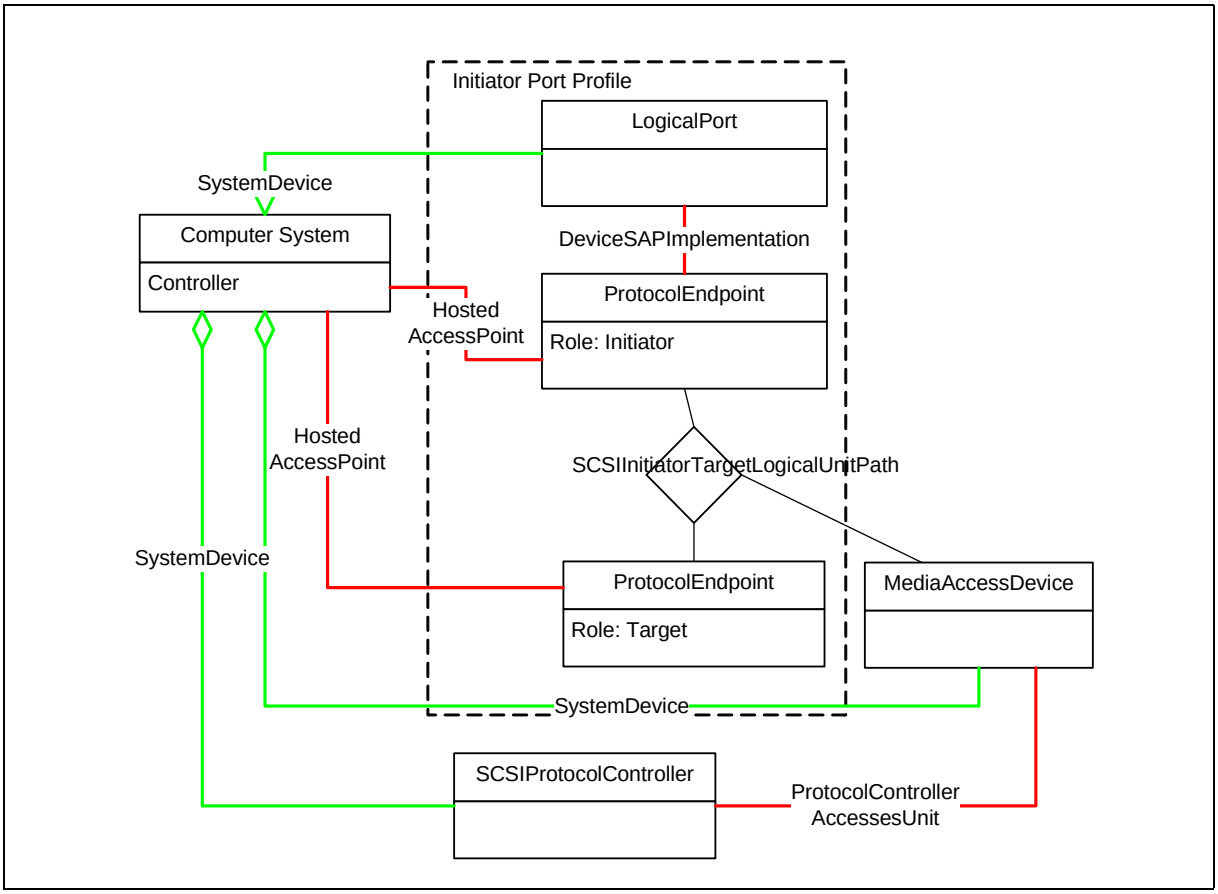

Figure 32 - Device “Pass Through” Example

8.3.15 Implementation of Extent Composition Profile

The Extent Composition Profile allows a Host Hardware RAID Controller Profile to expose the underlying storage composition of StoragePools and StorageVolumes. Composition of a StoragePool or StorageVolume is expressed through the use of StorageExtents associated to StoragePools and StorageExtents arranged in a hierarchical fashion.

For Host Hardware RAID Controller Profile, the use of the Extent Composition Profile is optional. However, the expectation is that most implementations will implement the Extent Composition Profile.

8.3.16 Disk Sparing

Many host-based RAID controllers have the ability to provide on-line, reserved storage components used to replace failed storage components. In the Host Hardware RAID Controller Profile, this behavior is modeled using the Disk Sparing Profile. If a controller supports on-line disk spares, then the implementation shall conditionally model this behavior using the Disk Sparing Profile.

Determination: At least one member of the primordial StorageExtents shall be associated to an instance of StorageRedundancySet. In turn, at least one or more of the StorageExtents that comprise a StoragePool or StorageVolume shall be associated to the same StorageRedundancySet.

8.3.17 Multi-function controllers

Many host-based RAID controllers support both RAID and non-RAID functionality on separate ports within the same controller card. If a controller supports multi-functional ports then the implementation shall conditionally model this behavior using multiple primordial storage pools.

Determination: Each primordial storage pool shall represent a set of RAID or non-RAID storage attached to the RAID or non-RAID port. The CIM_StorageCapabilities class associated to the primordial storage pool shall signify which pools have RAID capabilities and which do not. The following CIM_StorageCapabilities properties and values shall be used to signify non-RAID primordial storage pool:

- PackageRedundancy(Min/Max/Default) = 0
- DataRedundancy(Min/Max/Default) = 1
- ExtentStripeLengthDefault = 1
- UserDataStripeDepthDefault = null
- ParityLayoutDefault = nul

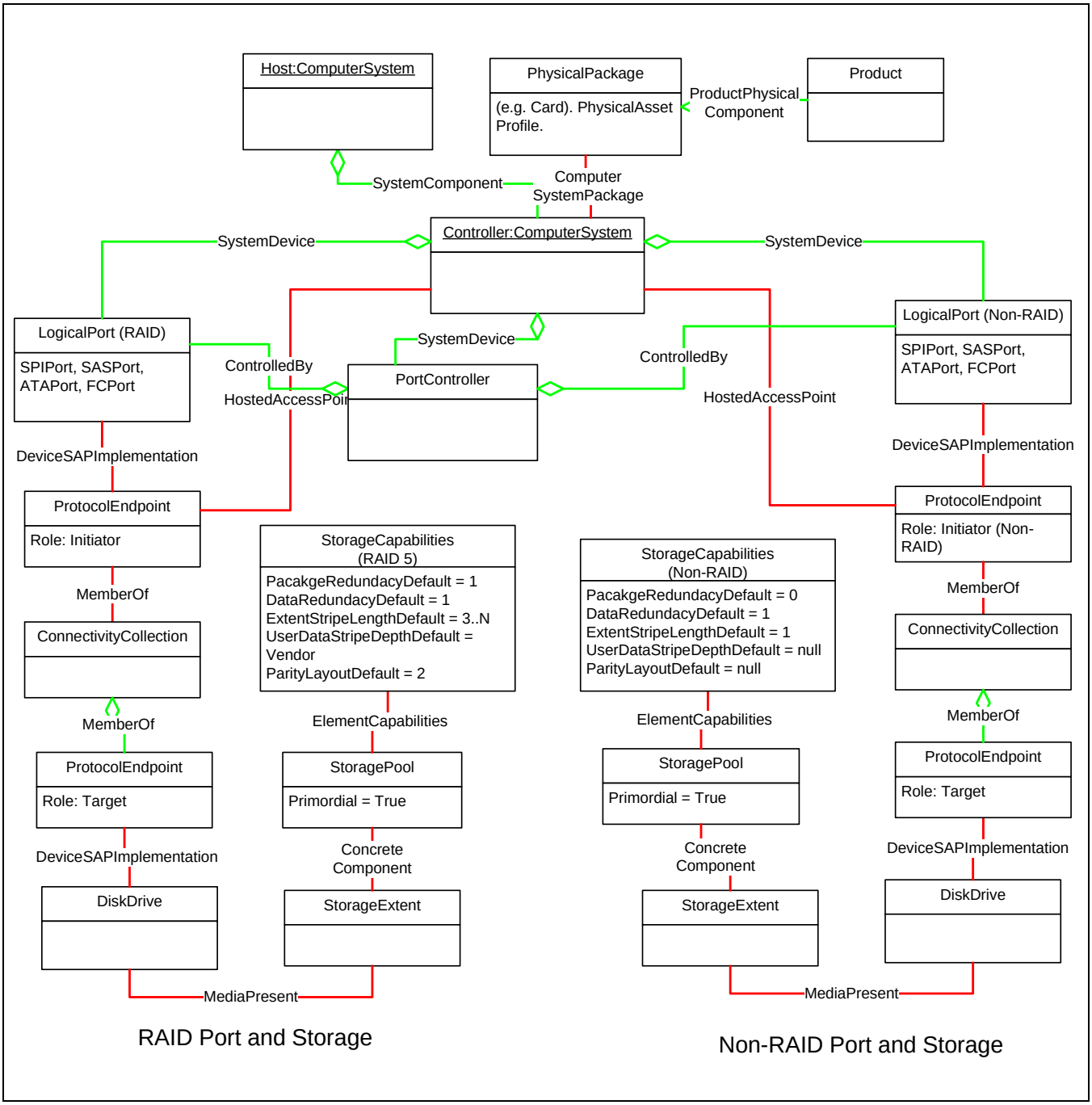

Figure 33 - Example of Mutli-Function Controllers

8.3.18 Health and Fault Management Consideration

Not defined in this standard.

8.3.19 Cascading Considerations

Not defined in this standard.

8.4 Methods

8.4.1 Extrinsic Methods of the Profile

8.4.1.1 AlarmDevice.SetAlarmIndicator

This method is used to enable/disable the audible or visual alarm indicator for the controller.

8.4.1.2 AlarmDevice.SetAlarmState

This method is used to set the state of the alarm. For the Host Hardware RAID Controller Profile the supported RequestedState parameters are:

- Off: Turns the alarm off for the current event, will alarm again for next event, state will automatically change from "off".
- Alternating: Turns the alarm on and off in an alternating fashion. This may be used to test the alarm.

8.4.2 Intrinsic Methods of this Profile

The profile supports read methods and association traversal. Specifically, the list of intrinsic operations supported

are as follows:

- GetInstance
- Associators
- AssociatorNames
- References
- ReferenceNames
- EnumerateInstances
- EnumerateInstanceNames

8.5 Use Cases

No recipes are included in this version of the standard.

8.6 CIM Elements

Table 48 describes the CIM elements for Host Hardware RAID Controller.

Table 48 - CIM Elements for Host Hardware RAID Controller

Element Name	Requirement	Description
8.6.1 CIM_AlarmDevice	Optional	Represents indicator LEDs.
8.6.2 CIM_AssociatedAlarm	Optional	Associates AlarmDevice and LogicalPort.
8.6.3 CIM_ComputerSystem (Host Hardware RAID Controller)	Mandatory	System that represents the Host Hardware RAID controller. Associated to RegisteredProfile.
8.6.4 CIM_ComputerSystemPackage (Associates PhysicalPackage to ComputerSystem)	Mandatory	Associates controller ComputerSystem and PhysicalPackage from the Physical Asset profile.
8.6.5 CIM_ControlledBy	Mandatory	Associates PortController to LogicalPorts.

Table 48 - CIM Elements for Host Hardware RAID Controller

Element Name	Requirement	Description
8.6.6 CIM_ElementCapabilities (ImplementationCapabilities to System)	Optional	Associates the conformant Host Hardware RAID Controller ComputerSystem to the CIM_ImplementationCapabilities supported by the implementation.
8.6.7 CIM_ImplementationCapabilities (ImplementationCapabilities)	Optional	The capabilities of the profile implementation.
8.6.8 CIM_LogicalIdentity	Mandatory	Used to associate the ComputerSystem representing the controller with PortController.
8.6.9 CIM_MediaAccessDevice	Mandatory	Represents a tape or optical drive.
8.6.10 CIM_PortController	Mandatory	Serves as a component of the server ComputerSystem and is associated to the controller ComputerSystem.
8.6.11 CIM_Product	Mandatory	Asset information about the RAID controller.
8.6.12 CIM_ProductPhysicalComponent	Mandatory	Associates Product and PhysicalPackage.
8.6.13 CIM_ProtocolControllerForUnit (Extent or MediaAccessDevice)	Mandatory	Associates SCSIProtocolController to StorageExtent or MediaAccessDevice.
8.6.14 CIM_ProtocolControllerForUnit (Volume)	Mandatory	Associated ProtocolController to StorageVolume.
8.6.15 CIM_Realizes (Associates PhysicalPackage to PortController)	Mandatory	Associates PortController and PhysicalPackage from the Physical Asset profile.
8.6.16 CIM_SAPAvailableForElement	Mandatory	Associates SCSIProtocolController to the DAPort ProtocolEndpoint.
8.6.17 CIM_SCSIProtocolController	Mandatory	Represents the target/device aspects of storage exported by the RAID controller.
8.6.18 CIM_SoftwareIdentity (Driver)	Mandatory	Override SoftwareIdentity from Software Inventory profile to assure Classifications property includes Driver.
8.6.19 CIM_SoftwareIdentity (FCode/BIOS)	Optional	Override SoftwareIdentity from Software Inventory profile to assure Classifications property includes FCODE/BIOS.
8.6.20 CIM_SoftwareIdentity (Firmware)	Optional	Override SoftwareIdentity from Software Inventory profile to assure Classifications property includes Firmware.
8.6.21 CIM_StorageExtent	Optional	Models an imported volume from an external array.
8.6.22 CIM_SystemComponent	Mandatory	Associates ComputerSystems representing the hosting system and the RAID controller.
8.6.23 CIM_SystemDevice (Associates System to AlarmDevice)	Optional	Associates System to AlarmDevice.
8.6.24 CIM_SystemDevice (Associates controller system to PortController)	Mandatory	Associates controller system to PortController.
8.6.25 CIM_SystemDevice (System to SCSIProtocolController)	Mandatory	Links SCSIProtocolController to the controller system.
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_ComputerSystem	Mandatory	Addition of a new Host Hardware RAID controller instance.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_ComputerSystem	Mandatory	Deletion of an Host Hardware RAID controller instance.

8.6.1 CIM_AlarmDevice

Represents indicator LEDs.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 49 describes class CIM_AlarmDevice.

Table 49 - SMI Referenced Properties/Methods for CIM_AlarmDevice

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	
VisibleAlarm		Mandatory	
AudibleAlarm		Mandatory	
Urgency		Mandatory	
SetAlarmState()		Mandatory	
SetAlarmIndicator()		Mandatory	

8.6.2 CIM_AssociatedAlarm

Associates AlarmDevice and LogicalPort.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 50 describes class CIM_AssociatedAlarm.

Table 50 - SMI Referenced Properties/Methods for CIM_AssociatedAlarm

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

8.6.3 CIM_ComputerSystem (Host Hardware RAID Controller)

System that represents the Host Hardware RAID controller.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Shall be associated to RegisteredProfile using ElementConformsToProfile association. The RegisteredProfile instance shall have RegisteredName set to 'Host Hardware RAID Controller', RegisteredOrganization set to 'SNIA', and RegisteredVersion set to '1.5.0'.

Table 51 describes class CIM_ComputerSystem (Host Hardware RAID Controller).

Table 51 - SMI Referenced Properties/Methods for CIM_ComputerSystem (Host Hardware RAID Controller)

Properties	Flags	Requirement	Description & Notes
CreationClassName		Mandatory	
Name		Mandatory	Identifier for the Host Hardware RAID Controller.
NameFormat		Mandatory	Format for Name property. Shall be 'HID' for a hardware ID or 'Other'.
ElementName		Mandatory	User friendly name.
Dedicated		Mandatory	Shall include 30 (Host-Based RAID controller).
PrimaryOwnerContact	M	Optional	Contact a details for owner.
PrimaryOwnerName	M	Optional	Owner of the Host Hardware RAID.

8.6.4 CIM_ComputerSystemPackage (Associates PhysicalPackage to ComputerSystem)

Associates controller ComputerSystem and PhysicalPackage from the Physical Asset profile. Overrides the definition in the PhysicalAsset profile to clarify that this association references the controller and not the hosting ComputerSystem.

Requirement: Mandatory

Table 52 describes class CIM_ComputerSystemPackage (Associates PhysicalPackage to ComputerSystem).

Table 52 - SMI Referenced Properties/Methods for CIM_ComputerSystemPackage (Associates Physical-Package to ComputerSystem)

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	Reference to ComputerSystem with Dedicated=30 (Host-based RAID controller).
Antecedent		Mandatory	

8.6.5 CIM_ControlledBy

Associates PortController to LogicalPorts.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 53 describes class CIM_ControlledBy.

Table 53 - SMI Referenced Properties/Methods for CIM_ControlledBy

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	Reference to FCPort.
Antecedent		Mandatory	Reference to PortController.

8.6.6 CIM_ElementCapabilities (ImplementationCapabilities to System)

Associates the conformant Host Hardware RAID Controller ComputerSystem to the CIM_ImplementationCapabilities supported by the implementation.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 54 describes class CIM_ElementCapabilities (ImplementationCapabilities to System).

Table 54 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (ImplementationCapabilities to System)

Properties	Flags	Requirement	Description & Notes
Capabilities		Mandatory	The ImplementationCapabilities.
ManagedElement		Mandatory	The conformant Host Hardware RAID Controller ComputerSystem that has ImplementationCapabilities.

8.6.7 CIM_ImplementationCapabilities (ImplementationCapabilities)

The capabilities (features) of the profile implementation.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 55 describes class CIM_ImplementationCapabilities (ImplementationCapabilities).

Table 55 - SMI Referenced Properties/Methods for CIM_ImplementationCapabilities (ImplementationCapabilities)

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	An opaque, unique id for the implementation capability of an implementation.
ElementName		Optional	A provider supplied user-friendly name for this CIM_ImplementationCapabilities element.

Table 55 - SMI Referenced Properties/Methods for CIM_ImplementationCapabilities (ImplementationCapabilities)

Properties	Flags	Requirement	Description & Notes
SupportedCapacityOptimizations		Mandatory	This array of strings lists the capacity optimization techniques that are supported by the implementation. Valid string values are "none" | "SNIA:Thin Provisioning" | "SNIA:Data Compression" | "SNIA:Data Deduplication".
SupportedViews		Mandatory	This array of strings lists the view classes that are supported by the implementation. Valid string values are "none" | "SNIA:VolumeView" | "SNIA:DiskDriveView" | "SNIA:StoragePoolView" | "SNIA:ReplicaPairView".

8.6.8 CIM_LogicalIdentity

Associates the ComputerSystem representing the controller and the PortController.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 56 describes class CIM_LogicalIdentity.

Table 56 - SMI Referenced Properties/Methods for CIM_LogicalIdentity

Properties	Flags	Requirement	Description & Notes
SameElement		Mandatory	Reference to the ComputerSystem respresenting the controller.
SystemElement		Mandatory	Reference to the PortController.

8.6.9 CIM_MediaAccessDevice

Represents a tape or optical drive.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 57 describes class CIM_MediaAccessDevice.

Table 57 - SMI Referenced Properties/Methods for CIM_MediaAccessDevice

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	
Name		Mandatory	

Table 57 - SMI Referenced Properties/Methods for CIM_MediaAccessDevice

Properties	Flags	Requirement	Description & Notes
OperationalStatus		Mandatory	Shall be 2|5|6|8|10|11 (Okay or Predictive Failure or Error or Starting or Stopping or Stopped).
LocationIndicator		Optional	

8.6.10 CIM_PortController

Serves as a component of the server ComputerSystem and is associated to the controller ComputerSystem.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 58 describes class CIM_PortController.

Table 58 - SMI Referenced Properties/Methods for CIM_PortController

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	
ControllerType		Mandatory	Shall be 1 or 4 (Other or FC).
OtherControllerType		Conditional	Conditional requirement: For non-FC, PortController.OtherControllerType is mandatory.Shall be SPI or SAS or ATA or SAS/SATA.

8.6.11 CIM_Product

Asset information about the RAID controller.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 59 describes class CIM_Product.

Table 59 - SMI Referenced Properties/Methods for CIM_Product

Properties	Flags	Requirement	Description & Notes
Name		Mandatory	Shall have the same value as PhysicalPackage.Model.
IdentifyingNumber		Mandatory	Shall have the same value as PhysicalPackage.SerialNumber.
Vendor		Mandatory	Shall have the same value as PhysicalPackage.Manufacturer.

Table 59 - SMI Referenced Properties/Methods for CIM_Product

Properties	Flags	Requirement	Description & Notes
Version		Mandatory	Shall have the same value as PhysicalPackage.Version. Represents a version for the physical element.
ElementName		Mandatory	

8.6.12 CIM_ProductPhysicalComponent

Associates Product and PhysicalPackage.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 60 describes class CIM_ProductPhysicalComponent.

Table 60 - SMI Referenced Properties/Methods for CIM_ProductPhysicalComponent

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	
PartComponent		Mandatory	

8.6.13 CIM_ProtocolControllerForUnit (Extent or MediaAccessDevice)

Associates SCSIProtocolController to StorageExtent or MediaAccessDevice.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 61 describes class CIM_ProtocolControllerForUnit (Extent or MediaAccessDevice).

Table 61 - SMI Referenced Properties/Methods for CIM_ProtocolControllerForUnit (Extent or MediaAccess-Device)

Properties	Flags	Requirement	Description & Notes
DeviceNumber		Mandatory	Logical Unit Number of the associated Device. Shall be formatted as unseparated uppercase hexadecimal digits, with no leading 0x.
DeviceAccess		Optional	The access rights granted to the referenced logical unit as exposed through referenced ProtocolController.
Antecedent		Mandatory	
Dependent		Mandatory	Reference to a StorageVolume.

8.6.14 CIM_ProtocolControllerForUnit (Volume)

Associated ProtocolController to StorageVolume.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 62 describes class CIM_ProtocolControllerForUnit (Volume).

Table 62 - SMI Referenced Properties/Methods for CIM_ProtocolControllerForUnit (Volume)

Properties	Flags	Requirement	Description & Notes
DeviceNumber		Mandatory	Address (e.g. LUN) of the associated Device. Shall be formatted as unseparated uppercase hexadecimal digits, with no leading 0x.
DeviceAccess		Optional	The access rights granted to the referenced logical unit as exposed through referenced ProtocolController.
Antecedent		Mandatory	
Dependent		Mandatory	Reference to a StorageVolume.

8.6.15 CIM_Realizes (Associates PhysicalPackage to PortController)

Associates PortController and PhysicalPackage from the Physical Asset profile.

Requirement: Mandatory

Table 63 describes class CIM_Realizes (Associates PhysicalPackage to PortController).

Table 63 - SMI Referenced Properties/Methods for CIM_Realizes (Associates PhysicalPackage to PortController)

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

8.6.16 CIM_SAPAvailableForElement

Associates SCSIProtocolController to the DAPort ProtocolEndpoint.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 64 describes class CIM_SAPAvailableForElement.

Table 64 - SMI Referenced Properties/Methods for CIM_SAPAvailableForElement

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	
AvailableSAP		Mandatory	

8.6.17 CIM_SCSIProtocolController

Represents the target/device aspects of storage exported by the RAID controller.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 65 describes class CIM_SCSIProtocolController.

Table 65 - SMI Referenced Properties/Methods for CIM_SCSIProtocolController

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	

8.6.18 CIM_SoftwareIdentity (Driver)

Override SoftwareIdentity from Software Inventory profile to assure Classifications property includes Driver.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 66 describes class CIM_SoftwareIdentity (Driver).

Table 66 - SMI Referenced Properties/Methods for CIM_SoftwareIdentity (Driver)

Properties	Flags	Requirement	Description & Notes
Classifications		Mandatory	Shall include 2 (Driver).

8.6.19 CIM_SoftwareIdentity (FCode/BIOS)

Override SoftwareIdentity from Software Inventory profile to assure Classifications property includes FCODE/BIOS.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 67 describes class CIM_SoftwareIdentity (FCode/BIOS).

Table 67 - SMI Referenced Properties/Methods for CIM_SoftwareIdentity (FCode/BIOS)

Properties	Flags	Requirement	Description & Notes
Classifications		Mandatory	Shall include 11 (FCODE/BIOS).

8.6.20 CIM_SoftwareIdentity (Firmware)

Override SoftwareIdentity from Software Inventory profile to assure Classifications property includes Firmware.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 68 describes class CIM_SoftwareIdentity (Firmware).

Table 68 - SMI Referenced Properties/Methods for CIM_SoftwareIdentity (Firmware)

Properties	Flags	Requirement	Description & Notes
Classifications		Mandatory	Shall include 10 (Firmware).

8.6.21 CIM_StorageExtent

Created By: External
Modified By: External
Deleted By: External
Requirement: Optional

Table 69 describes class CIM_StorageExtent.

Table 69 - SMI Referenced Properties/Methods for CIM_StorageExtent

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	
BlockSize		Mandatory	
NumberOfBlocks		Mandatory	The number of blocks as reported by the hardware.
ConsumableBlocks		Mandatory	The number of usable blocks.
Primordial		Mandatory	Shall be true.
OperationalStatus		Mandatory	

8.6.22 CIM_SystemComponent

Associates ComputerSystems representing the hosting system and the RAID controller.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 70 describes class CIM_SystemComponent.

Table 70 - SMI Referenced Properties/Methods for CIM_SystemComponent

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	ComputerSystem with Dedicated=0 (Not Dedicated) hosting controllers.
PartComponent		Mandatory	ComputerSystem with Dedicated=30(Host-based RAID controller)representing a controller.

8.6.23 CIM_SystemDevice (Associates System to AlarmDevice)

Associates System to AlarmDevice.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 71 describes class CIM_SystemDevice (Associates System to AlarmDevice).

Table 71 - SMI Referenced Properties/Methods for CIM_SystemDevice (Associates System to AlarmDevice)

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	Reference to ComputerSystem with Dedicated=30 (Host-based RAID controller).
PartComponent		Mandatory	

8.6.24 CIM_SystemDevice (Associates controller system to PortController)

Associates controller system to PortController.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 72 describes class CIM_SystemDevice (Associates controller system to PortController).

Table 72 - SMI Referenced Properties/Methods for CIM_SystemDevice (Associates controller system to PortController)

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	Reference to ComputerSystem with Dedicated = 30 (Host-based RAID controller).
PartComponent		Mandatory	

8.6.25 CIM_SystemDevice (System to SCSIProtocolController)

Links SCSIProtocolController to the controller system.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 73 describes class CIM_SystemDevice (System to SCSIProtocolController).

Table 73 - SMI Referenced Properties/Methods for CIM_SystemDevice (System to SCSIProtocolController)

Properties	Flags	Requirement	Description & Notes
PartComponent		Mandatory	
GroupComponent		Mandatory	Reference to ComputerSystem with Dedicated = 30 (Host-based RAID controller).

EXPERIMENTAL

EXPERIMENTAL

9 iSCSI Initiator Profile

9.1 Description

9.1.1 Synopsis

Profile Name: iSCSI Initiator (Autonomous Profile)

Version: 1.1.0

Organization: SNIA

Central Class: PortController

Scoping Class: ComputerSystem

Related Profiles: Table 74 describes the supported profiles for iSCSI Initiator.

Table 74 - Supported Profiles for iSCSI Initiator

Profile Name	Organization	Version	Requirement	Description
iSCSI Initiator Ports	SNIA	1.2.0	Optional	

9.1.2 Overview

An iSCSI initiator is the hardware and driver combination that acts as a client to an iSCSI target device. iSCSI initiators may utilize general –purpose Network Interface Cards (NICs) or hardware optimized for storage such as TCP Offload Engines (TOEs). iSCSI initiators may be running on a customer server or the “back end” of a bridge or virtualizer.

iSCSI terminology, shown in Table 75, spans SCSI and network concepts and introduces new terms. Table 75 is a summary of some key iSCSI terms, their equivalent CIM classes, and definitions (from the IETF iSCSI RFC).

Table 75 - iSCSI Terminology

iSCSI Term	CIM Class Name	Notes
Network Entity	ComputerSystem	The Network Entity represents a device or gateway that is accessible from the IP network. A Network Entity shall have one or more Network Portals, each of which can be used to gain access to the IP network by some iSCSI Nodes contained in that Network Entity.
Session	iSCSISession	The group of TCP connections that link an initiator with a target form a session (loosely equivalent to a SCSI I-T nexus). TCP connections can be added and removed from a session. Across all connections within a session, an initiator sees one and the same target.
Connection	iSCSIConnection	A connection is a TCP connection. Communication between the initiator and target occurs over one or more TCP connections. The TCP connections carry control messages, SCSI commands, parameters, and data within iSCSI Protocol Data Units (iSCSI PDUs).
SCSI Port	iSCSIProtocolEndpoint	A SCSI Port using an iSCSI service delivery subsystem. A collection of Network Portals that together act as a SCSI initiator or target.

Table 75 - iSCSI Terminology (Continued)

Network Portal	TCPProtocolEndpoint, IPProtocolEndpoint, EthernetPort	The Network Portal is a component of a Network Entity that has a TCP/IP network address and that may be used by an iSCSI Node within that Network Entity for the connection(s) within one of its iSCSI sessions. A Network Portal in an initiator is identified by its IP address. A Network Portal in a target is identified by its IP address and its listening TCP port.
Node	SCSIProtocolController	The iSCSI Node represents a single iSCSI initiator or iSCSI target. There are one or more iSCSI Nodes within a Network Entity. The iSCSI Node is accessible via one or more Network Portals. An iSCSI Node is identified by its iSCSI Name. The separation of the iSCSI Name from the addresses used by and for the iSCSI Node allows multiple iSCSI nodes to use the same address, and the same iSCSI node to use multiple addresses.

This profile requires the iSCSI Initiator Port Profile (see *Storage Management Technical Specification, Part 3 Common Profiles, 1.7.0 Rev 5*, 15 iSCSI Initiator Port Profile) that includes classes (EthernetPort, iSCSIProtocolEndoint) that model SCSI ports and network portals.

Figure 34 models the relationships between the iSCSI port classes and physical and product classes. A single iSCSI card may contain multiple Ethernet ports PhysicalPackage subclass Card models an add-in card with multiple Ethernet ports. Other PhysicalPackage subclasses may be used to model Ethernet ports embedded on a mainboard. PortController models a common management interface to multiple Ethernet ports.

ComputerSystem models the system hosting the initiator components. This is the same instance as iSCSI Network Entity in Figure 34.

An implementation includes single instances of PhysicalPackage, Product, and PortController, plus SoftwareIdentity instances for the driver, firmware, and Fcode/BIOS. The Product instance may be shared across cards with the same make and model

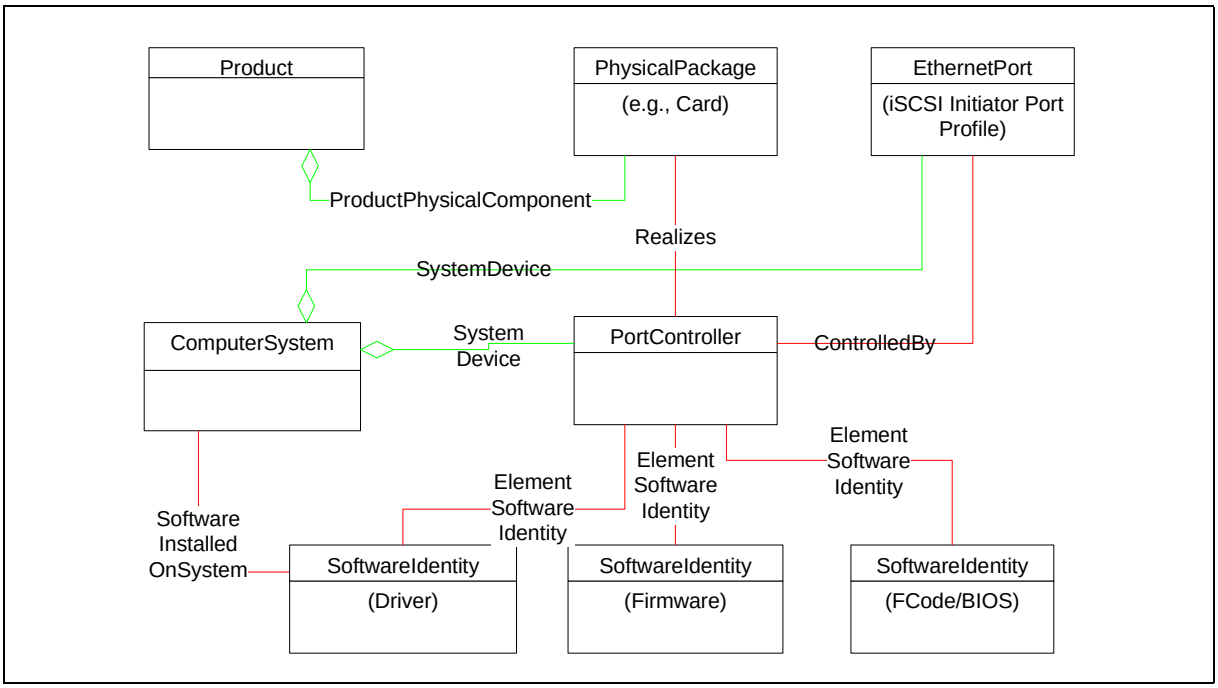

Figure 34 - iSCSI Product and Package Model

9.1.3 Sessions and Connections

A session is an active communication stream between an iSCSI initiator port and an iSCSI target port. However, any given session may contain part or all of the TCP/IP addresses within a Portal Group. Conceptually, a Portal Group is a pool of addresses which may be used to create/receive a session.

The implementation may optionally model iSCSI sessions and connections with instances of iSCSISession and iSCSIConnection classes associate to iSCSIProtocolEndpoint and TCPProtocolEndpoint (respectively) using EndpointOfNetworkPipe association.

Figure 35 shows the iSCSI Sessions and Connections Model.

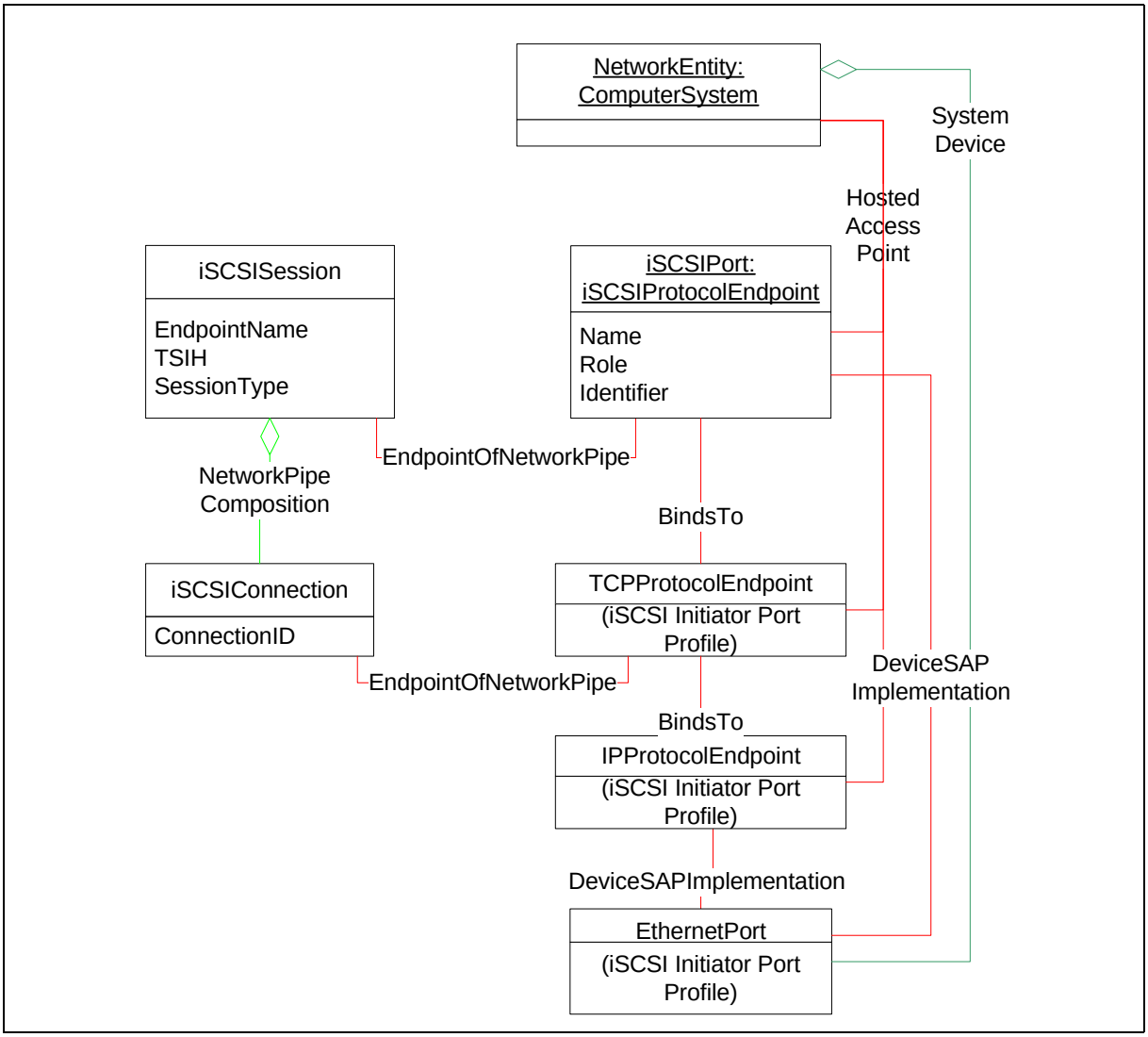

Figure 35 - iSCSI Sessions and Connections Model

There should be a single instance of SCSIProtocolController representing the initiator iSCSI node, shown in Figure 36. This is associated via SystemDevice to the ComputerSystem. See Figure 34.

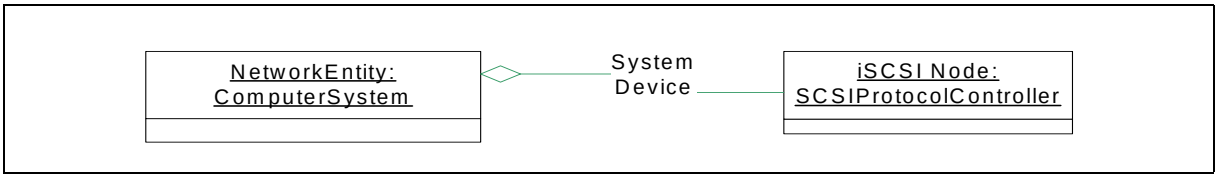

Figure 36 - iSCSI Initiator Node

9.1.4 Durable Names and Correlatable IDs of the Profile

The Name property for the iSCSI node (SCSIProtocolController) shall be a compliant iSCSI name as described in *Storage Management Technical Specification, Part 3 Common Profiles, 1.7.0 Rev 5* 7.8 and NameFormat shall be set to “iSCSI Name”.

The Name property for iSCSIProtocolEndpoint shall be a compliant iSCSI name as described in *Storage Management Technical Specification, Part 3 Common Profiles, 1.7.0 Rev 5* 7.8 and ConnectionType shall be set to “iSCSI”.

The Name property for EthernetPort shall be a compliant iSCSI name as described in *Storage Management Technical Specification, Part 2 Common Architecture, 1.7.0 Rev 5*, 7.8.

9.2 Health and Fault Management Considerations

The status of an Ethernet port may be determined by the value of the OperationalStatus property. Table 76 defines the possible states that shall be supported for EthernetPort.OperationalStatus. The main OperationalStatus shall be the first element in the array.

Table 76 - OperationalStatus Values

OperationalStatus	Description
OK	Port is online
Error	Port has a failure
Stopped	Port is disabled
InService	Port is in Self Test

9.3 Methods of the Profile

Not defined in this standard

9.4 Use Cases

Not defined in this version of the standard.

9.5 CIM Elements

Table 77 describes the CIM elements for iSCSI Initiator.

Table 77 - CIM Elements for iSCSI Initiator

Element Name	Requirement	Description
9.5.1 CIM_BindsTo (TCPProtocolEndpoint to IPProtocolEndpoint)	Mandatory	
9.5.2 CIM_BindsTo (iSCSIProtocolEndpoint to TCPProtocolEndpoint)	Mandatory	
9.5.3 CIM_ComputerSystem	Mandatory	Associated to RegisteredProfile.
9.5.4 CIM_ControlledBy	Optional	
9.5.5 CIM_DeviceSAPImplementation (EthernetPort to IPProtocolEndpoint)	Optional	
9.5.6 CIM_DeviceSAPImplementation (EthernetPort to iSCSIProtocolEndpoint)	Optional	
9.5.7 CIM_ElementSoftwareIdentity	Mandatory	
9.5.8 CIM_EndpointOfNetworkPipe (Between iSCSI Session and iSCSIProtocolENdpoint)	Mandatory	
9.5.9 CIM_EndpointOfNetworkPipe (Between iSCSI connection and TCPProtocolENdpoint)	Mandatory	
9.5.10 CIM_InstalledSoftwareIdentity	Optional	
9.5.11 CIM_NetworkPipeComposition	Mandatory	
9.5.12 CIM_PhysicalPackage	Mandatory	
9.5.13 CIM_PortController	Optional	
9.5.14 CIM_Product	Mandatory	
9.5.15 CIM_ProductPhysicalComponent	Mandatory	
9.5.16 CIM_Realizes	Mandatory	
9.5.17 CIM_SAPAvailableForElement	Mandatory	
9.5.18 CIM_SCSIProtocolController	Mandatory	
9.5.19 CIM_SoftwareIdentity	Optional	
9.5.20 CIM_SystemDevice (to EthernetPort)	Mandatory	
9.5.21 CIM_SystemDevice (to PortController)	Mandatory	
9.5.22 CIM_SystemDevice (to ProtocolController)	Mandatory	
9.5.23 CIM_iSCSIConnection	Optional	
9.5.24 CIM_iSCSISession	Mandatory	
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_PortController	Optional	PortController (HBA) Creation.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_PorController	Optional	PortController (HBA) Removal.

9.5.1 CIM_BindsTo (TCPProtocolEndpoint to IPProtocolEndpoint)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 78 describes class CIM_BindsTo (TCPProtocolEndpoint to IPProtocolEndpoint).

Table 78 - SMI Referenced Properties/Methods for CIM_BindsTo (TCPProtocolEndpoint to IPProtocolEndpoint)

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

9.5.2 CIM_BindsTo (iSCSIProtocolEndpoint to TCPProtocolEndpoint)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 79 describes class CIM_BindsTo (iSCSIProtocolEndpoint to TCPProtocolEndpoint).

Table 79 - SMI Referenced Properties/Methods for CIM_BindsTo (iSCSIProtocolEndpoint to TCPProtocolEndpoint)

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

9.5.3 CIM_ComputerSystem

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Shall be associated to RegisteredProfile using ElementConformsToProfile association. The RegisteredProfile instance shall have RegisteredName set to 'iSCSI Initiator', RegisteredOrganization set to 'SNIA', and RegisteredVersion set to '1.1.0'.

Table 80 describes class CIM_ComputerSystem.

Table 80 - SMI Referenced Properties/Methods for CIM_ComputerSystem

Properties	Flags	Requirement	Description & Notes
CreationClassName		Mandatory	
Name		Mandatory	The name of the host containing the iSCSI initiator.

Table 80 - SMI Referenced Properties/Methods for CIM_ComputerSystem

Properties	Flags	Requirement	Description & Notes
ElementName		Mandatory	
NameFormat		Mandatory	
OtherIdentifyingInfo	C	Mandatory	
OperationalStatus		Mandatory	
Dedicated		Mandatory	Shall be "Not Dedicated".
OtherDedicatedDescriptions		Optional	

9.5.4 CIM_ControlledBy

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 81 describes class CIM_ControlledBy.

Table 81 - SMI Referenced Properties/Methods for CIM_ControlledBy

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

9.5.5 CIM_DeviceSAPImplementation (EthernetPort to IPProtocolEndpoint)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 82 describes class CIM_DeviceSAPImplementation (EthernetPort to IPProtocolEndpoint).

Table 82 - SMI Referenced Properties/Methods for CIM_DeviceSAPImplementation (EthernetPort to IPProtocolEndpoint)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

9.5.6 CIM_DeviceSAPImplementation (EthernetPort to iSCSIProtocolEndpoint)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 83 describes class CIM_DeviceSAPImplementation (EthernetPort to iSCSIProtocolEndpoint).

Table 83 - SMI Referenced Properties/Methods for CIM_DeviceSAPImplementation (EthernetPort to iSCSIProtocolEndpoint)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

9.5.7 CIM_ElementSoftwareIdentity

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 84 describes class CIM_ElementSoftwareIdentity.

Table 84 - SMI Referenced Properties/Methods for CIM_ElementSoftwareIdentity

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

9.5.8 CIM_EndpointOfNetworkPipe (Between iSCSI Session and iSCSIProtocolENdpoint)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 85 describes class CIM_EndpointOfNetworkPipe (Between iSCSI Session and iSCSIProtocolENdpoint).

Table 85 - SMI Referenced Properties/Methods for CIM_EndpointOfNetworkPipe (Between iSCSI Session and iSCSIProtocolENdpoint)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

9.5.9 CIM_EndpointOfNetworkPipe (Between iSCSI connection and TCPProtocolENdpoint)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 86 describes class CIM_EndpointOfNetworkPipe (Between iSCSI connection and TCPProtocolENdpoint).

Table 86 - SMI Referenced Properties/Methods for CIM_EndpointOfNetworkPipe (Between iSCSI connection and TCPProtocolENdpoint)

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

9.5.10 CIM_InstalledSoftwareIdentity

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 87 describes class CIM_InstalledSoftwareIdentity.

Table 87 - SMI Referenced Properties/Methods for CIM_InstalledSoftwareIdentity

Properties	Flags	Requirement	Description & Notes
InstalledSoftware		Mandatory	
System		Mandatory	

9.5.11 CIM_NetworkPipeComposition

Requirement: Mandatory

Table 88 describes class CIM_NetworkPipeComposition.

Table 88 - SMI Referenced Properties/Methods for CIM_NetworkPipeComposition

Properties	Flags	Requirement	Description & Notes
PartComponent		Mandatory	
GroupComponent		Mandatory	

9.5.12 CIM_PhysicalPackage

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 89 describes class CIM_PhysicalPackage.

Table 89 - SMI Referenced Properties/Methods for CIM_PhysicalPackage

Properties	Flags	Requirement	Description & Notes
Manufacturer		Mandatory	Maps to IMA_PHBA_PROPERTIES.vendor.
Model		Mandatory	Maps to IMA_PHBA_PROPERTIES.model.

9.5.13 CIM_PortController

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 90 describes class CIM_PortController.

Table 90 - SMI Referenced Properties/Methods for CIM_PortController

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	
ControllerType		Mandatory	

9.5.14 CIM_Product

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 91 describes class CIM_Product.

Table 91 - SMI Referenced Properties/Methods for CIM_Product

Properties	Flags	Requirement	Description & Notes
ElementName		Mandatory	
Name		Mandatory	
IdentifyingNumber		Mandatory	Maps to IMA_PHBA_PROPERTIES, serialNumber.
Vendor		Mandatory	Maps to IMA_PHBA_PROPERTIES, vendor.
Version		Mandatory	Maps to IMA_PHBA_PROPERTIES, hardwareVersion.

9.5.15 CIM_ProductPhysicalComponent

Created By: Static

Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 92 describes class CIM_ProductPhysicalComponent.

Table 92 - SMI Referenced Properties/Methods for CIM_ProductPhysicalComponent

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	
PartComponent		Mandatory	

9.5.16 CIM_Realizes

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 93 describes class CIM_Realizes.

Table 93 - SMI Referenced Properties/Methods for CIM_Realizes

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

9.5.17 CIM_SAPAvailableForElement

Requirement: Mandatory

Table 94 describes class CIM_SAPAvailableForElement.

Table 94 - SMI Referenced Properties/Methods for CIM_SAPAvailableForElement

Properties	Flags	Requirement	Description & Notes
AvailableSAP		Mandatory	
ManagedElement		Mandatory	

9.5.18 CIM_SCSIProtocolController

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 95 describes class CIM_SCSIProtocolController.

Table 95 - SMI Referenced Properties/Methods for CIM_SCSIProtocolController

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	
ElementName		Mandatory	iSCSI Alias.
Name	CD	Mandatory	Maps to IMA_NODE_PROPERTIES, name.
NameFormat		Mandatory	

9.5.19 CIM_SoftwareIdentity

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 96 describes class CIM_SoftwareIdentity.

Table 96 - SMI Referenced Properties/Methods for CIM_SoftwareIdentity

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
VersionString		Mandatory	Maps to IMA_PHBA_PROPERTIES, driverVersion/firmwareVersion/ optionRomVersion as per the Classifications property.
Manufacturer		Mandatory	Maps to IMA_PHBA_PROPERTIES.vendor.
Classifications		Mandatory	Either 'Driver', 'Firmware', or 'BIOS/FCode' (2, 10, or 11).

9.5.20 CIM_SystemDevice (to EthernetPort)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 97 describes class CIM_SystemDevice (to EthernetPort).

Table 97 - SMI Referenced Properties/Methods for CIM_SystemDevice (to EthernetPort)

Properties	Flags	Requirement	Description & Notes
PartComponent		Mandatory	
GroupComponent		Mandatory	

9.5.21 CIM_SystemDevice (to PortController)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 98 describes class CIM_SystemDevice (to PortController).

Table 98 - SMI Referenced Properties/Methods for CIM_SystemDevice (to PortController)

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	
PartComponent		Mandatory	

9.5.22 CIM_SystemDevice (to ProtocolController)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 99 describes class CIM_SystemDevice (to ProtocolController).

Table 99 - SMI Referenced Properties/Methods for CIM_SystemDevice (to ProtocolController)

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	
PartComponent		Mandatory	

9.5.23 CIM_iSCSIConnection

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 100 describes class CIM_iSCSIConnection.

Table 100 - SMI Referenced Properties/Methods for CIM_iSCSIConnection

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
ConnectionID		Mandatory	
MaxReceiveDataSegmentLength		Mandatory	Maps to IMA_GetMaxRecvDataSegmentLengthProperties, IMA_SetMaxRecvDataSegmentLength.
MaxTransmitDataSegmentLength		Mandatory	
HeaderDigestMethod		Mandatory	

Table 100 - SMI Referenced Properties/Methods for CIM_iSCSIConnection

Properties	Flags	Requirement	Description & Notes
OtherHeaderDigestMethod		Optional	
DataDigestMethod		Mandatory	
OtherDataDigestMethod		Optional	
ReceivingMarkers		Mandatory	
SendingMarkers		Mandatory	
ActiveiSCSIVersion		Mandatory	
AuthenticationMethodUsed		Mandatory	Maps to IMA_GetInUseInitiatorAuthMethods.
MutualAuthentication		Mandatory	

9.5.24 CIM_iSCSISession

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 101 describes class CIM_iSCSISession.

Table 101 - SMI Referenced Properties/Methods for CIM_iSCSISession

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
Directionality		Mandatory	
SessionType		Mandatory	
TSIH		Mandatory	
EndPointName		Mandatory	Maps to IMA_TARGET_PROPERTIES, name.
CurrentConnections		Mandatory	
InitialR2T		Mandatory	Maps to IMA_GetInitialR2TProperties, IMA_SetInitialR2T.
ImmediateData		Mandatory	Maps to IMA_GetImmediateDataProperties, IMA_SetImmediateData.
MaxOutstandingR2T		Mandatory	Maps to IMA_GetMaxOutstandingR2TProperties, IMA_SetMaxOutstandingR2T.
MaxUnsolicitedFirstDataBurstLength		Mandatory	Maps to IMA_GetMaxFirstBurstLengthProperties, IMA_SetMaxFirstBurstLength.
MaxDataBurstLength		Mandatory	Maps to IMA_GetMaxBurstLengthProperties, IMA_SetMaxBurstLength.
DataSequenceInOrder		Mandatory	Maps to IMA_GetDataSequenceInOrderProperties, IMA_SetDataSequenceInOrder.
DataPDUInOrder		Mandatory	Maps to IMA_GetDataPDUInOrderProperties, IMA_SetDataPDUInOrder.
ErrorRecoveryLevel		Mandatory	Maps to IMA_GetErrorRecoveryLevelProperties, IMA_SetErrorRecoveryLevel.
MaxConnectionsPerSession		Mandatory	Maps to IMA_GetMaxConnectionsProperties, IMA_SetMaxConnections.
DefaultTimeToWait		Mandatory	Maps to IMA_GetDefaultTime2WaitProperties, IMA_SetDefaultTime2Wait.
DefaultTimeToRetain		Mandatory	Maps to IMA_GetDefaultTime2RetainProperties, IMA_SetDefaultTime2Retain.

EXPERIMENTAL

EXPERIMENTAL

10 SCSI Multipath Management Profile

10.1 Description

10.1.1 Synopsis

Profile Name: SCSI Multipath Management

Version: 1.6.0

Organization: SNIA

Central Class: SCSIPathConfigurationService

Scoping Class: Base Server ComputerSystem

Related Profiles: Not defined in this standard:

10.1.2 Overview

Multipath access to SCSI devices is handled in a similar way on many operating systems. As viewed from host adapters, each combination of host adapter (initiator) port, target device port, and logical unit appears to be a separate logical unit. For example, each path to a multipath device appears to be a separate device. Multipath drivers aggregate these into a single device that acts to storage applications like a single path device, but provides administrative interfaces for load balancing and failback.

Host Discovered Resources incorporates multipath logic as part of the mapping from logical (operating system) resources to hardware resources. If the discovered block storage has a single path, then LogicalIdentity associates the discovered StorageVolume instance with the OS/Partition StorageExtent/ LogicalDisk representing the underlying volume. The subclass of StorageExtent follows the extent naming conventions described in 7.1.3.

The rest of the examples in this section use LogicalDisks since multipath disk arrays are more common, but the same approach can be extended to other storage types. For example, a TapeDrive can model multipath access to a tape drive.

MultipathConfigurationCapabilities allows clients to determine which features and capabilities are exposed. SCSIPathConfigurationService may provide methods for management load balancing and failback. A system may have multiple multipath drivers with different capabilities and interfaces – each

driver is modeled with a separate instance of MultipathConfigurationCapabilities and SCSIPathConfigurationService, as illustrated in Figure 37.

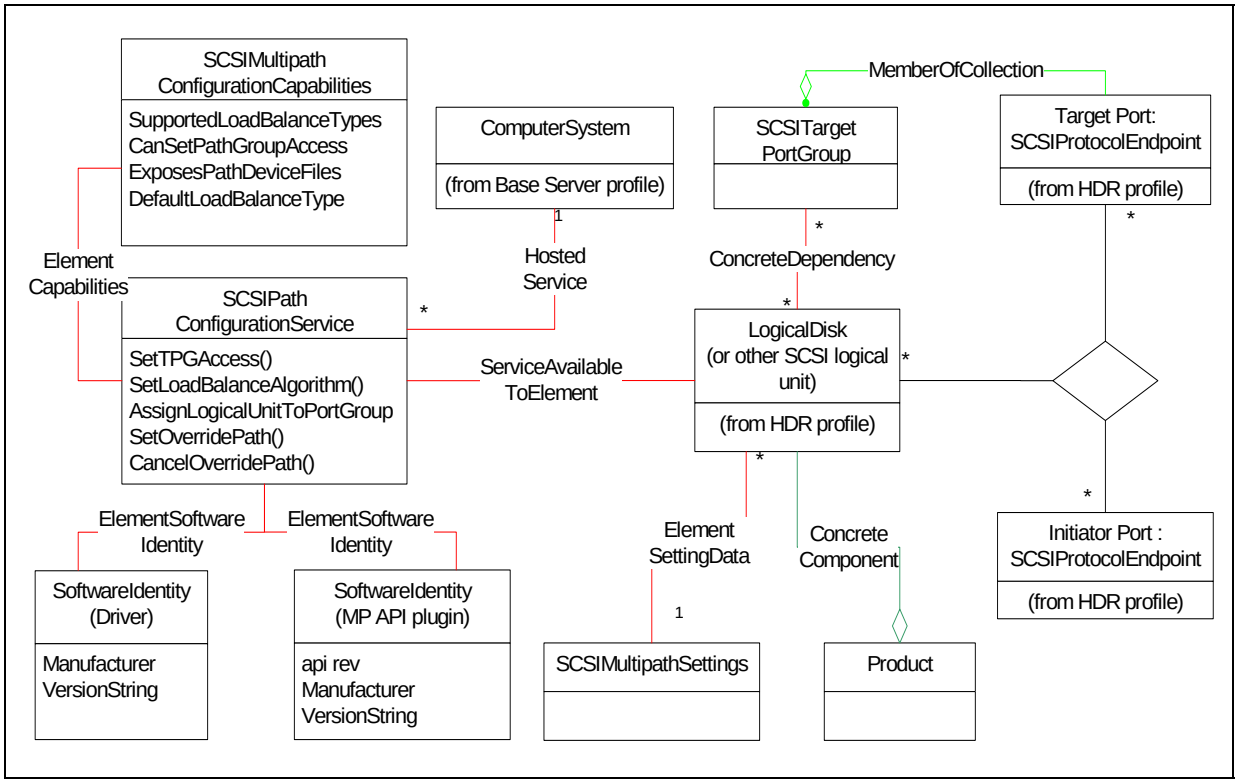

Figure 37 - Multipath Management Class Diagram

All references to ComputerSystem in the SCSI Multipath Management Profile implies a single instance for a customer server or storage system as defined in the Base Server Profile. See Annex A: (Informative) Host Profile Deployment Guidelines, 1.6.0 Rev 3 for information on the use of host profiles with Base Server profile.

Figure 38 shows the relationship of target and initiator ports (SCSIProtocolEndpoint instances) and a disk

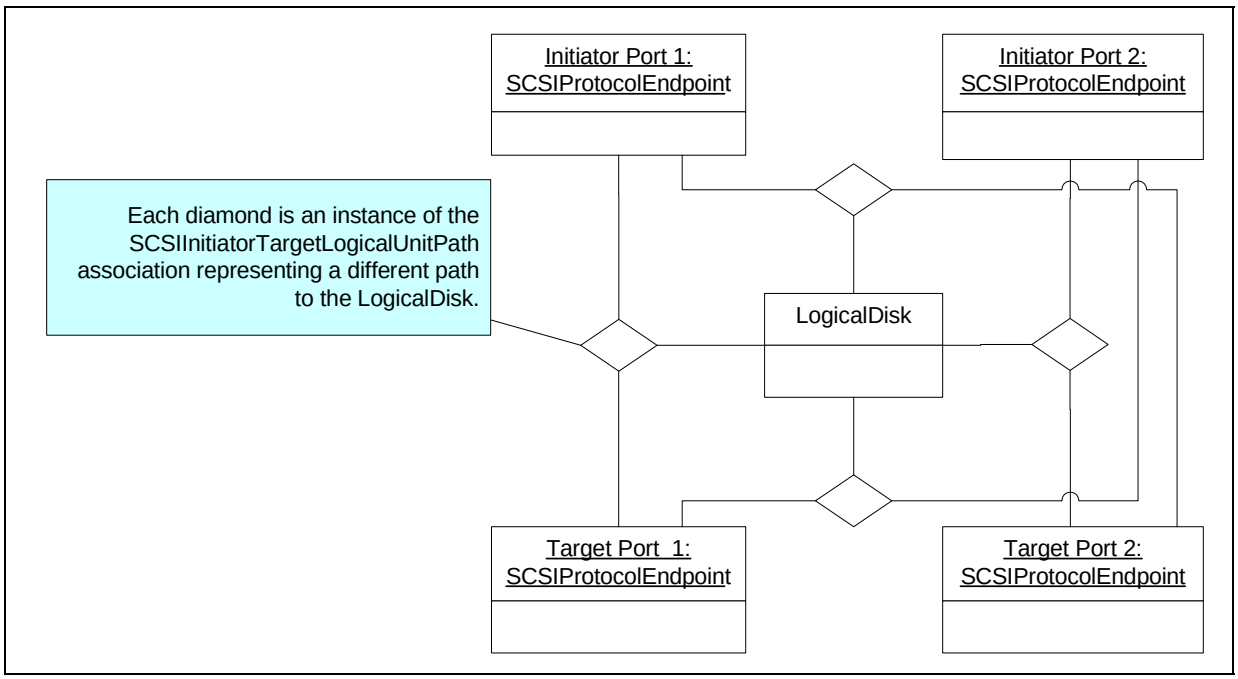

Figure 38 - Four Path Instance Diagram

(LogicalDisk) with four paths. SCSIInitiatorTargetLogicalUnitPath instances represent each path and associate each permutation of initiator SCSIProtocolEndpoint, target SCSIProtocolEndpoint, and the LogicalDisk.

10.1.3 Asymmetric Multipath Target Devices

Some devices implement asymmetric multipath access, i.e., in non-failover mode, each LUN is only available through certain target ports, but can be accessed through other ports during failover. The SMI-S model uses the SPC-3 interface for asymmetric access. This model has target port groups – collections of target ports sharing a common access state for a group of logical units. Mutipath drivers for asymmetric access devices optionally provide an interface to “failback” after a failover condition has been corrected. The SMI-S interface follows the SPC-3 interface; the caller shall specify the desired access state for each target port group (TargetPortGroup). This interface is the SetTPGAccess method of SCSIPathConfigurationService. Driver support for this method (and other methods and capabilities) is indicated by properties of MultipathConfigurationCapabilities.

In the past, devices exposed vendor-specific SCSI multipath interfaces. As such, drivers with device-specific logic were shipped with target devices, logical volume managers, and HBAs. The SPC-3 has been enhanced to allow more interoperability and operating systems are including multipath support for any target that complies with the standards. However, there are still cases where a single customer host includes multiple multipath drivers, each with different capabilities and interfaces. And a single target device may be connected in such a way that multiple multipath drivers are involved at multiple places in the driver stack.

The SNIA Multipath Management API provides an interoperable interface to multipath driver features. Each multipath driver includes a corresponding plug-in for the multipath API. The SNIA Multipath Management Profile utilizes the Multipath API to interface to each multipath driver and provide all the associations from the discovered hardware resources to the consumable operating system resources.

The instrumentation shall instantiate SCSIInitiatorTargetLogicalUnitPath instances representing each path to SCSI logical units (LogicalDevice subclasses) attached to the hosting system.

The instrumentation shall instantiate at least one instance of SCSIMultipathConfigurationCapabilities for each multipath API plug-in registered on the system.

If the multipath API plug-ins provide support for interfaces to change load balancing and force failover, the instrumentation should support these methods.

10.2 Health and Fault Management Considerations

This profile specifies logical paths between elements (ports and logical units). The health and fault management information for these elements is specified in the profiles for those elements - for example, port profiles.

10.3 Methods of the Profile

All methods are part of SCSIPathConfigurationService and are optional.

10.3.1 SCSIPathConfigurationService.SetTPGAccess

This method allows a client to manually failover or failback. The parameters are:

- LogicalDevice - A reference to an instance of a subclass of LogicalDevice representing a SCSI logical unit where the command shall be sent.
- TargetPortGroups - Array of references to instances of SCSITargetPortGroup. All the referenced TargetPortGroup instances shall be part of the same target device
- AccessStates[] - An array of desired access states. Each access state in this array is the desired access state for the SCSITargetPortGroup in the corresponding entry in the TargetPortGroups parameter. The Active value is not part of SPC-3; it is a convenience for clients that are not sure whether to specify Active/Optimized or Active/Non-optimized. The instrumentation selects a value based on historic information, knowledge of the target configuration, or trial and error. Note that SCSITargetPortGroup.AccessState includes the value 'Transitioning' that is excluded here - a caller cannot request transitioning, though it may be reported by a target device.

10.3.2 SCSIPathConfigurationService.SetLoadBalanceAlgorithm

This method requests that the target change the load balance algorithm for the referenced LogicalDevice instance. The parameters are

- LogicalDevice - a reference to an instance of a subclass of LogicalDevice representing a SCSI logical unit.
- LoadBalanceAlgorithm - The desired load balance algorithm - possible values are “Unknown”, “Other”, “No Load Balancing”, “Round Robin”, “Least Blocks”, “Least IO”, or “Product Specific”
- OtherLoadBalanceAlgorithm - When LoadBalanceAlgorithm is 'Other', this parameter specifies a description of the load balancing algorithm. When LoadBalanceAlgorithm is 'Product Specific', this property provides a string specifying the vendor/product/version of the ManagedElement.

10.3.3 SCSIPathConfigurationService.AssignLogicalUnitToPortGroup

This method allows an administrator to assign a logical unit to a target port group. Each LU is typically associated with two target port groups, one in active state and one in standby state. The result of this method is that the LU associations change to a pair of target port groups. Only valid if the target device supports asymmetric access state and SCSIMultipathConfigurationCapabilities SupportsLuAssignment is set. The parameters are:

- LogicalDevice - a reference to an instance of a subclass of LogicalDevice representing a SCSI logical unit.
- TargetPortGroup - A reference to a target port group. The Target Port Group should be in an active state.

10.3.4 SCSIPathConfigurationService.SetOverridePath

This method allows an administrator to temporarily disable load balancing for a specific logical unit. The path specified as a parameter shall have its AdministrativeOverride property set to 'Overriding' and all I/O to the logical unit shall be directed to this path. All other paths to this logical unit shall have AdministrativeOverride set to 'Overridden'. There is one parameter:

- Path - A reference to a SCSIInitiatorTargetLogicalUnitPath.

10.3.5 SCSIPathConfigurationService.CancelOverridePath

This method clears an override path as set in SetOverridePath and load balancing is enabled. All paths to the logical unit specified as a parameter shall have AdministrativeOverride property set to 'No override in effect'. There is one parameter:

- Path - A reference to a SCSIInitiatorTargetLogicalUnitPath.

After an override is canceled, the previous load balance algorithm should be restored.

10.4 Use Cases

Not defined in this version of the standard.

10.5 CIM Elements

Table 102 describes the CIM elements for SCSI Multipath Management.

Table 102 - CIM Elements for SCSI Multipath Management

Element Name	Requirement	Description
10.5.1 CIM_ConcreteComponent	Mandatory	Associates Product and LogicalDevice subclass instances representing SCSI logical units.
10.5.2 CIM_ConcreteDependency	Mandatory	Associates SCSITargetPortGroup to LogicalDevice subclass instances representing SCSI logical units.
10.5.3 CIM_ElementCapabilities	Mandatory	Associates SCSIMultipathConfigurationCapabilities and SCSIPathConfigurationService.
10.5.4 CIM_ElementConformsToProfile (SCSIPathConfigurationService to SCSI Multipath Management RegisteredProfile)	Mandatory	Ties the SCSIPathConfigurationService to the registered profile for SCSI Multipath Management.
10.5.5 CIM_ElementSettingData	Mandatory	Associates SCSIMultipathSettings and LogicalDevice subclass instances representing SCSI logical units.
10.5.6 CIM_ElementSoftwareIdentity (Driver)	Mandatory	Associates SCSIPathConfigurationService and the Driver SoftwareIdentity instance.
10.5.7 CIM_ElementSoftwareIdentity (MP API Plugin)	Mandatory	Associates SCSIPathConfigurationService and the MP API Plugin SoftwareIdentity instance.
10.5.8 CIM_HostedService	Mandatory	Associates SCSIPathConfigurationService and the ComputerSystem from Base Server.
10.5.9 CIM_MemberOfCollection	Mandatory	Associates SCSITargetPortGroup and SCSIProtocolEndpoint.
10.5.10 CIM_Product	Mandatory	Models a Product as defined in MP API.
10.5.11 CIM_SCSIMultipathConfigurationCapabilities	Mandatory	A class derived from Capabilities that models the capabilities of a multipath driver.
10.5.12 CIM_SCSIMultipathSettings	Mandatory	Settings related to management of multiple paths to SCSI devices.

Table 102 - CIM Elements for SCSI Multipath Management

Element Name	Requirement	Description
10.5.13 CIM_SCSIPathConfigurationService	Mandatory	A class providing methods related to management of multiple paths to SCSI devices.
10.5.14 CIM_SCSITargetPortGroup	Mandatory	Models SCSI Target Port Groups.
10.5.15 CIM_ServiceAvailableToElement	Mandatory	Associates SCSIPathConfigurationService with instances representing SCSI logical units.
10.5.16 CIM_SoftwareIdentity (Driver)	Mandatory	Driver.
10.5.17 CIM_SoftwareIdentity (MP API Plugin)	Mandatory	MP API Plugin.

10.5.1 CIM_ConcreteComponent

Associates Product and LogicalDevice subclass instances representing SCSI logical units.

Requirement: Mandatory

Table 103 describes class CIM_ConcreteComponent.

Table 103 - SMI Referenced Properties/Methods for CIM_ConcreteComponent

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	Reference to Product.
PartComponent		Mandatory	Reference to LogicalDevice subclass representing a SCSI logical unit.

10.5.2 CIM_ConcreteDependency

Associates SCSITargetPortGroup to LogicalDevice subclass instances representing SCSI logical units.

Requirement: Mandatory

Table 104 describes class CIM_ConcreteDependency.

Table 104 - SMI Referenced Properties/Methods for CIM_ConcreteDependency

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	Reference to LogicalDevice subclass representing a SCSI logical unit.
Dependent		Mandatory	Reference to SCSITargetPortGroup.

10.5.3 CIM_ElementCapabilities

Associates SCSIMultipathConfigurationCapabilities and SCSIPathConfigurationService.

Created By: External
Modified By: External
Deleted By: External
Requirement: Mandatory

Table 105 describes class CIM_ElementCapabilities.

Table 105 - SMI Referenced Properties/Methods for CIM_ElementCapabilities

Properties	Flags	Requirement	Description & Notes
Capabilities		Mandatory	Reference to SCSIMultipathConfigurationCapabilities.
ManagedElement		Mandatory	Reference to SCSIPathConfigurationService.

10.5.4 CIM_ElementConformsToProfile (SCSIPathConfigurationService to SCSI Multipath Management RegisteredProfile)

The CIM_ElementConformsToProfile ties SCSIPathConfigurationService to the registered profile for SCSI Multipath Management.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 106 describes class CIM_ElementConformsToProfile (SCSIPathConfigurationService to SCSI Multipath Management RegisteredProfile).

Table 106 - SMI Referenced Properties/Methods for CIM_ElementConformsToProfile (SCSIPathConfigurationService to SCSI Multipath Management RegisteredProfile)

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	A SCSIPathConfigurationService instance that represents the SCSI Multipath Management.
ConformantStandard		Mandatory	RegisteredProfile instance describing the SCSI Multipath Management profile.

10.5.5 CIM_ElementSettingData

Associates SCSIMultipathSettings and LogicalDevice subclass instances representing SCSI logical units.

Requirement: Mandatory

Table 107 describes class CIM_ElementSettingData.

Table 107 - SMI Referenced Properties/Methods for CIM_ElementSettingData

Properties	Flags	Requirement	Description & Notes
SettingData		Mandatory	Reference to SCSIMultipathSettings.
ManagedElement		Mandatory	Reference to LogicalDevice subclass representing a SCSI logical unit.

10.5.6 CIM_ElementSoftwareIdentity (Driver)

Associates SCSIPathConfigurationService and the Driver SoftwareIdentity instance.

Created By: External

Modified By: External

Deleted By: External

Requirement: Mandatory

Table 108 describes class CIM_ElementSoftwareIdentity (Driver).

Table 108 - SMI Referenced Properties/Methods for CIM_ElementSoftwareIdentity (Driver)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	Reference to Driver SoftwareIdentity.
Dependent		Mandatory	Reference to the SCSIPathConfigurationService.

10.5.7 CIM_ElementSoftwareIdentity (MP API Plugin)

Associates SCSIPathConfigurationService and the MP API Plugin SoftwareIdentity instance.

Created By: External
Modified By: External
Deleted By: External
Requirement: Mandatory

Table 109 describes class CIM_ElementSoftwareIdentity (MP API Plugin).

Table 109 - SMI Referenced Properties/Methods for CIM_ElementSoftwareIdentity (MP API Plugin)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	Reference to MP API Plugin SoftwareIdentity.
Dependent		Mandatory	Reference to the SCSIPathConfigurationService.

10.5.8 CIM_HostedService

Associates SCSIPathConfigurationService and the ComputerSystem from Base Server.

Created By: External
Modified By: External
Deleted By: External
Requirement: Mandatory

Table 110 describes class CIM_HostedService.

Table 110 - SMI Referenced Properties/Methods for CIM_HostedService

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	Reference to ComputerSystem in Base Server.
Dependent		Mandatory	Reference to SCSIPathConfigurationService.

10.5.9 CIM_MemberOfCollection

Associates SCSITargetPortGroup and SCSIProtocolEndpoint representing a target port.

Requirement: Mandatory

Table 111 describes class CIM_MemberOfCollection.

Table 111 - SMI Referenced Properties/Methods for CIM_MemberOfCollection

Properties	Flags	Requirement	Description & Notes
Collection		Mandatory	Reference to SCSITargetPortGroup.
Member		Mandatory	Reference to a target SCSIProtocolEndpoint.

10.5.10 CIM_Product

Models a Product as defined in MP API.

Requirement: Mandatory

Table 112 describes class CIM_Product.

Table 112 - SMI Referenced Properties/Methods for CIM_Product

Properties	Flags	Requirement	Description & Notes
ElementName		Mandatory	
Name		Mandatory	
IdentifyingNumber		Mandatory	
Vendor		Mandatory	
Version		Mandatory	

10.5.11 CIM_SCSIMultipathConfigurationCapabilities

A class derived from Capabilities that models the capabilities of a multipath driver.

Created By: External
Modified By: External
Deleted By: External
Requirement: Mandatory

Table 113 describes class CIM_SCSIMultipathConfigurationCapabilities.

Table 113 - SMI Referenced Properties/Methods for CIM_SCSIMultipathConfigurationCapabilities

Properties	Flags	Requirement	Description & Notes
ElementName		Mandatory	Unique ID for the capabilities instance.
SupportedLoadBalanceTypes		Mandatory	
CanSetTPGAccess		Mandatory	
ExposesPathDeviceFiles		Mandatory	
DefaultLoadBalanceType		Mandatory	

10.5.12CIM_SCSIMultipathSettings

A class derived from CIM_SettingData describing settings related to management of multiple paths to SCSI devices. It is associated to one of more instances of subclasses of LogicalDevice that represent SCSI logical units.

Requirement: Mandatory

Table 114 describes class CIM_SCSIMultipathSettings.

Table 114 - SMI Referenced Properties/Methods for CIM_SCSIMultipathSettings

Properties	Flags	Requirement	Description & Notes
Asymmetric		Mandatory	Indicates whether the associated logical unit has asymmetric multipath access.
CurrentLoadBalanceType		Mandatory	
OtherCurrentLoadBalanceType		Conditional	Conditional requirement: support for CurrentLoadBalanceType of \Other\'.'
AutoFailbackEnabled		Mandatory	
PollingRateMax		Optional	
CurrentPollingRate		Optional	

10.5.13CIM_SCSIPathConfigurationService

A class derived from CIM_Service providing methods related to management of multiple paths to SCSI devices.

Created By: External

Modified By: External

Deleted By: External

Requirement: Mandatory

Table 115 describes class CIM_SCSIPathConfigurationService.

Table 115 - SMI Referenced Properties/Methods for CIM_SCSIPathConfigurationService

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	The scoping System CreationClassName.
SystemName		Mandatory	The scoping System Name.
CreationClassName		Mandatory	The name of the concrete subclass.
Name		Mandatory	Uniquely identifies the Service.
SetTPGAccess()		Conditional	Conditional requirement: support for SetTPGAccess method.
SetLoadBalanceAlgorithm()		Optional	
AssignLogicalUnitToPortGroup()		Optional	
SetOverridePath()		Conditional	Conditional requirement: support for override path methods.
CancelOverridePath()		Conditional	Conditional requirement: support for override path methods.

10.5.14CIM_SCSITargetPortGroup

A class derived from SystemSpecificCollection that models SCSI Target Port Groups. SCSITargetPortGroup is part of the model for devices with asymmetric access to logical units - access is optimized for a subset of target ports. SCSITargetPortGroup is aggregated to SCSIProtocolEndpoints that expose a common access state.

Created By: External

Modified By: External

Deleted By: External

Requirement: Mandatory

Table 116 describes class CIM_SCSITargetPortGroup.

Table 116 - SMI Referenced Properties/Methods for CIM_SCSITargetPortGroup

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	Opaque and unique identifier.
ElementName		Optional	A user-friendly name.
AccessState		Optional	Access to all associated logical units through all aggregated ports share this access state.
SupportsLuAssignment		Mandatory	Indicates whether the implementation provides an interface to reassign logical units to target port groups.
ExplicitFailover		Mandatory	Indicatges the implementation provides an interface to explicitly request activation of a TPG.
Preferred		Optional	Indicates that access to the associated logical units through ports in this TPG is preferred over access through other ports.
Identifier		Optional	An integer identifier for the TPG.

10.5.15CIM_ServiceAvailableToElement

Associates SCSIPathConfigurationService with instances representing SCSI logical units.

Created By: External

Modified By: External

Deleted By: External

Requirement: Mandatory

Table 117 describes class CIM_ServiceAvailableToElement.

Table 117 - SMI Referenced Properties/Methods for CIM_ServiceAvailableToElement

Properties	Flags	Requirement	Description & Notes
ServiceProvided		Mandatory	Reference to SCSIPathConfigurationService.
UserOfService		Mandatory	Reference to LogicalDevice subclass representing a SCSI logical unit.

10.5.16CIM_SoftwareIdentity (Driver)

SoftwareIdentity representing the Driver software.

Created By: External
Modified By: External
Deleted By: External
Requirement: Mandatory

Table 118 describes class CIM_SoftwareIdentity (Driver).

Table 118 - SMI Referenced Properties/Methods for CIM_SoftwareIdentity (Driver)

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	The name used to identify this SoftwareIdentity.
VersionString		Mandatory	Software Version should be in the form [Major], [Minor].[Revision] or [Major].[Minor][letter][revision].
Manufacturer		Mandatory	Manufacturer of this Software.
Classifications		Mandatory	Shall be 2 (Driver).

10.5.17CIM_SoftwareIdentity (MP API Plugin)

SoftwareIdentity representing the MP API plugin software.

Created By: External
Modified By: External
Deleted By: External
Requirement: Mandatory

Table 119 describes class CIM_SoftwareIdentity (MP API Plugin).

Table 119 - SMI Referenced Properties/Methods for CIM_SoftwareIdentity (MP API Plugin)

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	The name used to identify this SoftwareIdentity.
VersionString		Mandatory	Software Version should be in the form [Major], [Minor].[Revision] or [Major].[Minor][letter][revision].
Manufacturer		Mandatory	Manufacturer of this Software.
Classifications		Mandatory	Shall be 1 (Other).
ClassificationDescriptions		Mandatory	Shall be 'MP API Plugin'.

EXPERIMENTAL

DEPRECATED

11 SB Multipath Management Profile

See the last version of this profile in SMI-S Version 1.6.1. Revison 5.

DEPRECATED

EXPERIMENTAL

12 Memory Configuration Profile

12.1 Synopsis

Profile Name: Memory Configuration

Version: 1.0.0

Organization: SNIA

Central Class: MemoryConfigurationService

Scoping Class: ComputerSystem

Related Profiles: Table 120 describes the related profiles for the Memory Configuration Profile:

Table 120 - Related Profiles

Profile Name	Organization	Version	Requirement	Description
Profile Registration	DMTF	1.1.0	Mandatory	
Multi-type System Memory	DMTF	1.0.0a	Mandatory	DMTF DSP1071

12.2 Description

Memory subsystems which offer logical configuration options are candidates for management using the Memory Configuration Profile. This profile describes memory resource pools which serve to group memory resources with like characteristics. For example, a system containing both volatile memory modules and battery backed persistent memory modules might group the logical capacity on these memory modules into two separate pools. A configuration service provides allocation, deallocation and assignment related operations for a pool. An allocation operation reserves some amount of capacity from a pool and assigns it to the system. A deallocation returns allocated capacity to the pool to be reused. Assignment indicates that access to an allocated memory resource is to be restricted in some way. Consider the case of memory being assigned as a cache; access to the cache would likely be restricted to hardware or software performing the caching operation and not generally available to the system. Finally the profile describes various means for a system to advertise the memory configuration options it supports. Availability of memory configuration features is contingent upon a number of factors including the capabilities of the system memory modules, the memory controller features and the features of the BIOS, operating system or other memory management software.

12.2.1 Class Diagram

Figure 39 shows the Memory Configuration Profile class hierarchy. For simplicity, the prefix CIM_ has been removed from the names of the classes.

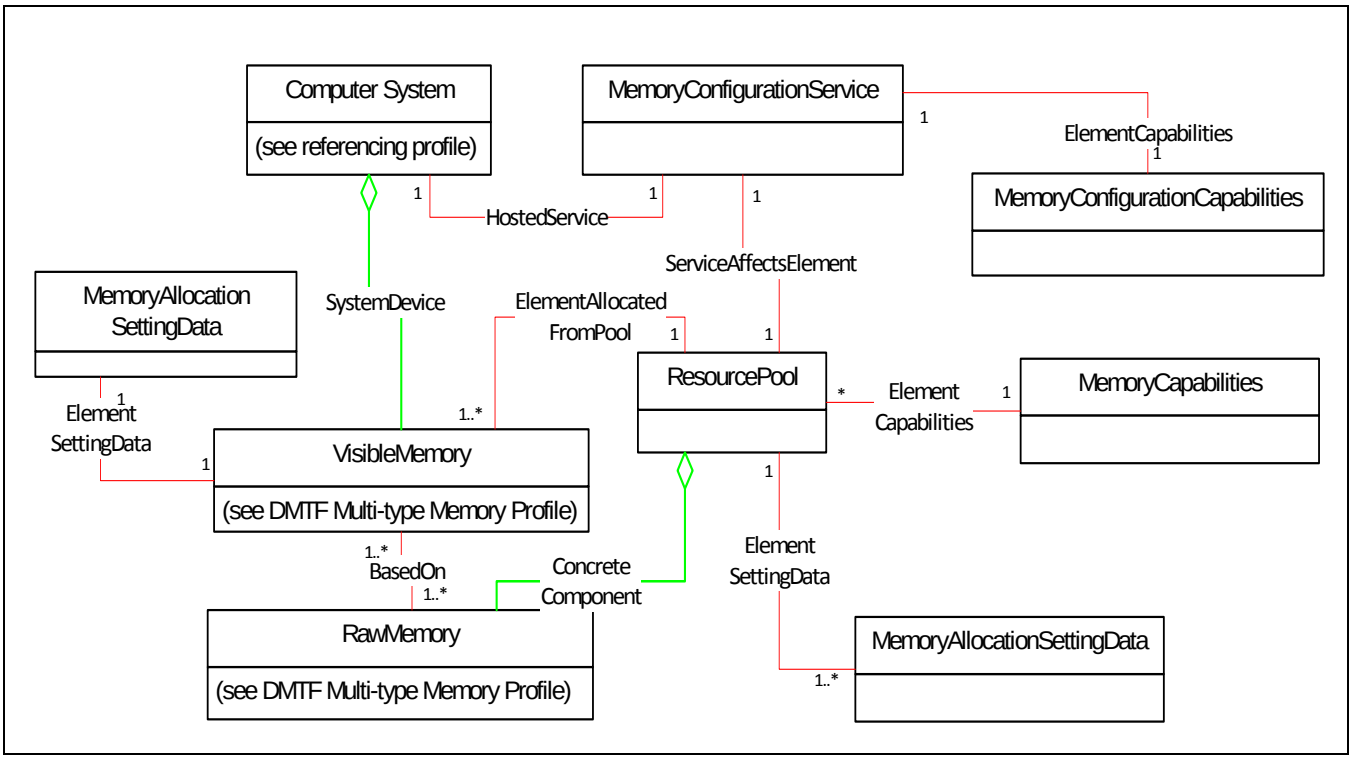

Figure 39 - Memory Configuration Class Diagram

12.2.2 MemoryConfigurationService

The MemoryConfigurationService is the central class of the profile. It models the system's support for modifying a system's logical memory configuration.

12.2.3 ResourcePool(Memory)

ResourcePool instances aggregate memory capacity with like capabilities.

12.2.4 MemoryConfigurationCapabilities & MemoryCapabilities

The profile supports run-time detection of memory features. MemoryConfigurationCapabilities advertises the system's support for modification of the logical memory configuration. MemoryCapabilities indicates the capabilities of the memory modules and other system components.

12.2.5 MemoryAllocationSettingData

MemoryAllocationSettingData records the parameters used during an allocation or assignment operation.

12.2.6 Registered Profile

Figure 39 includes CIM_RegisteredProfile which is expected to be implemented per the Profile Registration Profile (*Storage Management Technical Specification, Part 3 Common Profiles, 1.7.0 Rev 5* Clause 36 Profile Registration Profile).

12.2.7 DMTF: Multi-type System Memory Profile

RawMemory and VisibleMemory classes are described in detail in the DMTF Multi-type System Memory Profile. A RawMemory instance represents the unconfigured (primordial) capacity of a given memory module. RawMemory instances are the constituents of a (memory) ResourcePool. VisibleMemory instances model memory capacity that has been allocated (concrete) from a (memory) ResourcePool.

12.3 Implementation

12.3.1 CIM_ResourcePool (memory)

Implementations of this profile shall contain at least one instance of CIM_ResourcePool. CIM_ResourcePool pools are primordial in nature, containing unprepared or unassigned capacity. It is presumed that CIM_RawMemory instances that together constitute a given CIM_ResourcePool are compatible such that they can be manipulated in aggregate by a MemoryConfigurationService. This profile does not mandate any specific level of compatibility between members of a ResourcePool.

12.3.2 Determining Pool Capacity

The capacity of a (memory) ResourcePool instance shall be the total unconfigured capacity of the pool's constituent CIM_RawMemory instances. Any capacity gains or losses due to device usage/configuration (e.g. replication, metadata) shall not be included in the capacity reported for the pool.

12.3.3 CIM_MemoryCapabilities

Implementations of this profile shall include at least one instance of CIM_MemoryCapabilities. Each instance shall describe the capabilities of one or more instances of CIM_ResourcePools as determined by the CIM_ElementCapabilities association.

12.3.4 CIM_MemoryConfigurationService

Implementations of this profile shall contain at least one instance of CIM_MemoryConfigurationService. Instances of this class shall conditionally support allocation, deallocation and assignment of memory resources from associated (see 12.3.8 CIM_ServiceAffectsElement) CIM_ResourcePools. Availability of any given extrinsic shall be determined by examining the CIM_MemoryConfigurationCapabilities instance to which it is associated (CIM_ElementCapabilities).

12.3.5 CIM_MemoryConfigurationCapabilities

Implementations of this profile shall contain at least one instance of CIM_MemoryConfigurationCapabilities. Each instance shall be associated with one or more instances of CIM_MemoryConfigurationService via a CIM_ElementCapabilities association.

12.3.6 CIM_MemoryAllocationSettingData

The availability of CIM_MemoryAllocationSettingData instances is conditional. Whenever CIM_MemoryConfigurationService extrinsics are used to allocate a CIM_VisibleMemory instance, a companion instance of CIM_MemoryAllocationSettingData shall be available. The companion instance shall be the goal setting used in the AllocateFromPool extrinsic method (see 12.4.1.1 CIM_MemoryConfigurationService.AllocateFromPool()).

When an allocation operation results in a staged request that requires some triggering event (e.g. a reboot) an instance of CIM_MemoryAllocationSettingData shall exist to represent the staged request. These instances are associated with the impacted CIM_ResourcePool rather than a CIM_VisibleMemory instance.

12.3.6.1 Specify Allocation Properties

Workload performance for a given memory region is influenced by the properties specified in the CIM_MemoryAllocationSettingData instance at memory region creation time.

12.3.7 CIM_ElementCapabilities

CIM_ElementCapabilities shall associate the CIM_MemoryConfigurationCapabilities to CIM_MemoryConfigurationService. Capabilities so associated shall describe the extrinsic method support offered by the service instance. The characteristics attribute of CIM_ElementCapabilities is not used.

CIM_ElementCapabilities shall associate CIM_MemoryCapabilities to CIM_ResourcePoolCapabilities so associated describe the memory module and related platform memory features that determine configurability. The characteristics attribute of CIM_ElementCapabilities is not used.

12.3.8 CIM_ServiceAffectsElement

CIM_ServiceAffectsElement associations shall relate CIM_MemoryConfigurationService instances with the CIM_ResourcePool(s) they are able to operate on.

12.3.9 CIM_ConcreteComponent

CIM_RawMemory instances pooled together for management purposes are represented by CIM_ResourcePool instances. CIM_RawMemory instances shall be associated with a CIM_ResourcePool instance by the CIM_ConcreteComponent association.

12.3.10 CIM_ElementAllocatedFromPool

CIM_VisibleMemory instances are created from resources managed in aggregate as a CIM_ResourcePool. CIM_VisibleMemory instances shall be associated to the CIM_ResourcePool from which they draw their resources by a CIM_ElementAllocatedFromPool association.

12.3.11 CIM_HostedService

An instance of the CIM_HostedService shall associate a CIM_MemoryConfigurationService with the CIM_ComputerSystem where its underlying configuration (software and hardware) facilities are located.

12.3.12 CIM_ElementSettingData

CIM_ElementSettingData represents the relationship between an allocated memory extent and the settings supplied during its creation. An instance of CIM_ElementSettingData shall exist between CIM_VisibleMemory and an associated CIM_MemoryAllocationSettingData instance. For these instances IsCurrent shall be set to 1 “Is Current”. IsNext shall be set to 2 “Is Not Next”. The value IsDefault is determined by the implementation.

CIM_ElementSettingData shall also be used to associate staged CIM_MemoryAllocationSetting instances to the ResourcePool that hosts the resources impacted by the CIM_MemoryAllocationSetting instance. Once the allocation transitions from a staged “request” to an actual CIM_VisibleMemory instance this association no longer exists. For these instances IsCurrent shall be set to 2 “Is Not Current”. IsNext shall be set to 1 “Is Next”. The value IsDefault is determined by the implementation.

12.3.13 CIM_ElementConformsToProfile

CIM_ElementConformsToProfile shall associate a RegisteredProfile instance representing the MemoryConfigurationService profile to the central class of the profile CIM_MemoryConfigurationService.

12.4 Methods

This clause details the requirements for supporting intrinsic operations and extrinsic methods for the CIM elements defined by this profile.

12.4.1 CIM_MemoryConfigurationService

Implementations of this profile shall support the operations listed in Table 121 for CIM_MemoryConfigurationService Each operation shall be supported as defined in DMTF DSP0200 *CIM Operations over HTTP*.

Table 121 - Operations: CIM_MemoryConfigurationService

Operation	Requirement	Messages
GetInstance	Mandatory	None
Associators	Mandatory	None
AssociatorNames	Mandatory	None
References	Mandatory	None
ReferenceNames	Mandatory	None
EnumerateInstances	Mandatory	None
EnumerateInstanceNames	Mandatory	None

12.4.1.1 CIM_MemoryConfigurationService.AllocateFromPool()

The AllocateFromPool() method is used to configure unprepared, raw memory resources resulting in one or more system visible memory regions. Support for this method is signaled by its inclusion in the SupportedAsynchronousActions attribute in CIM_MemoryConfigurationCapabilities. Return values are listed in Table 122.

Table 122 - CIM_MemoryConfigurationService.AllocateFromPool() Method: Return Code Values

Value	Description
0	Request was successfully executed
1	Request was successfully staged for future execution.
2	Method is not supported in this implementation
2	Unknown
3	Timeout
4	Failed
5	Invalid parameter
4096	Request was successfully staged for future execution
4097	Insufficient Resources
4098	Inconsistent Parameters
4099	Request did not complete in its entirety and partial results could not be undone

AllocateFromPool() parameters are listed in Table 123:

Table 123 - CIM_MemoryConfigurationService.AllocateFromPool() Method: Parameters

Qualifiers	Name	Type	Description
IN, REQ	Pool	CIM_ResourcePool REF	The pool of memory resources from which to allocate the visible memory region.
IN,REQ	Goal[]	string	One or more embedded instances of CIM_MemoryAllocationSetting which describe the allocation request.
OUT	Extents[]	CIM_VisibleMemory REF	If the return code is 0, this value shall contain references to the created VisibleMemory instances. When the request is successful but has been staged for future execution or an error occurs, this parameter is not used.

12.4.1.2 CIM_MemoryConfigurationService.ReturnToPool()

The ReturnToPool() method deletes an instance of CIM_VisibleMemory, returning capacity to the CIM_ResourcePool from which it originated, as shown in Table 124.

Table 124 - CIM_MemoryConfigurationService.ReturnToPool() Method: Return Code Values

Value	Description
0	Request was successfully executed
1	Request was successfully staged for future execution
2	Method is not supported in this implementation
3	Unknown
4	Timeout
5	Failed
4096	Request was successfully staged for future execution
4099	Request did not complete in its entirety and partial results could not be undone

ReturnToPool() parameters are listed in Table 125:

Table 125 - CIM_MemoryConfigurationService.ReturnToPool() Method: Parameters

Qualifiers	Name	Type	Description
IN, REQ	MemoryExtent[]	CIM_VisibleMemory REF	The instance to deallocate

12.4.2 CIM_ResourcePool (memory)

Table 126 lists operations and requirements for CIM_ResourcePool.

Table 126 - Operations: CIM_ResourcePool

Operation	Requirement	Messages
GetInstance	Mandatory	None

Table 126 - Operations: CIM_ResourcePool

Operation	Requirement	Messages
Associators	Mandatory	None
AssociatorNames	Mandatory	None
References	Mandatory	None
ReferenceNames	Mandatory	None
EnumerateInstances	Mandatory	None
EnumerateInstanceNames	Mandatory	None

12.4.3 CIM_MemoryCapabilities

Implementations of this profile shall support the operations listed in Table 127 for the CIM_MemoryCapabilities class.

Table 127 - Operations: CIM_MemoryCapabilities

Operation	Requirement	Messages
GetInstance	Mandatory	None
Associators	Mandatory	None
AssociatorNames	Mandatory	None
References	Mandatory	None
ReferenceNames	Mandatory	None
EnumerateInstances	Mandatory	None
EnumerateInstanceNames	Mandatory	None

12.4.4 CIM_MemoryConfigurationCapabilities

Implementations of this profile shall support the operations listed in Table 128 for the CIM_MemoryConfigurationCapabilities class.

Table 128 - Operations: CIM_MemoryConfigurationCapabilities

Operation	Requirement	Messages
GetInstance	Mandatory	None
Associators	Mandatory	None
AssociatorNames	Mandatory	None
References	Mandatory	None
ReferenceNames	Mandatory	None
EnumerateInstances	Mandatory	None
EnumerateInstanceNames	Mandatory	None

12.4.5 CIM_MemoryAllocationSettingData

Table 129 lists operations and requirements for CIM_MemoryAllocationSettingData.

Table 129 - Operations: CIM_MemoryAllocationSettingData

Operation	Requirement	Messages
GetInstance	Mandatory	None
Associators	Mandatory	None
AssociatorNames	Mandatory	None
References	Mandatory	None
ReferenceNames	Mandatory	None
EnumerateInstances	Mandatory	None
EnumerateInstanceNames	Mandatory	None
DeleteInstance	Optional	

12.4.6 CIM_ElementCapabilities

Table 130 lists operations and requirements for CIM_ElementCapabilities.

Table 130 - Operations: CIM_ElementCapabilities

Operation	Requirement	Messages
GetInstance	Mandatory	None
EnumerateInstances	Mandatory	None
EnumerateInstanceNames	Mandatory	None

12.4.7 CIM_ServiceAffectsElement

Table 131 lists operations and requirements for CIM_ServiceAffectsElement.

Table 131 - Operations: CIM_ServiceAffectsElement

Operation	Requirement	Messages
GetInstance	Mandatory	None
EnumerateInstances	Mandatory	None
EnumerateInstanceNames	Mandatory	None

12.4.8 CIM_ConcreteComponent

Table 132 lists operations and requirements for CIM_ConcreteComponent.

Table 132 - Operations: CIM_ConcreteComponent

Operation	Requirement	Messages
GetInstance	Mandatory	None
EnumerateInstances	Mandatory	None
EnumerateInstanceNames	Mandatory	None

12.4.9 CIM_ElementAllocatedFromPool

Table 133 lists operations and requirements for CIM_ElementAllocatedFromPool

Table 133 - Operations: CIM_ElementAllocatedFromPool

Operation	Requirement	Messages
GetInstance	Mandatory	None
EnumerateInstances	Mandatory	None
EnumerateInstanceNames	Mandatory	None

12.4.10CIM_HostedService

Table 134 lists operations and requirements for CIM_HostedService.

Table 134 - Operations: CIM_HostedService

Operation	Requirement	Messages
GetInstance	Mandatory	None
EnumerateInstances	Mandatory	None
EnumerateInstanceNames	Mandatory	None

12.4.11CIM_ElementSettingData

Table 135 lists operations and requirements for CIM_ElementSettingData.

Table 135 - Operations: CIM_ElementSettingData

Operation	Requirement	Messages
GetInstance	Mandatory	None
EnumerateInstances	Mandatory	None
EnumerateInstanceNames	Mandatory	None

12.4.12CIM_ElementConformsToProfile

Table 136 lists operations and requirements for CIM_ElementConformsToProfile.

Table 136 - Operations: CIM_ElementConformsToProfile

Operation	Requirement	Messages
GetInstance	Mandatory	None
EnumerateInstances	Mandatory	None
EnumerateInstanceNames	Mandatory	None

12.5 Use Cases

This section contains object diagrams and use cases for this profile.

12.5.1 Advertising Profile Conformance

Figure 40 shows how an instance of CIM_RegisteredProfile is used to indicate the presence of a conforming implementation of the Memory Configuration Profile and to identify instances of its central class CIM_MemoryConfigurationService

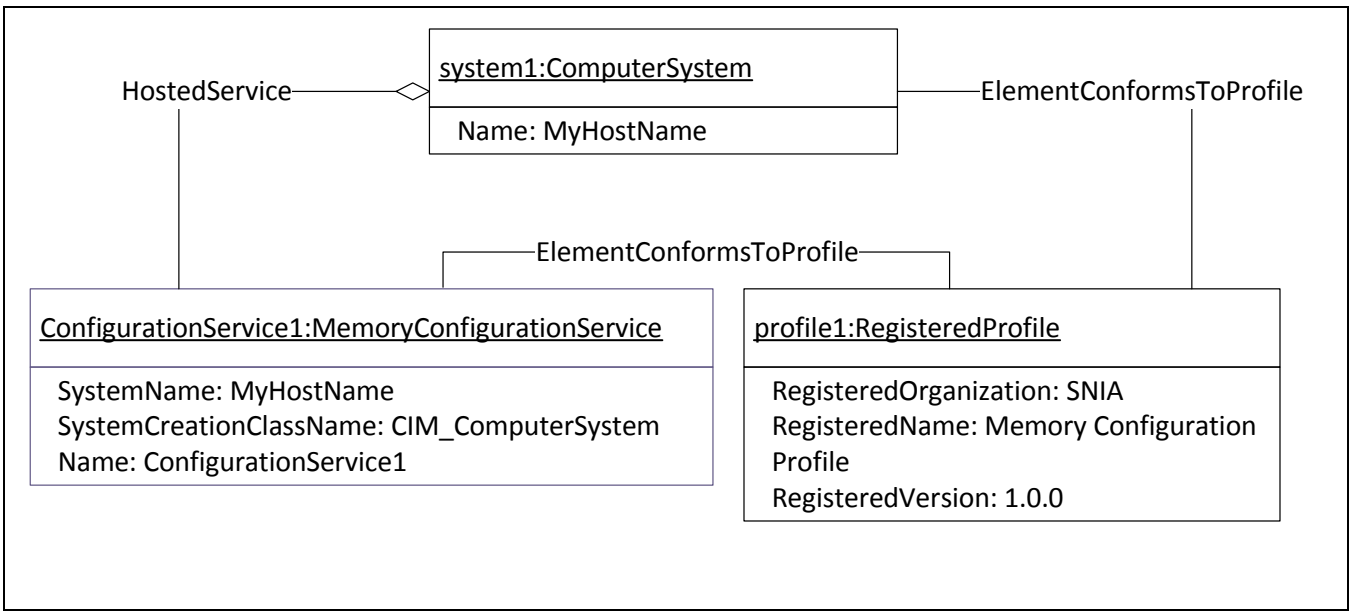

Figure 40 - Use Case - Profile Registration

12.5.2 Determine Support for Memory Configuration

One of several possible ways to determine configuration capabilities is to enumerate MemoryConfigurationCapabilities instances and examine the SupportedOperations attribute. If the desired configuration capability is located in the SupportedOperations list, the service instance offering this capability is located using the ElementCapabilities association. Figure 41 illustrates the case where the MemoryConfigurationService advertises support for memory allocation.

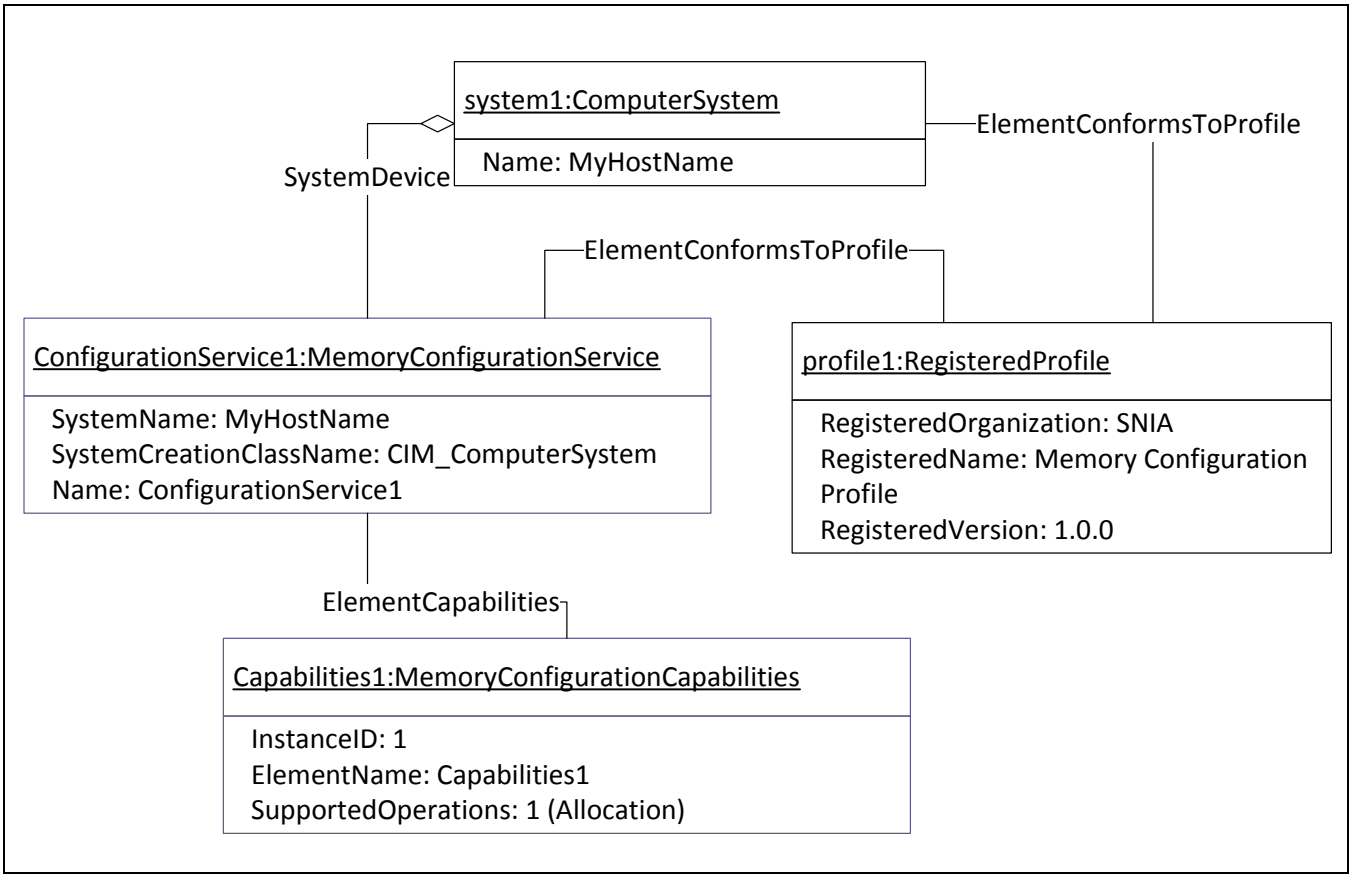

Figure 41 - Use Case - Memory Configuration Capabilities

12.5.3 Determine Support for Memory Features

A client seeking memory resources with specific capabilities can enumerate instances of CIM_MemoryCapabilities. For example, if a client is looking for memory with persistent characteristics it can examine the MemoryModes attribute looking for the persistent value. If a MemoryCapabilities instance with the persistent capability is found the CIM_ElementCapabilities association is used to locate the memory resource pool. An instance diagram for this use case is shown in Figure 42.

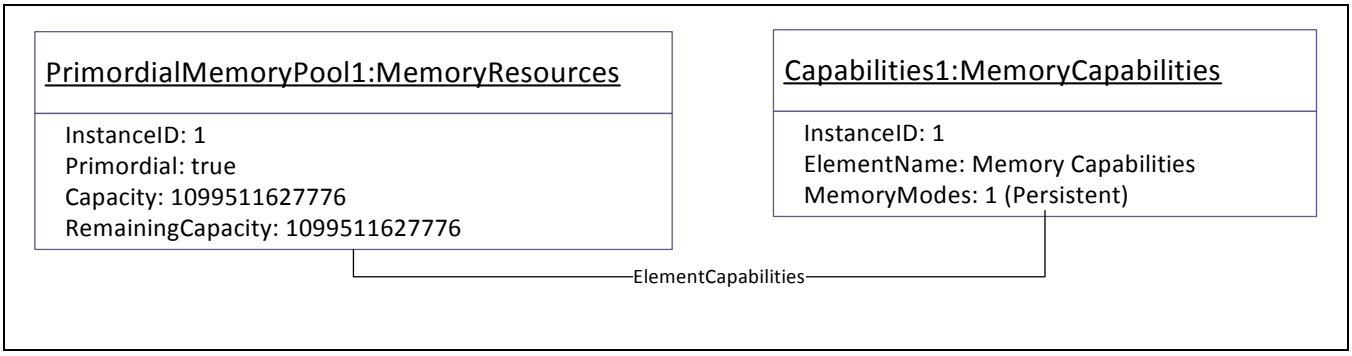

Figure 42 - Use Case - Memory Capability Discovery

12.5.4 Determine Available Capacity

Capacity available for allocation for a given pool is determined by enumerating instances of CIM_ResourcePool and subtracting the Reserved attribute from the Capacity attribute, as shown in Figure 42.

12.5.5 Allocate Capacity

Clients wishing to allocate capacity from a (memory) CIM_ResourcePool can locate the MemoryConfigurationService that handles allocation requests by following the CIM_ServiceAffectsElement association from the pool to the service. Once the service is located the AllocateFromPool method is used to request an allocation. The end result of the allocation is the appearance of a CIM_VisibleMemory instance along with a CIM_MemoryAllocationSettingData instance. The CIM_VisibleMemory instance is associated to the CIM_ResourcePool by an ElementAllocatedFromPool association. The CIM_ResourcePool.Reserved value is updated to reflect the allocation. In some cases the allocation request may require a reset or other event to complete. In this case the AllocateFromPool call results in a CIM_MemoryAllocationSetting instance associated with the CIM_ResourcePool until the triggering event occurs. The cases where a triggering event is required is given in Figure 43, the completed allocation is shown in Figure 44.

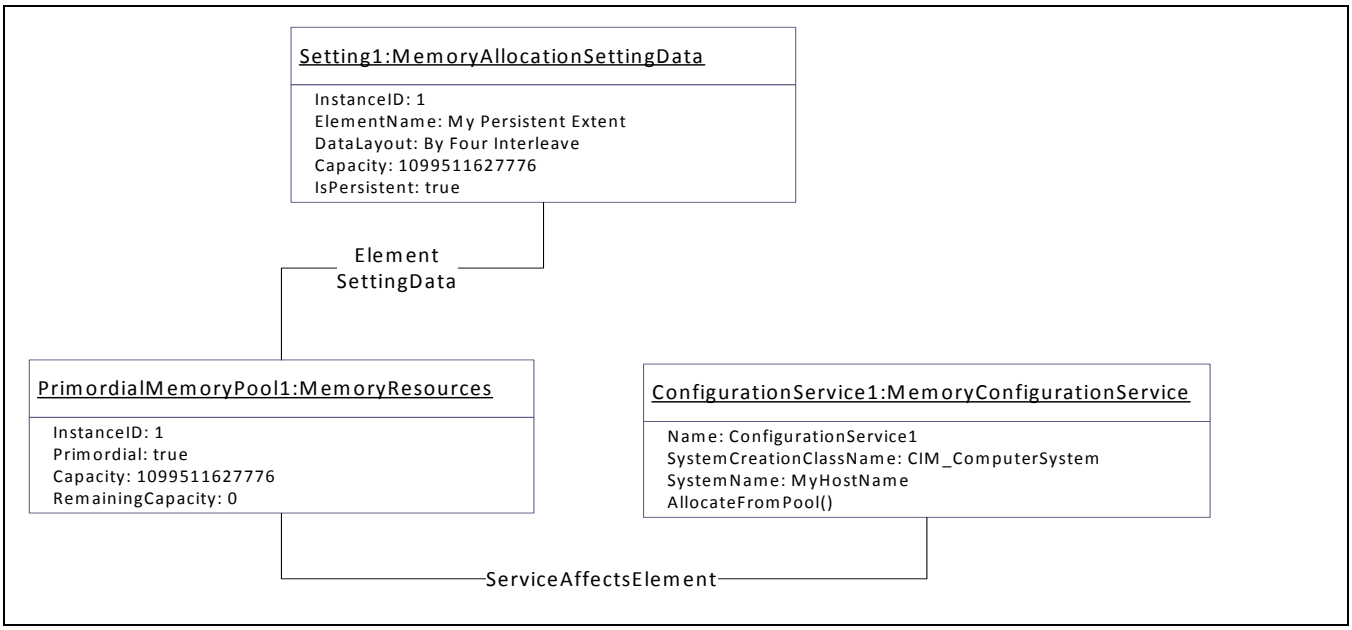

Figure 43 - Use Case - Allocation

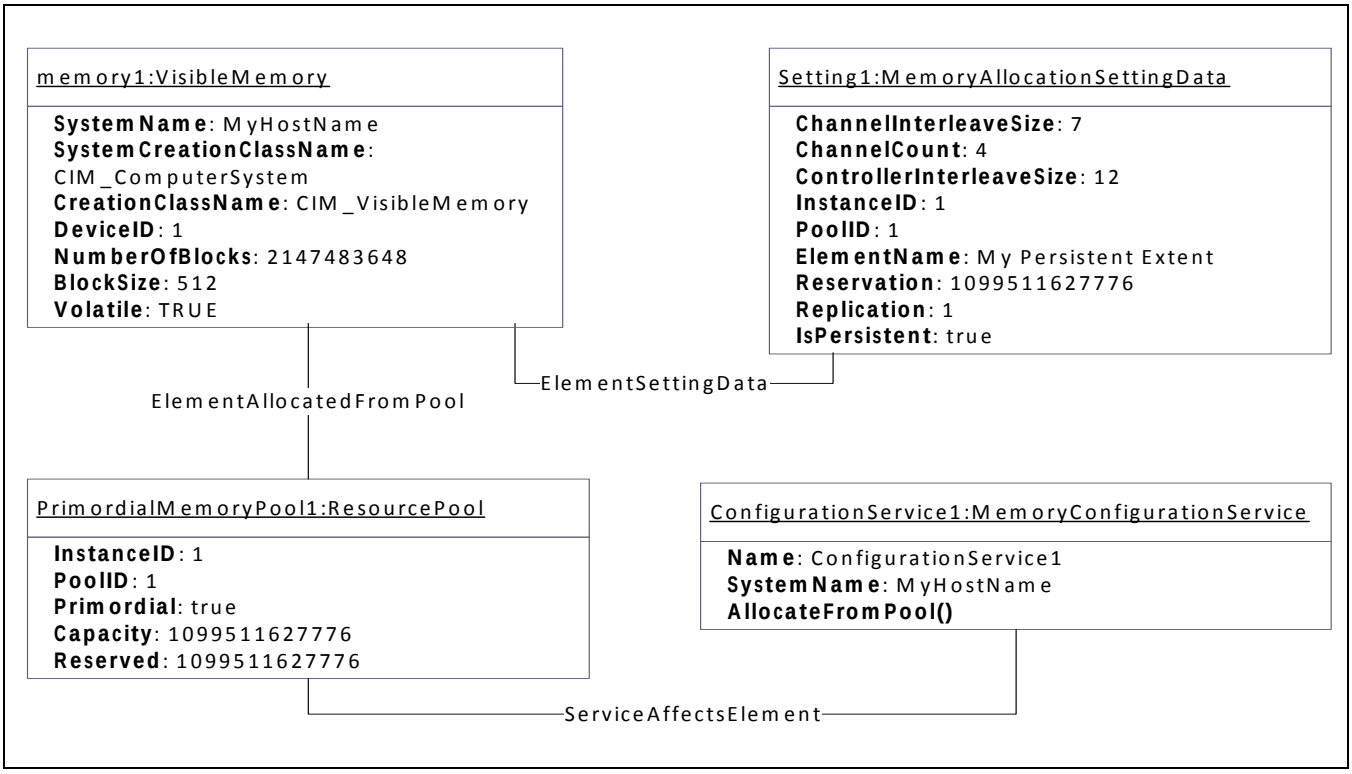

Figure 44 - Use Case - Allocation Complete

12.5.6 Deallocate Capacity

Clients wishing to deallocate a CIM_VisibleMemory instance first locate the pool from which it was allocated (ElementAllocatedFromPool to ResourcePool). Refer to Figure 45. Secondly locate the MemorConfigurationService instance associated with the pool (ServiceAffectsElement to MemoryConfigurationService). The ReturnToPool() method on the MemoryConfigurationService is then called with a reference to the VisibleMemory instance to be deleted. As with allocate, the deallocation may require a triggering event such as a device or system reset to take effect.

Figure 45 and Figure 46 depict the model before and after deallocation.

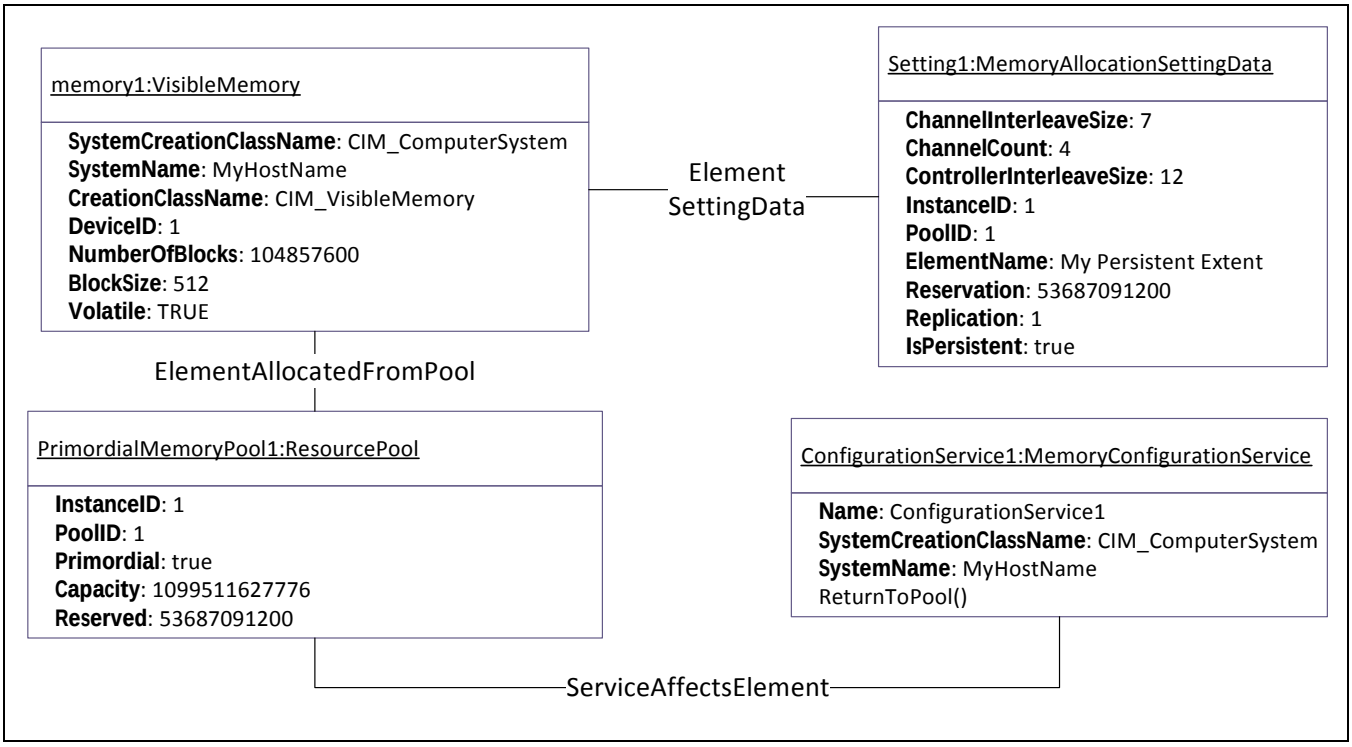

Figure 45 - Use Case - Before Deallocation

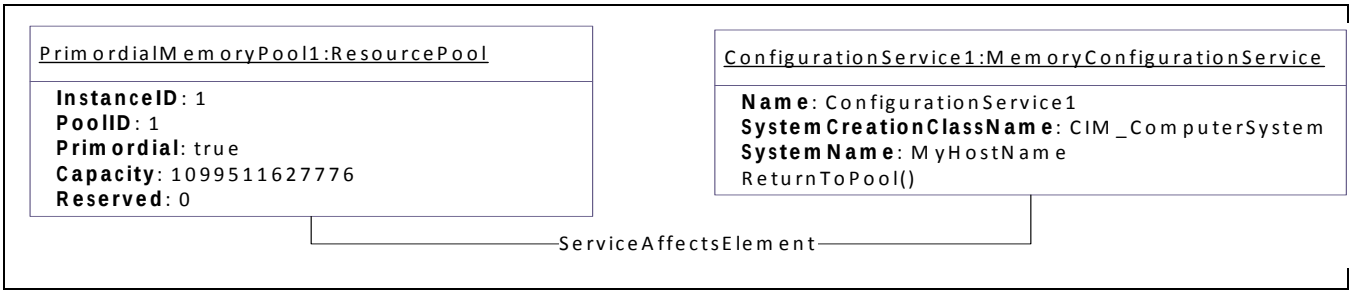

Figure 46 - Use Case - After Deallocation

12.5.7 Delete a Pending Request

An allocation request that has not taken place because it requires a state transition of some kind (e.g., a system reset) can be deleted by using the intrinsic DeleteInstance() method on the MemoryAllocationSettingData instance that represents the request.

12.6 CIM Elements

Table 137 shows the instances of CIM Elements for this profile. Instances of the following CIM Elements shall be implemented as described in Clauses (“Implementation”) and (“Methods”) may impose additional requirements on these elements.

Table 137 - CIM Elements

Element Name	Requirement	Description
CIM_RegisteredProfile	Mandatory	See 12.6.1

Table 137 - CIM Elements

Element Name	Requirement	Description
CIM_MemoryConfigurationService	Mandatory	See 12.6.2
CIM_ResourcePool	Mandatory	See 12.6.3
CIM_MemoryCapabilities	Mandatory	See 12.6.4
CIM_MemoryConfigurationCapabilities	Mandatory	See 12.6.5
CIM_MemoryAllocationSettingData	Mandatory	See 12.6.6
CIM_ElementSettingData	Mandatory	See 12.6.7
CIM_HostedService	Mandatory	See 12.6.8
CIM_ElementAllocatedFromPool	Mandatory	See 12.6.9
CIM_ConcreteComponent	Mandatory	See 12.6.10
CIM_ElementCapabilities	Mandatory	See 12.6.11

12.6.1 CIM_RegisteredProfile

CIM_RegisteredProfile is used to advertise implementation conformance with the Memory Configuration Profile. The CIM_RegisteredProfile class is defined by the Profile Registration Profile (*Storage Management Technical Specification, Part 3 Common Profiles, 1.7.0 Rev 5* 36 Profile Registration Profile). With the exception of the mandatory values specified in Table 138, the behavior of the CIM_RegisteredProfile instance is per the Profile Registration Profile. Table 138 contains the requirements for elements of this class.

Table 138 - Class: CIM_RegisteredProfile

Elements	Requirement	Notes
RegisteredName	Mandatory	This property shall have a value of “Memory Configuration”
RegisteredVersion	Mandatory	This property shall have a value of “1.0.0”
RegisteredOrganization	Mandatory	This property shall have a value of 11 (SNIA)

12.6.2 CIM_MemoryConfigurationService

The CIM_MemoryConfigurationService provides extrinsic methods suitable for configuring a memory subsystem. Implementations support attributes shown in Table 139.

Table 139 - Class: CIM_MemoryConfigurationService

Element	Requirement	Notes
CreationClassName	Mandatory	Key
Name	Mandatory	Key
SystemCreateClassName	Mandatory	Key
SystemName	Mandatory	Key
AllocateFromPool()	Conditional	Support is conditional on the existence of AllocateFromPool (2) in SupportedSynchronousOperations and SupportedAsynchronousActions in CIM_MemoryConfigurationCapabilities.
ReturnToPool()	Conditional	Support is conditional on the existence of ReturnToPool (2) in SupportedSynchronousOperations and SupportedAsynchronousActions in CIM_MemoryConfigurationCapabilities.

12.6.3 CIM_ResourcePool

CIM_ResourcePool (see Table 140) represents a pool of memory resources. It provides an aggregation point for like memory resources and a source of capacity for the allocation extrinsic provided by the CIM_MemoryConfigurationService.

Table 140 - Class: CIM_ResourcePool

Element	Requirement	Notes
InstaceID	Mandatory	Key
PoolID	Mandatory	None
Primordial	Mandatory	True
Capacity	Mandatory	Total raw capacity of the pool. This value can also be calculated by summing the capacity of the constituent CIM_RawMemory instances.
ResourceType	Optional	Memory
AllocationUnits	Mandatory	“Bytes”
RemainingCapacity	Optional	The total capacity allocated from the pool. This value is calculated in terms of the raw capacity consumed by any allocations (e.g. if an allocation utilizes more capacity than is exposed for metadata or replication, the Reserved total includes this extra capacity)

12.6.4 CIM_MemoryCapabilities

CIM_MemoryCapabilities models the features of a given set of memory resources. Implementations support attributes as given in Table 141.

Table 141 - Class: CIM_MemoryCapabilities

Element	Requirement	Notes
InstanceID	Mandatory	Key
ElementName	Mandatory	User friendly name for this capability instance
MemoryModes	Mandatory	Indicates the platform's support for various memory access mode.
ReplicationSupport	Mandatory	Indicates the platform's support for various types of memory replication.
ReliabilitySupport	Mandatory	Indicates the platform's support for memory reliability features.
Alignment	Optional	Alignment required when allocating memory regions
ChannelInterleaveSupport	Optional	List of supported memory channel interleave sizes
ChannelInterleaveWaySupport	Optional	Describes support for interleaving across memory channels.
MemoryControllerInterleaveSupport	Optional	Describes support for interleaving across memory controllers.
MemoryAffinitySupport	Optional	Processor affinity support (e.g., UMA, NUMA)

12.6.5 CIM_MemoryConfigurationCapabilities

CIM_MemoryConfigurationCapabilties describes the features available from an associated MemoryConfigurationService instance. Implementations support attributes as given in Table 142.

Table 142 - Class: CIM_MemoryConfigurationCapabilities

Element	Requirement	Notes
InstanceID	Mandatory	Key
ElementName	Mandatory	User friendly name for this capability instance
SupportedSynchronousOperations[]	Mandatory	A list of synchronous operations supported by the associated CIM_MemoryConfigurationService.
SupportedAsynchronousOperations[]	Mandatory	A list of asynchronous operations supported by the associated CIM_MemoryConfigurationService

12.6.6 CIM_MemoryAllocationSettingData

CIM_MemoryAllocationSettingData, shown in Table 143, describe parameters related to the allocation of visible memory from a memory resource pool. It can be used to describe an existing allocation in which case it associates with a CIM_VisibleMemory via CIM_ElementSettingData as the current setting. It can represent an allocation request that is in progress (e.g. requires a system reset to take effect) in which case it is associated with CIM_ResourcePool by CIM_ElementSettingData as the next setting.

Table 143 - Class: CIM_MemoryAllocationSettingData

Element	Requirement	Notes
InstanceID	Mandatory	Key
ElementName	Mandatory	Use friendly name for this setting instance
ChangeableType	Optional	“Other”, “Memory”, “Non-Volatile Memory”
Replication	Optional	If the allocated memory region is to utilize a channel interleave, this value indicates the number of channels to utilize.
ChannelInterleaveSize	Optional	If the allocated memory region is to utilize a channel interleave, this value indicates the number of bytes written/read before switching to the next channel.
ChannelCount	Optional	Boolean
ControllerInterleaveSize	Optional	If the allocated memory region is to utilize a controller interleave, this value indicates the number of bytes written/read before switching to the next controller
AllocationUnits	Mandatory	Specifies the units of the Reservation capacity property. Generally expected to be “bytes”.
Reservation	Mandatory	Requested capacity. The units of this type are given by AllocationUnits field.
PoolID	Mandatory	The PoolID of the CIM_ResourcePool instance from which this allocation will/has drawn resources.
Parent	Optional	If used, this value restricts the allocation to a specific NUMA node by identifying the DeviceID of the node processor.

12.6.7 CIM_ElementSettingData

12.6.7.1 Relating CIM_MemoryAllocationSettingData to CIM_VisibleMemory

The CIM_ElementSettingData association, shown in Table 144, links instances of CIM_VisibleMemory to their associated CIM_MemoryAllocationSettingData instances. This usage of the association would include the “Is Current” value for the IsCurrent attribute

Table 144 - Class: CIM_ElementSettingData - Use 1

Element	Requirement	Notes
SettingData	Mandatory	This property shall be a reference to an instance of the CIM_MemoryAllocationSettingData.
ManagedElement	Mandatory	This property shall be a reference to an instance of CIM_VisibleMemory.
IsCurrent	Optional	Used to indicate a setting that is currently in-use (e.g. associated with an existing memory extent). This shall be true for this usage.
IsNext	Optional	Used to indicate a requested setting that has not yet taken effect. This shall be false for this usage.

12.6.7.2 Relating CIM_MemoryAllocationSettingData to CIM_ResourcePool

The CIM_ElementSettingData association can also be used to link pending settings to the pool from which capacity will be drawn when the setting is finalized. This usage of the association would set the IsNext attribute to “Is Next”. Table 145 contains the requirements for elements of this class.

Table 145 - Class: CIM_ElementSettingData - Use 2

Element	Requirement	Notes
SettingData	Mandatory	This property shall be a reference to an instance of the CIM_MemoryAllocationSettingData.
ManagedElement	Mandatory	**This property shall be a reference to an instance of CIM_ResourcePool.**
IsCurrent	Optional	Used to indicate a setting that is currently in-use (e.g. associated with an existing memory extent). When representing staged requests, this shall be set to false.
IsNext	Optional	Used to indicate a requested setting that has not yet taken effect. When representing staged requests this shall be set to true.

12.6.8 CIM_HostedService

The CIM_HostedService association is used to relate an instance of CIM_ComputerSystem with a CIM_MemoryConfigurationService instance. Table 146 contains the requirements for elements of this class.

Table 146 - Class: CIM_HostedService

Element	Requirement	Notes
Antecedent	Mandatory	This property shall be a reference to an instance of CIM_ComputerSystem. Cardinality is “1”.
Dependent	Mandatory	This property shall be a reference to an instance of CIM_MemoryConfigurationService. Cardinality is “1”.

12.6.9 CIM_ElementAllocatedFromPool

The CIM_ElementAllocatedFromPool association is used to relate the CIM_VisibleMemory instance to the CIM_ResourcePool instance to which it applies. Table 147 contains the requirements for elements of this class.

Table 147 - Class: CIM_ElementAllocatedFromPool

Element	Requirement	Notes
Antecedent	Mandatory	This property shall be a reference to an instance of CIM_ResourcePool
Dependent	Mandatory	This property shall be a reference to an instance of CIM_VisibleMemory

12.6.10CIM_ConcreteComponent

The CIM_ConcreteComponent association is used to relate the CIM_RawMemory to the CIM_ResourcePool of which it is part. Table 148 contains the requirements for elements of this class.

Table 148 - Class: CIM_ConcreteComponent

Element	Requirement	Notes
GroupComponent	Mandatory	This property shall be a reference to an instance of the CIM_ResourcePool class. Cardinality is “1”
PartComponent	Mandatory	This property shall be a reference to an instance of the CIM_RawMemory. Cardinality is “1.*”.

12.6.11CIM_ElementCapabilities

12.6.11.1 Relating CIM_ResourcePool to CIM_MemoryCapabilities

The CIM_ElementCapabilities association is used to relate the CIM_ResourcePool to a description of its capabilities given by CIM_MemoryCapabilities. Table 149 contains the requirements for elements of this class.

Table 149 - Class: CIM_ElementCapabilities use 1

Element	Requirement	Notes
ManagedElement	Mandatory	This property shall be a reference to an instance of the CIM_MemoryConfigurationService class. Cardinality is “1”.
Capabilities	Mandatory	This property shall be a reference to an instance of the CIM_MemoryCapabilities class. Cardinality is “1”.

12.6.11.2 Relating CIM_MemoryConfigurationService to CIM_MemoryConfigurationCapabilities

The CIM_ElementCapabilities association is used to relate the CIM_MemoryConfigurationService to a description of its capabilities given by CIM_MemoryConfigurationCapabilities. Table 150 contains the requirements for elements of this class.

Table 150 - Class CIM_ElementCapabilities use 2

Element	Requirement	Notes
ManagedElement	Mandatory	This property shall be a reference to an instance of the CIM_MemoryConfigurationService class. Cardinality is “1.*”.

Table 150 - Class CIM_ElementCapabilities use 2

Element	Requirement	Notes
Capabilities	Mandatory	This property shall be a reference to an instance of the CIM_MemoryConfigurationCapabilities class. Cardinality is “1”.

EXPERIMENTAL

EXPERIMENTAL

13 Persistent Memory Configuration Profile

13.1 Synopsis

Profile Name: Persistent Memory Configuration

Version: 1.0.0

Organization: SNIA

Central Class: Persistent Memory Service

Scoping Class: ComputerSystem

Related Profiles: Table 151 describes the related profiles for the Memory Configuration Profile:

Table 151 - Related Profiles

Profile Name	Organization	Version	Requirement	Description
Profile Registration	DMTF	1.1.0	Mandatory	DMTF DSP1033
Multi-Type System Memory	DMTF	1.0.0a	Mandatory	DMTF DSP1071
Memory Configuration	SNIA	1.0.0	Mandatory	

13.2 Description

Persistent memory, like other forms of storage media is only valuable when its contents can be reliably located and protected. The Persistent Memory Configuration Profile describes a model for managing persistent memory that provides for these features. In this profile, persistent memory regions are utilized as primordial extents. A management service is described which provides allocation, deallocation and modification methods that operate on these primordial persistent memory extents. Allocation requests utilize primordial extents to create concrete system addressable storage extents. Deallocation requests revert concrete extents into their primordial form. Modification requests can grow, shrink and make other types of modifications to concrete extents.

13.2.1 Class Diagram

Figure 47 shows the Persistent Memory Configuration Profile class hierarchy. For simplicity, the prefix CIM_ has been removed from the names of the classes

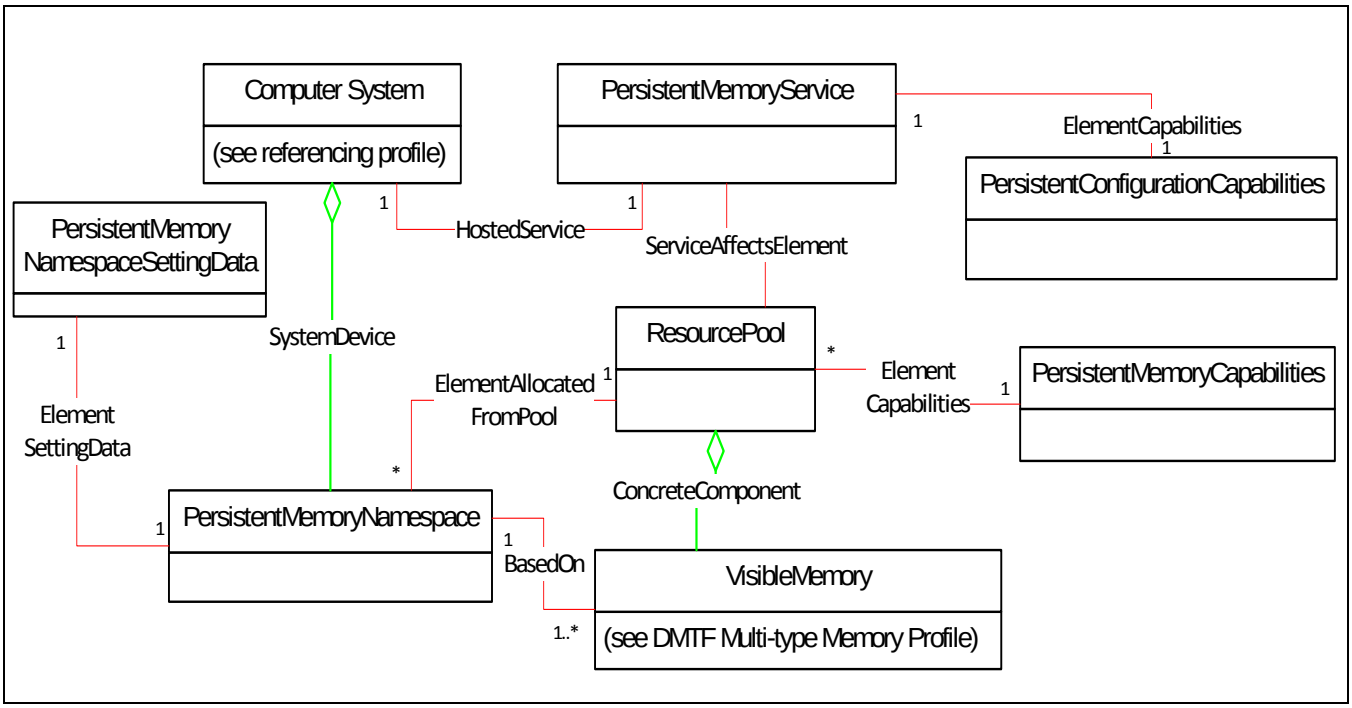

Figure 47 - Persistent Memory Configuration: Class Diagram

13.2.1.1 CIM_PersistentMemoryService

The PersistentMemoryService is the central class of the profile. It models the system's support for modifying its persistent memory configuration.

13.2.1.2 CIM_ResourcePool (persistent memory)

The ResourcePool aggregates persistent memory primordial extents (VisibleMemory).

13.2.1.3 CIM_PersistentConfigurationCapabilities & CIM_PersistentMemoryCapabilities

This profile supports run-time detection of memory features. PersistentConfigurationCapabilities advertises the system's support for persistent memory namespace configuration. CIM_PersistentMemoryCapabilities indicates the capabilities of the VisibleMemory primordial extents and other system components as regards their ability to support various persistent memory quality of service levels.

13.2.1.4 CIM_PersistentMemoryNamespace

PersistentMemoryNamespace models a named region of persistent memory.

13.2.1.5 CIM_PersistentMemoryNamespaceSettingData

PersistentMemoryNamespaceSettingData reflects the parameters specified during namespace creation or modification that define the quality of service offered by the associated PersistentMemoryNamespace.

13.2.1.6 CIM_RegisteredProfile

Figure 2-1 includes CIM_RegisteredProfile which is expected to be implemented per the Profile Registration Profile (*Storage Management Technical Specification, Part 3 Common Profiles, 1.7.0 Rev 5* Clause 36 Profile Registration Profile).

13.2.1.7 DMTF Multi-Type System Memory Profile & SNIA Memory Configuration Profile

The VisibleMemory class is described in detail in the DMTF Multi-Type System Memory Profile. The SNIA Memory Configuration Profile describes a mechanism by which VisibleMemory instances can be created. VisibleMemory instances model memory capacity that has been allocated (concrete) from a memory resources pool. In the context of this profile VisibleMemory instances are presumed to be persistent and to require further configuration (i.e. PersistentMemoryNamespace creation) before being useful to the system.

13.3 Implementation

13.3.1 CIM_ ResourcePool

3.3 Implementations of this profile shall contain at least one instance of CIM_ResourcePool. The pool shall serve as an aggregation point for CIM_VisibleMemory instances which represent persistent memory extents.

13.3.1.1 Determining Pool Capacity

The capacity of a CIM_ResourcePool instance shall be the sum of the capacities of the constituent CIM_VisibleMemory extents.

13.3.2 CIM_PersistentMemoryCapabilities

Implementations of this profile shall include at least one instance of CIM_PersistentMemoryCapabilities. Each instance shall describe the capabilities of one or more instances of CIM_ResourcePool as determined by the CIM_ElementCapabilities association.

13.3.3 CIM_PersistentMemoryService

Implementations of this profile shall contain at least one instance of CIM_PersistentMemoryService. Instances of this class shall conditionally support allocation, deallocation and modification of PersistentMemoryNamespace devices. Availability of any given extrinsic shall be determined by examining the CIM_PersistentConfigurationCapabilities instance to which it is associated (CIM_ElementCapabilities).

13.3.4 CIM_PersistentConfigurationCapabilities

Implementations of this profile shall contain at least one instance of CIM_PersistentConfigurationCapabilities. Each instance shall be associated with one or more instances of CIM_PersistentMemoryService via a CIM_ElementCapabilities association.

13.3.5 CIM_PersistentMemoryNamespace

CIM_PersistentMemoryNamespace shall be used to represent configured persistent memory. Instances shall exist conditionally, when the necessary configuration steps required to create a persistent memory device have occurred.

13.3.6 CIM_PersistentMemoryNamespaceSettingData

CIM_PersistentMemoryNamespaceSettingData shall represent settings supplied during the configuration or modification of a CIM_PersistentMemoryNamespace. Existence of CIM_PersistentMemoryNamespaceSettingData instances shall be conditional. When an instance of CIM_PersistentMemoryNamespace exists there shall be an associated (ElementSettingData) CIM_PersistentMemoryNamespaceSettingData instance.

13.3.7 CIM_ElementCapabilities

CIM_ElementCapabilities shall associate the CIM_PersistentConfigurationCapabilities to CIM_PersistentMemoryService. Capabilities so associated shall describe the extrinsic method support offered by the service instance.

13.3.8 CIM_ServiceAffectsElement

CIM_ServiceAffectsElement associations shall relate CIM_PersistentMemoryService instances with the CIM_ResourcePool instances they are able to operate on.

13.3.9 CIM_ConcreteComponent

Persistent CIM_VisibleMemory instances pooled together for management purposes are represented by CIM_ResourcePool instances. CIM_VisibleMemory instances shall be associated with a CIM_ResourcePool instance by the CIM_ConcreteComponent association.

13.3.10 CIM_ElementAllocatedFromPool

CIM_PersistentMemoryNamespace instances shall be associated to the CIM_ResourcePool from which their capacity is drawn using the CIM_ElementAllocatedFromPool association.

13.3.11 CIM_HostedService

The CIM_PersistentMemoryService shall be associated to the CIM_ComputerSystem by CIM_HostedService.

13.3.12 CIM_ElementSettingData

Instances of CIM_PersistentMemoryNamespaceSettingData shall be associated to the CIM_PersistentMemoryNamespace instance to which they apply using an ElementSettingData.

13.3.13 CIM_ElementConformsToProfile

CIM_ElementConformsToProfile shall associate a CIM_RegisteredProfile instance representing the Persistent Memory Service profile to the central class of the profile CIM_PersistentMemoryService.

13.3.14 CIM_SystemDevice

Instances of PersistentMemoryNamespace shall be associated to the scoping instance of CIM_ComputerSystem by CIM_SystemDevice.

13.4 Methods

This clause details the requirements for supporting intrinsic operations and extrinsic methods for the CIM elements defined by this profile.

13.4.1 CIM_ResourcePool

Implementations of this profile shall support the operations listed in Table 152 for CIM_ResourcePool.

Table 152 - Operations: CIM_ResourcePool

Operation	Requirement	Messages
GetInstance	Mandatory	None
Associators	Mandatory	None
AssociatorNames	Mandatory	None
References	Mandatory	None

Table 152 - Operations: CIM_ResourcePool

Operation	Requirement	Messages
ReferenceNames	Mandatory	None
EnumerateInstances	Mandatory	None
EnumerateInstanceNames	Mandatory	None

13.4.2 CIM_PersistentMemoryCapabilities

Implementations of this profile shall support the operations listed in Table 153 for CIM_PersistentMemoryCapabilities.

Table 153 - Operations: CIM_PersistentMemoryCapabilities

Operation	Requirement	Messages
GetInstance	Mandatory	None
Associators	Mandatory	None
AssociatorNames	Mandatory	None
References	Mandatory	None
ReferenceNames	Mandatory	None
EnumerateInstances	Mandatory	None
EnumerateInstanceNames	Mandatory	None

13.4.3 CIM_PersistentMemoryService

Implementations of this profile shall support the operations listed in Table 154 for CIM_PersistentMemoryService.

Table 154 - Operations: CIM_PersistentMemoryService

Operation	Requirement	Messages
GetInstance	Mandatory	None
Associators	Mandatory	None
AssociatorNames	Mandatory	None
References	Mandatory	None
ReferenceNames	Mandatory	None
EnumerateInstances	Mandatory	None
EnumerateInstanceNames	Mandatory	None
AllocateFromPool	Optional	None
ReturnToPool	Optional	None
ModifyNamespace	Optional	None

13.4.3.1 CIM_PersistentMemoryService.AllocateFromPool()

The AllocateFromPool() method is used to configure persistent memory extents resulting in a system visible persistent memory device. See Table 155 for return values and Table 156 for parameters.

Table 155 - AllocateFromPool() Method: Return Code Values

Value	Description
0	Request was successfully executed
1	Not Supported
2	Unknown
3	Timeout
4	Failed
5	Invalid Parameter
4096	Request was successfully staged for future execution.
4097	Inconsistent Resources
4098	Inconsistent Parameters
4099	Request did not complete in its entirety and partial results could not be undone

AllocateFromPool() parameters are listed in Table 156

Table 156 - AllocateFromPool() Method: Parameters

Qualifiers	Name	Type	Description
IN, REQ	Pool	CIM_ResourcePool REF	The pool of memory resources from which to allocate the visible memory region.
IN,REQ	Goal	string	An embedded instance of CIM_PersistentMemoryNamespaceSettingData which describes the allocation request.
OUT	Namespace	CIM_PersistentMemoryNamespace REF	Object path of the resulting CIM_PersistentMemoryNamespace. Undefined if the return code is other than 0

13.4.3.2 CIM_PersistentMemoryService.ReturnToPool()

The ReturnToPool() method deletes an instance of CIM_PersistentMemoryNamespace, returning the capacity it occupied to the CIM_ResourcePool instance from which it was allocated. Note that associated CIM_PersistentMemoryNamespaceSettingData are deleted as well. See Table 157 for return values and Table 158 for parameters.

Table 157 - ReturnToPool() Method: Return Code Values

Value	Description
0	Request was successfully executed
1	Method is not supported in this implementation
2	Unknown

Table 157 - ReturnToPool() Method: Return Code Values

Value	Description
3	Timeout
4	Failed
5	Invalid Parameter
4096	Request Was Successfully Staged for Future Execution
4099	Request did not complete in its entirety and partial results could not be undone

ReturnToPool() parameters are listed in Table 158:

Table 158 - ReturnToPool() Method: Parameters

Qualifiers	Name	Type	Description
IN, REQ	Namespace	CIM_PersistentMemoryNamespace REF	The persistent memory namespace instance to be deleted.

13.4.3.3 CIM_PersistentMemoryService.ModifyNamespace()

The ModifyNamespace() method is used to alter the characteristics of an existing namespace. Increases to the namespace size allocate resources from the same pool that existing resources have been drawn from. Likewise shrinking a namespace returns capacity to the source pool. See Table 159 for return values.

Table 159 - ModifyNamespace() Method: Return Code Values

Value	Description
0	Request was successfully executed
1	Method is not supported in this implementation
2	Unknown
3	Timeout
4	Failed
5	Invalid Parameter
4096	Request was successfully staged for future execution
4097	Insufficient Resources
4098	Inconsistent Parameters
4099	Request did not complete in its entirety and partial results could not be undone

ModifyNamespace() parameters are listed in Table 160:

Table 160 - ModifyNamespace() Method: Parameters

Qualifiers	Name	Type	Description
IN	Namespace	CIM_PersistentMemoryNamespace REF	The persistent memory namespace instance to be changed.

Table 160 - ModifyNamespace() Method: Parameters

Qualifiers	Name	Type	Description
IN	Goal	string	Embedded instance of CIM_PersistentMemoryNamespaceSettingData. Note that the implementation may restrict the fields in the NamespaceSetting instance that apply to a modify operation. Modifying a property for which modification is not supported results in an Invalid Parameter return code. The goal is expected to contain values for all fields with those that are not being changed populated with their current values.

13.4.4 CIM_PersistentConfigurationCapabilities

Implementations of this profile shall support the operations listed in Table 161 for CIM_PersistentConfigurationCapabilities.

Table 161 - Operations: CIM_PersistentConfigurationCapabilities

Operation	Requirement	Messages
GetInstance	Mandatory	None
Associators	Mandatory	None
AssociatorNames	Mandatory	None
References	Mandatory	None
ReferenceNames	Mandatory	None
EnumerateInstances	Mandatory	None
EnumerateInstanceNames	Mandatory	None

13.4.5 CIM_PersistentMemoryNamespace

Implementations of this profile shall support the operations listed in Table 162 for CIM_PersistentMemoryNamespace

DeleteInstance performs the exact same operation as CIM_PersistentMemoryService.ReturnToPool().

Table 162 - Operations: CIM_PersistentMemoryNamespace

Operation	Requirement	Messages
GetInstance	Mandatory	None
Associators	Mandatory	None
AssociatorNames	Mandatory	None
References	Mandatory	None
ReferenceNames	Mandatory	None
EnumerateInstances	Mandatory	None

Table 162 - Operations: CIM_PersistentMemoryNamespace

Operation	Requirement	Messages
EnumerateInstanceNames	Mandatory	None
DeleteInstance	Optional	None

13.4.6 CIM_PersistentMemoryNamespaceSettingData

Implementations of this profile shall support the operations listed in Table 163 for CIM_PersistentMemoryNamespaceSettingData.

Table 163 - Operations: CIM_PersistentMemoryNamespaceSettingData

Operation	Requirement	Messages
GetInstance	Mandatory	None
Associators	Mandatory	None
AssociatorNames	Mandatory	None
References	Mandatory	None
ReferenceNames	Mandatory	None
EnumerateInstances	Mandatory	None
EnumerateInstanceNames	Mandatory	None

13.4.7 CIM_ElementCapabilities

Implementations of this profile shall support the operations listed in Table 164 for CIM_ElementCapabilities.

Table 164 - Operations: CIM_ElementCapabilities

Operation	Requirement	Messages
GetInstance	Mandatory	None
Associators	Mandatory	None
AssociatorNames	Mandatory	None
References	Mandatory	None
ReferenceNames	Mandatory	None
EnumerateInstances	Mandatory	None
EnumerateInstanceNames	Mandatory	None

13.4.8 CIM_ServiceAffectsElement

Implementations of this profile shall support the operations listed in Table 165 for the CIM_ServiceAffectsElement class.

Table 165 - Operations: CIM_ServiceAffectsElement

Operation	Requirement	Messages
GetInstance	Mandatory	None
EnumerateInstances	Mandatory	None
EnumerateInstanceNames	Mandatory	None

13.4.9 CIM_ConcreteComponent

Implementations of this profile shall support the operations listed in Table 166 for the CIM_ConcreteComponent class.

Table 166 - Operations: CIM_ConcreteComponent

Operation	Requirement	Messages
GetInstance	Mandatory	None
EnumerateInstances	Mandatory	None
EnumerateInstanceNames	Mandatory	None

13.4.10 CIM_ElementAllocatedFromPool

Implementations of this profile shall support the operations listed in Table 167 for the CIM_ElementAllocatedFromPool class.

Table 167 - Operations: CIM_ElementAllocatedFromPool

Operation	Requirement	Messages
GetInstance	Mandatory	None
EnumerateInstances	Mandatory	None
EnumerateInstanceNames	Mandatory	None

13.4.11 CIM_HostedService

Implementations of this profile shall support the operations listed in Table 168 for the CIM_HostedService class.

Table 168 - Operations: CIM_HostedService

Operation	Requirement	Messages
GetInstance	Mandatory	None
EnumerateInstances	Mandatory	None
EnumerateInstanceNames	Mandatory	None

13.4.12CIM_ElementSettingData

Implementations of this profile shall support the operations listed in Table 169 for the CIM_ElementSettingData class.

Table 169 - Operations: CIM_ElementSettingData

Operation	Requirement	Messages
GetInstance	Mandatory	None
EnumerateInstances	Mandatory	None
EnumerateInstanceNames	Mandatory	None

13.4.13CIM_ElementConformsToProfile

Implementations of this profile shall support the operations listed in Table 170 for the CIM_ElementConformsToProfile class.

Table 170 - Operations: CIM_ElementConformsToProfile

Operation	Requirement	Messages
GetInstance	Mandatory	None
EnumerateInstances	Mandatory	None
EnumerateInstanceNames	Mandatory	None

13.4.14CIM_SystemDevice

Implementations of this profile shall support the operations listed in Table 171 for the CIM_SystemDevice class.

Table 171 - Operations: CIM_SystemDevice

Operation	Requirement	Messages
GetInstance	Mandatory	None
EnumerateInstances	Mandatory	None
EnumerateInstanceNames	Mandatory	None

13.5 Use Cases

This clause contains object diagrams and use cases for the Persistent Memory Configuration Profile.

13.5.1 Advertising Profile Conformance

Figure 48 shows how an instance of CIM_RegisteredProfile is used to indicate the presence of a conforming implementation of the Persistent Memory Configuration Profile and to identify instances of its central class CIM_PersistentMemoryService. Clients can follow the CIM_ElementConformsToProfile association directly to the central class or likewise identify the hosting ComputerSystem instance.

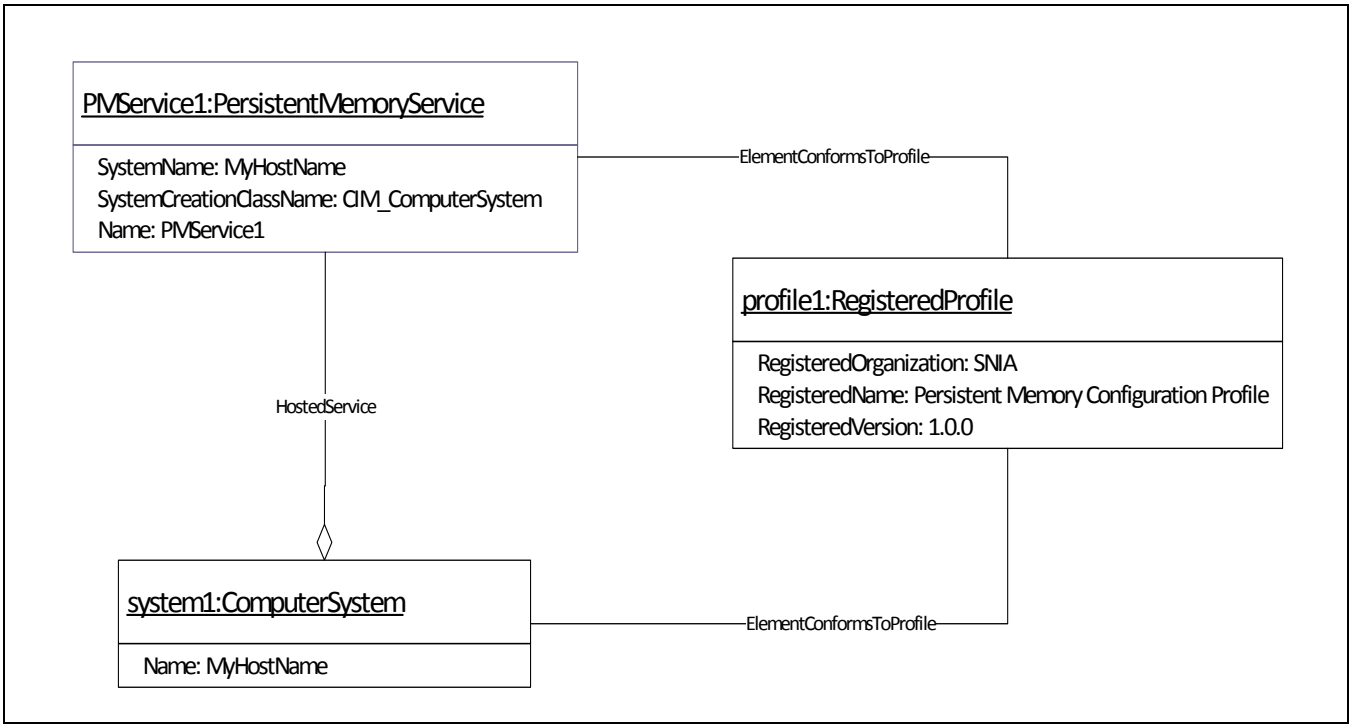

Figure 48 - Use Case: Profile Registration

13.5.2 Locate Existing Persistent Namespaces

Persistent namespaces are located by enumerating CIM_PersistentMemoryNamespace instances.

13.5.3 Determine Support for Persistent Memory Features and Operations

Support for persistent memory operations is signaled by the attributes of the CIM_PersistentConfigurationCapabilities and CIM_PersistentMemoryCapabilities classes. Support for creation/deletion/modification of CIM_PersistentMemoryNamespaces is determined by enumerating instances of CIM_PersistentConfigurationCapabilities and examining the supported operations attributes. Once an instance with the desired capabilities is found the ElementCapabilities association is used to locate the service that offers these capabilities. The pools that the service can act upon are located by following the CIM_ServiceAffectsElement association.

Support for specific persistent memory quality of service related features is determined by enumerating instances of CIM_PersistentMemoryCapabilities and examining attributes. For example, if the constituent VisibleMemory extents include mirrored persistent capacity the CIM_PersistentMemoryCapabilities instance would have its attributes set so as to indicate the ability of the ResourcePool to support mirrored PersistentMemoryNamespace creation.

These capability instances are given in Figure 49. In the example the PersistentMemoryService is advertising support for creating and deleting persistent memory namespaces.

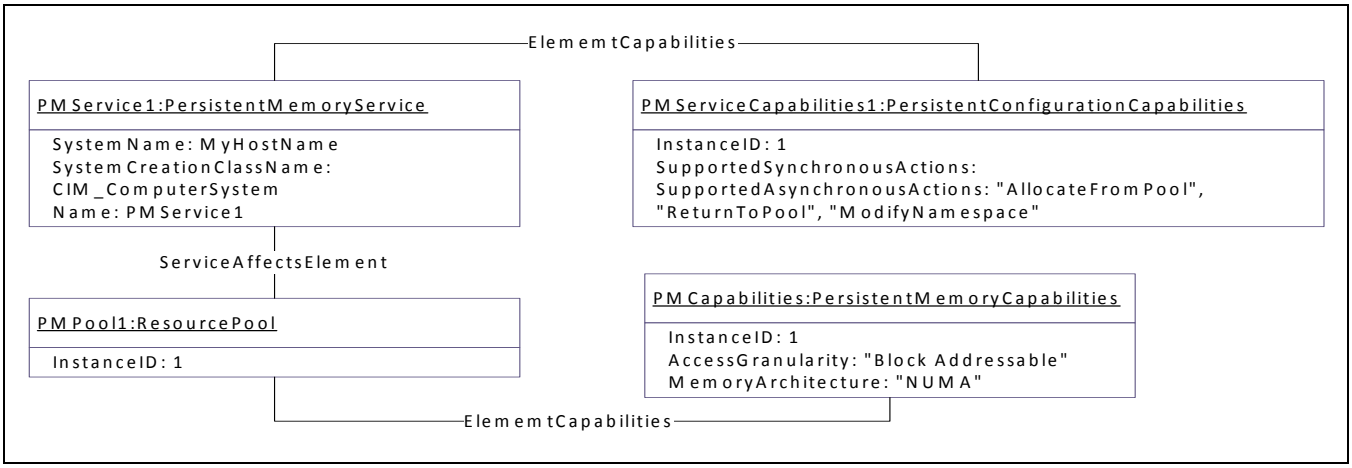

Figure 49 - Use Case - Detect Capabilities

13.5.4 Determine Availability of Persistent Capacity

Available persistent capacity is determined by enumerating ResourcePool instances and examining the RemainingCapacity attribute.

13.5.5 Create a Persistent Namespace

The steps to create a persistent namespace are:

- Enumerate instances of ResourcePool, find a pool that has sufficient remaining capacity and whose associated CIM_PersistentMemoryCapabilities indicates support for the desired quality of service.
- Follow the CIM_ServiceAffectsElement association to the PersistentMemoryService instance that handles the selected pool.
- Utilize the AllocateFromPool() extrinsic, specify pool, desired size, quality of service parameters.

When the AllocateFromPool() extrinsic returns with a successful return code the output parameter “Namespace” contains a reference to the created PersistentMemoryNamespace instance. A NamespaceSetting instance which matches the goal parameter to AllocateFromPool() is also created and associated to the new namespace by a CIM_ElementSettingData association.

Instances involved in this use case are given in Figure 50.

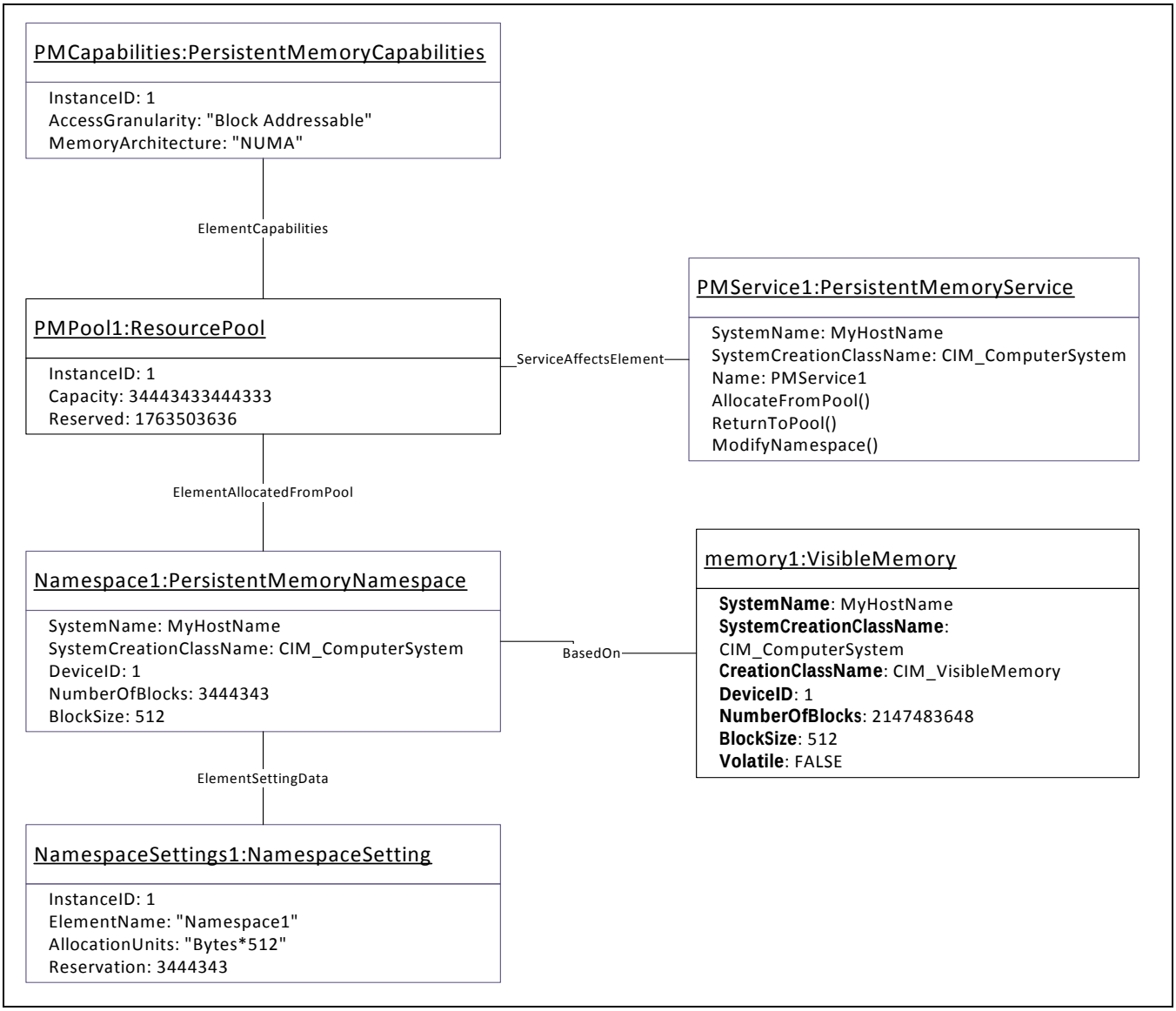

Figure 50 - Use Case - Create Namespace

13.5.6 Deallocate a Persistent Namespace

Deleting a namespace can be accomplished using the intrinsic DeleteInstance method. An alternate method is to use the ReturnToPool() extrinsic available on the PersistentMemoryService instance that was used to create it. This can be determined by starting at the PersistentMemoryNamespace instance (to be deleted) and following the CIM_ElementAllocatedFromPool and CIM_ServiceAffectsElement associations.

13.5.7 Modify a Persistent Namespace

Instances are modified using PersistentMemoryService.ModifyNamespace method. The service associated with the instance to be modified is found by starting at the PersistentMemoryNamespace instance (to be deleted) and following the CIM_ElementAllocatedFromPool and CIM_ServiceAffectsElement associations.

13.6 CIM Elements

Table 172 shows the instances of CIM Elements for this profile. Instances of the following CIM Elements shall be implemented as described in Clause 12 Memory Configuration Profile. Clauses ("Implementation") and ("Methods") may impose additional requirements on these elements.

Table 172 - CIM Elements

Element	Requirement	Description
CIM_RegisteredProfile	Mandatory	See 13.6.1
CIM_ResourcePool	Mandatory	See 13.6.2
CIM_PersistentMemoryCapabilities	Mandatory	See 13.6.3
CIM_PersistentMemoryService	Mandatory	See 13.6.4
CIM_PersistentConfigurationCapabilities	Mandatory	See 13.6.5
CIM_PersistentMemoryNamespace	Mandatory	See 13.6.6
CIM_PersistentMemoryNamespaceSettingData	Mandatory	See 13.6.7
CIM_ServiceAffectsElement	Mandatory	See 13.6.9
CIM_ElementSettingData	Mandatory	See 13.6.13
CIM_HostedService	Mandatory	See 13.6.12
CIM_ElementAllocatedFromPool	Mandatory	See 13.6.11
CIM_ConcreteComponent	Mandatory	See 13.6.10
CIM_ElementCapaboilities	Mandatory	See 13.6.8
CIM_ElementConformsToProfile	Mandatory	See 13.6.14
CIM_SystemDevice	Mandatory	See 13.6.15
CIM_BasedOn	Mandatory	See 13.6.16

13.6.1 CIM_RegisteredProfile

CIM_RegisteredProfile is used to advertise implementation conformance with the Persistent Memory Configuration Profile (see *Clause* 13). The CIM_RegisteredProfile class is defined by the *Storage Management Technical Specification, Part 3 Common Profiles, 1.7.0 Rev 5* Clause 36 Profile Registration Profile.

Table 173 - Class: CIM_RegisteredProfile

Elements	Requirement	Notes
RegisteredName	Mandatory	This property shall have a value of "Persistent Memory Configuration"
RegisteredVersion	Mandatory	This property shall have a value of "1.0.0"
RegisteredOrganization	Mandatory	This property shall have a value of 11 (SNIA)

13.6.2 CIM_ResourcePool

CIM_ResourcePool represents a pool of persistent memory resources. It provides an aggregation point for like persistent memory resources and a source of capacity for the allocation extrinsic provided by the CIM_PersistentMemoryService. ResourcePool properties are described in Table 174, “Class: CIM_ResourcePool”.

Table 174 - Class: CIM_ResourcePool

Element	Requirement	Notes
InstaceID	Mandatory	Key
PoolID	Mandatory	None
Primordial	Mandatory	True
Capacity	Mandatory	Total usable capacity, both allocated and unallocated in bytes.
ResourceType	Optional	“Non-Volatile Memory”
AllocationUnits	Mandatory	“Bytes”
RemainingCapacity	Optional	The total capacity allocated from the pool. This value is calculated in terms of the raw capacity consumed by any allocations (e.g. if an allocation utilizes more capacity than is exposed for meta data or replication, the Reserved total includes this extra capacity).

13.6.3 CIM_PersistentMemoryCapabilities

CIM_PersistentMemoryCapabilities describes the capabilities of pool(s) of persistent memory. Capabilities can be derived from the underlying persistent memory devices, memory controllers, platform firmware, or other available software. CIM_PersistentMemoryCapabilities are described in Table 175.

Table 175 - Class: CIM_PersistentMemoryCapabilities

Element	Requirement	Notes
InstanceID	Mandatory	Key
ElementName	Mandatory	User friendly name for this capability instance
SecurityFeatues	Optional	Indicate support for security features.
AccessGranularity	Optional	Describes the access types supported by the pool.
MemoryArchitecture	Optional	Memory architectures supported e.g. NUMA.
Replication	Optional	Indicates support for various types of replication.

13.6.4 CIM_PersistentMemoryService

The CIM_PersistentMemoryService provides extrinsic methods suitable for configuring persistent memory. Implementations support attributes as given in Table 176.

Table 176 - Class: CIM_PersistentMemoryService

Element	Requirement	Notes
CreationClassName	Mandatory	Key
Name	Mandatory	Key
SystemCreationClassName	Mandatory	Key
SystemName	Mandatory	Key

13.6.5 CIM_PersistentConfigurationCapabilities

CIM_PersistentConfigurationCapabilities describes the capabilities of the CIM_PersistentMemoryService. The properties are documented in Table 176.

Table 177 - Class: CIM_PersistentConfigurationCapabilities

Element	Requirement	Notes
InstanceID	Mandatory	Key
ElementName	Mandatory	User friendly name for this capability instance
SupportedSynchronousOperations[]	Mandatory	Enumeration indicating the available synchronous memory provisioning operations. Synchronous operations result in an immediate change to the platform memory configuration.

Table 177 - Class: CIM_PersistentConfigurationCapabilities

Element	Requirement	Notes
SupportedAsynchronousOperations[]	Mandatory	Enumeration indicating the available asynchronous memory provisioning operations. Asynchronous operations stage a memory provisioning request which takes effect after some triggering action such as a system reboot.

13.6.6 CIM_PersistentMemoryNamespace

CIM_PersistentMemoryNamespace represents a system usable persistent memory allocation. CIM_PersistentMemoryNamespace properties are described in Table 13.6.6.

Table 178 - Class: CIM_PersistentMemoryNamespace

Element	Requirement	Notes
CreationClassName	Mandatory	Key
DeviceID	Mandatory	Key
SystemCreationClassName	Mandatory	Key
SystemName	Mandatory	Key
Volatile	Mandatory	false
Name	Optional	Friendly name
HealthState	Mandatory	
OperationalStatus	Mandatory	
EnabledState	Optional	When not supported the device is presumed to always be enabled.
NumberOfBLocks	Mandatory	
BlockSize	Mandatory	In bytes. Note that block size can be 1 for byte addressable PM.

13.6.7 CIM_PersistentMemoryNamespaceSettingData

CIM_PersistentMemoryNamespaceSettingData represents the parameters used when the namespace was allocated or modified. CIM_PersistentMemoryNamespaceSettingData properties are described in Table 179.g.

Table 179 - Class: CIM_PersistentMemoryNamespaceSettingData

Element	Requirement	Notes
InstanceID	Mandatory	Key
ElementName	Optional	Use friendly name for this setting instance
AllocationUnits	Mandatory	Specifies the units of the Reservation capacity property. Generally expected to be “bytes”.

Table 179 - Class: CIM_PersistentMemoryNamespaceSettingData

Element	Requirement	Notes
Reservation	Mandatory	Requested capacity. The units of this value are given by the AllocationUnits field.
PoolID	Mandatory	The PoolID of the CIM_ResourcePool instance from which this allocation will/has drawn resources
ChannelCount	Optional	Boolean
ResourceType	Optional	Indicates whether the extent is byte addressable (“Non-Volatile Memory” and accessed like memory or block addressable (“Storage Volume”) and accessed like storage.
Parent	Optional	If used, this value identifies a specific resource within the pool that shall/has been used when making an allocation. The default value is an empty string.
AccessType	Optional	Read, Write, or Read/Write
Optimize	Optional	Other requested characteristics of the allocation.

13.6.8 CIM_ElementCapabilities

13.6.8.1 Relating PersistentMemoryService to PersistentConfigurationCapabilities

The ElementCapabilities association links the PersistentConfigurationCapabilities class that describes the available operations to the PersistentMemoryService class that offers the operations.Properties are described in Table 13.6.8.

Table 180 - CIM_ElementCapabilities

Element	Requirement	Notes
ManagedElement	Mandatory	This property shall be a reference to an instance of the CIM_PersistentMemoryService class.
Capabilities	Mandatory	This property shall be a reference to an instance of the CIM_PersistentConfigurationCapabilities class.

13.6.8.2 Relating (Persistent Memory) ResourcePool to CIM_PersistentMemoryCapabilities

The ElementCapabilities association links the CIM_PersistentMemoryCapabilities class that describes the available operations to the ResourcePool class that offers the operations. Properties are described in Table 13.6.8.

Table 181 - CIM_ElementCapabilities

Element	Requirement	Notes
ManagedElement	Mandatory	This property shall be a reference to an instance of the CIM_ResourcePool class.
Capabilities	Mandatory	This property shall be a reference to an instance of the CIM_PersistentConfigurationCapabilities class.

13.6.9 CIM_ServiceAffectsElement

The CIM_ServiceAffectsElement association is used to relate the CIM_PersistentMemoryService to the CIM_ResourcePools which it is able to act upon. Table 13.6.9 contains the requirements for elements of this class.

Table 182 - Class: CIM_ServiceAffectsElement

Element	Requirement	Notes
AffectedElement	Mandatory	This property shall be a reference to an instance of the CIM_ResourcePool class.
AffectingElement	Mandatory	This property shall be a reference to an instance of the CIM_PersistentMemoryService class.

13.6.10CIM_ConcreteComponent

The CIM_ConcreteComponent association is used to relate the CIM_VisibleMemory to the CIM_ResourcePool of which it is part. Table 183 contains the requirements for elements of this class.

Table 183 - Class: CIM_ConcreteComponent

Element	Requirement	Notes
GroupComponent	Mandatory	This property shall be a reference to an instance of the CIM_ResourcePool class.
PartComponent	Mandatory	This property shall be a reference to an instance of the CIM_VisibleMemory class.

13.6.11CIM_ElementAllocatedFromPool

The CIM_ElementAllocatedFromPool association relates PersistentMemoryNamespace instances to the ResourcePool from which they were allocated. The properties of CIM_ElementAllocatedFromPool are described in.

Table 184 - Class: CIM_ElementAllocatedFromPool

Element	Requirement	Notes
Antecedent	Mandatory	This property shall be a reference to an instance of the CIM_ResourcePool class.
Dependent	Mandatory	This property shall be a reference to an instance of the CIM_PersistentMemoryNamespace class.

13.6.12CIM_HostedService

The CIM_HostedService association is used to relate an instance of CIM_ComputerSystem with a CIM_PersistentMemoryService instance. Table 185 contains the requirements for elements of this class.

Table 185 - Class: CIM_HostedService

Element	Requirement	Notes
Antecedent	Mandatory	This property shall be a reference to an instance of CIM_ComputerSystem.
Dependent	Mandatory	This property shall be a reference to an instance of the CIM_PersistentMemoryService class.

13.6.13CIM_ElementSettingData

CIM_ElementSettingData links a CIM_PersistentMemoryNamespace to the CIM_PersistentMemoryNamespaceSettingData data that defines the quality of service attributes of the namespace. The properties are described in Table 186.

Table 186 - Class CIM_ElementSettingData

Element	Requirement	Notes
SettingData	Mandatory	This property shall be a reference to an instance of the CIM_PersistentMemoryNamespaceSettingData
ManagedElement	Mandatory	This property shall be a reference to an instance of CIM_PersistentMemoryNamespace.
IsCurrent	Optional	Used to indicate a setting that is currently in-use (e.g. associated with an existing memory extent). In most cases this should be true for this usage.
IsNext	Optional	Used to indicate a requested setting that has not yet taken effect. In most cases this should be false for this usage.
IsPending	Optional	Maybe used in conjunction with IsNext to indicate the need for a reset or other event before the setting is in force. In most cases this should be false for this usage.

13.6.14CIM_ElementConformsToProfile

The CIM_ElementConformsToProfile association links the central class (CIM_PersistentMemoryService) and to the system hosting the service (CIM_ComputerSystem). Table 187 described the properties of CIM_ElementConformsToProfile.

Table 187 - Class: CIM_ElementConformsToProfile

Element	Requirement	Notes
ConformantStandard	Mandatory	This property shall be a reference to an instance of CIM_RegisteredProfile
ManagedElement	Mandatory	This property shall be a reference to an instance of either CIM_ComputerSystem or CIM_PersistentMemoryService.

13.6.15CIM_SystemDevice

CIM_SystemDevice is used to link CIM_PersistentMemoryNamespace instances to the CIM_ComputerSystem where they are to be used. The properties of SystemDevice are described in Table 13.6.15.

Table 188 - Class: CIM_SystemDevice

Element	Requirement	Notes
ConformantStandard	Mandatory	This property shall be a reference to an instance of CIM_RegisteredProfile
ManagedElement	Mandatory	This property shall be a reference to an instance of either CIM_ComputerSystem CIM_PersistentMemoryService.

13.6.16CIM_BasedOn

CIM_BasedOn identifies the CIM_VisibleMemory regions upon which the associated CIM_PersistentMemoryNamespace is built. Usage is conditional upon the existence of a CIM_PersistentMemoryNamespace. Any usage of properties beyond those given in Table 189 is not specified by this document.

Table 189 - Class: CIM_BasedOn

Element	Requirement	Notes
ConformantStandard	Mandatory	This property shall be a reference to an instance of CIM_RegisteredProfile
ManagedElement	Mandatory	This property shall be a reference to an instance of either CIM_ComputerSystem or CIM_PersistentMemoryService.

EXPERIMENTAL

Annex A (informative) SMI-S Information Model

This standard is based on DMTF’s CIM schema, version 2.45.0. The DMTF schema is available in the machine-readable Managed Object Format (MOF) format. DMTF MOFs are simultaneously released both as an "Experimental" and a "Final" version of the schema. This provides developers with early access to experimental parts of the models. Both versions are available at

http://www.dmtf.org/standards/cim

Content marked as “Experimental” or “Implemented” may be based on DMTF’s Experimental MOFs.

EXPERIMENTAL

Annex A (Informative) Host Profile Deployment Guidelines

A.1 Introduction

This annex presents background information and guidelines related to deployment of profiles for host-based storage components.

A.2 Background - Early SMI-S Host Profiles

CIM-based solutions for hosts are generally designed with a common CIM Object Manager (i.e., CIMOM). To support this approach, providers may be deployed independently from the CIMOM. SMI-S deployments for devices external to hosts (arrays, switches, tape libraries) are generally designed as single-vendor solutions and are deployed as a monolithic package including the CIMOM and providers for the particular device. SMI-S external device profiles are optimized for this approach - with an autonomous profile containing a “top-level” System instance, with non-system elements defined in that autonomous profile and its supporting component profiles.

The FC HBA Profile was created in SMI-S 1.0 following the monolithic agent approach. The “top-level” System in the FC HBA profile represents the host containing the HBAs supported by the profile.

A.3 Limitations of Monolithic Agents for Host Storage

Different host storage profiles may be referring to the same real-world instances (e.g., the ports in HBA profiles are the same ports in Host Discovered Resources, and logical disks expressed through Host Discovered Resources may have host file systems). Modeling these as independent providers under the same CIMOM adds extra work for clients to recognize that separate CIM instances represent the same entity. Deploying these providers in separate CIMOMs is potentially even more complex for clients.

SMI-S is encouraging use of DMTF profiles for non-storage elements and eliminating the need for separate SMI-S and DMTF profiles for the same components.

A.4 Shared ComputerSystem Approach

SMI-S is now following the direction defined by DMTF with a ComputerSystem instance representing the host system defined in a device-independent autonomous profile. This will be the Base Server Profile. Newer profiles for storage functionality are defined as component profiles supported by the Base Server profile.

In some cases, vendors will prefer the monolithic approach. See B.6.1 for guidelines on deploying monolithic agents using the new profiles.

Some SMI-S profiles do not fit well in the shared ComputerSystem Approach.

- The Software Profile may be used to model software or firmware versions, but requires the implementation to provide the software version of the ComputerSystem. Although this works well for external devices, it is not appropriate for storage instrumentation to provide its information. Host profiles should use the Software Inventory Profile
- The Physical Package Package may be used to model the physical aspects of a system, but requires the implementation to model physical information related to the ComputerSystem. Instrumentation for storage components may not have access to information about the containing server. Host profiles should use the Physical Asset Profile instead.

A.5 Overview of Host Storage Models

A.5.1 Combined Profiles

B.5 and its subsections provide an overview of several SMI-S profiles, showing the differences between the FC HBA monolithic agent approach and the profiles following the shared ComputerSystem approach.

The figures in B.5 subsections provide examples of how the host storage profiles, the Base Server Profile, and the Profile Registration Profile are combined for host storage. The figures here do not provide details of the profiles discussed; these details can be found elsewhere in this standard. A cloud labeled “Other Storage Classes” in the figures is used to represent the details.

A.5.2 FC HBA Profile

The FC HBA Profile was defined following the monolithic agent approach and was implemented by vendors before the shared ComputerSystem approach was defined. See Figure B.1.

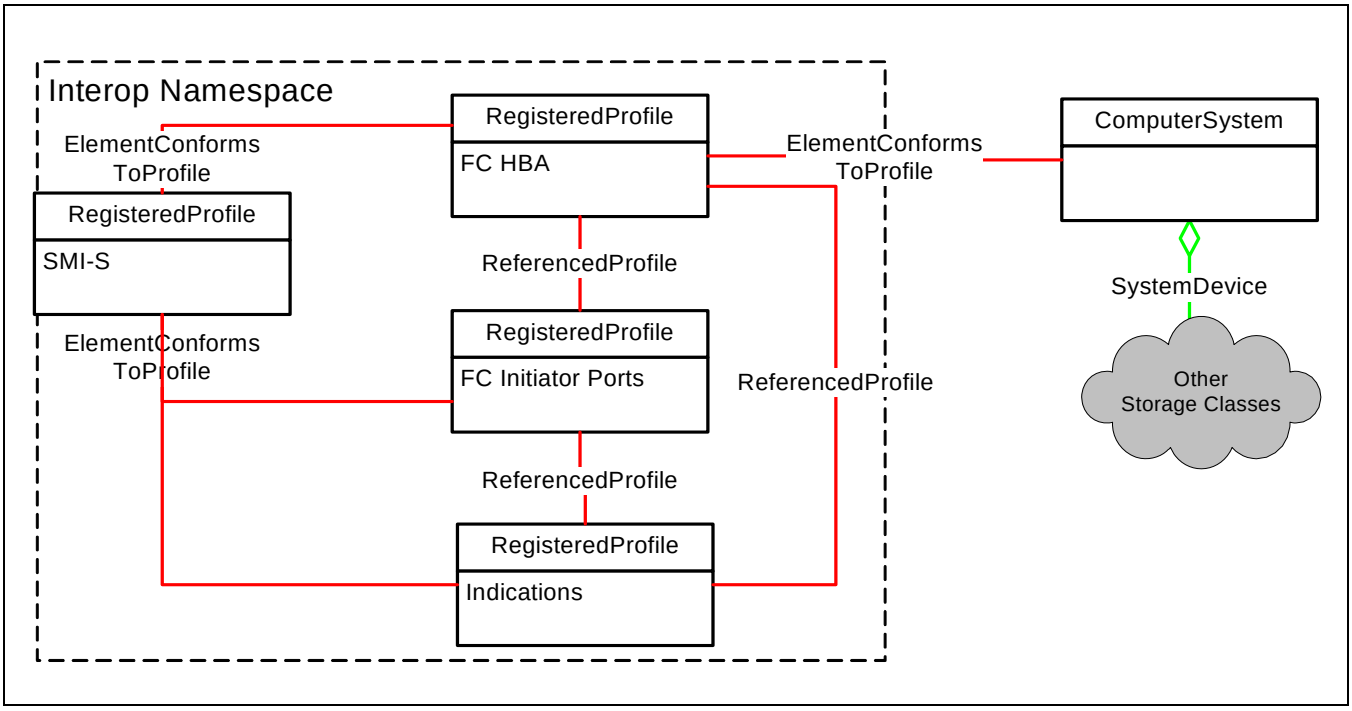

Figure B.1 - Profile Registration and FC HBA Profiles

If multiple FC HBA providers were installed on the same system, each would have its own instance of ComputerSystem representing the host system.

To maintain compatibility, FC HBA will continue to be defined as an autonomous profile for all SMI-S 1.x releases. The FC HBA profile may not be supported in future versions of SMI-S; vendors should migrate to the Base Server and Storage HBA profiles. See B.6.3 How to Deploy FC HBA with New Host Profiles for guidelines on incorporating FC HBA with newer host profiles.

A.5.3 Storage HBA Profile

The Storage HBA profile is the successor to the FC HBA profile. It has been redefined as a component profile of the Base Server Profile. It supports FCoE, SAS, parallel SCSI, or ATA as well as FC ports. See Figure B.2.

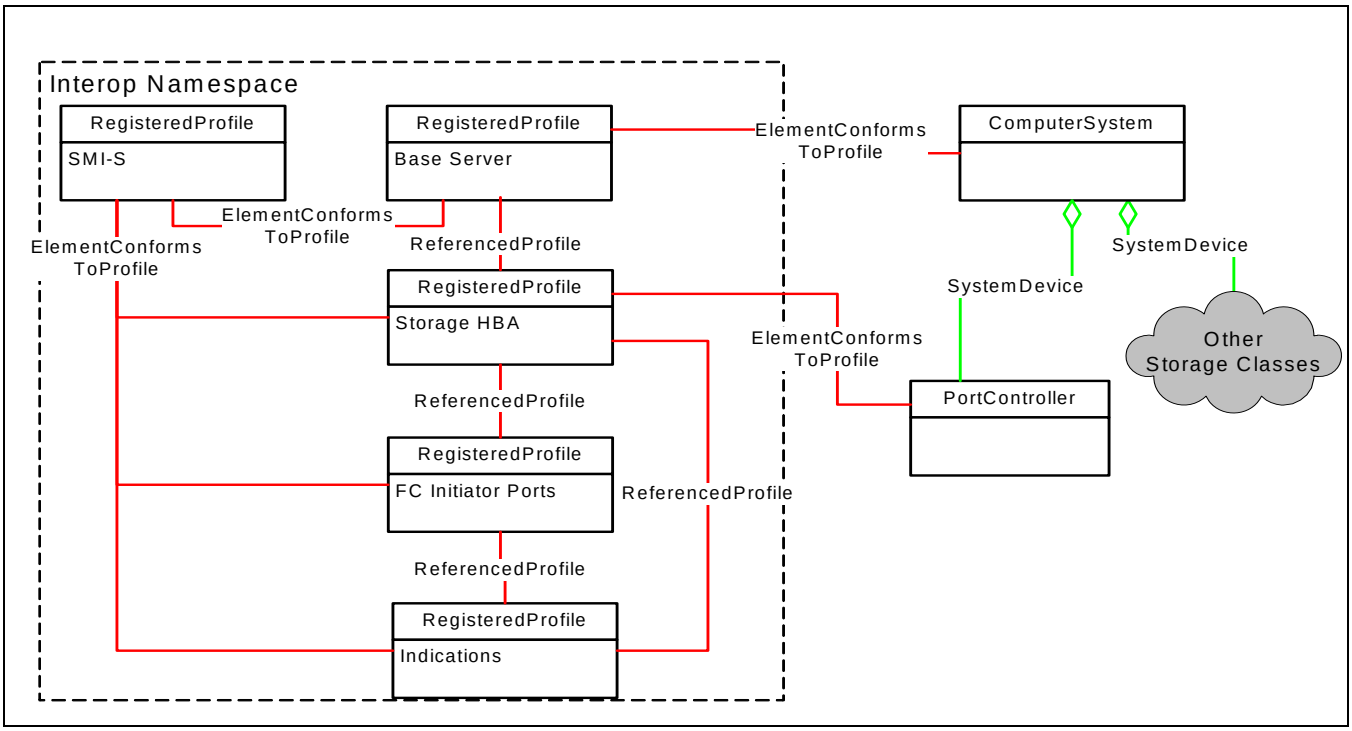

Figure B.2 - Profile Registration and Storage HBA Profiles

A.5.4 Host Hardware RAID Controller Profile

The Host Hardware RAID Controller Profile is a component profile of the Base Server profile. Unlike other profiles supporting the shared ComputerSystem approach, the Host Hardware RAID Controller Profile includes a ComputerSystem instance representing the RAID controller. This ComputerSystem instance provides compatibility with SMI-S profiles related to RAID in arrays. All associations other than SystemComponent which references ComputerSystem are referencing the controller ComputerSystem. SystemComponent references the Base Server Profile ComputerSystem. See Figure B.3.

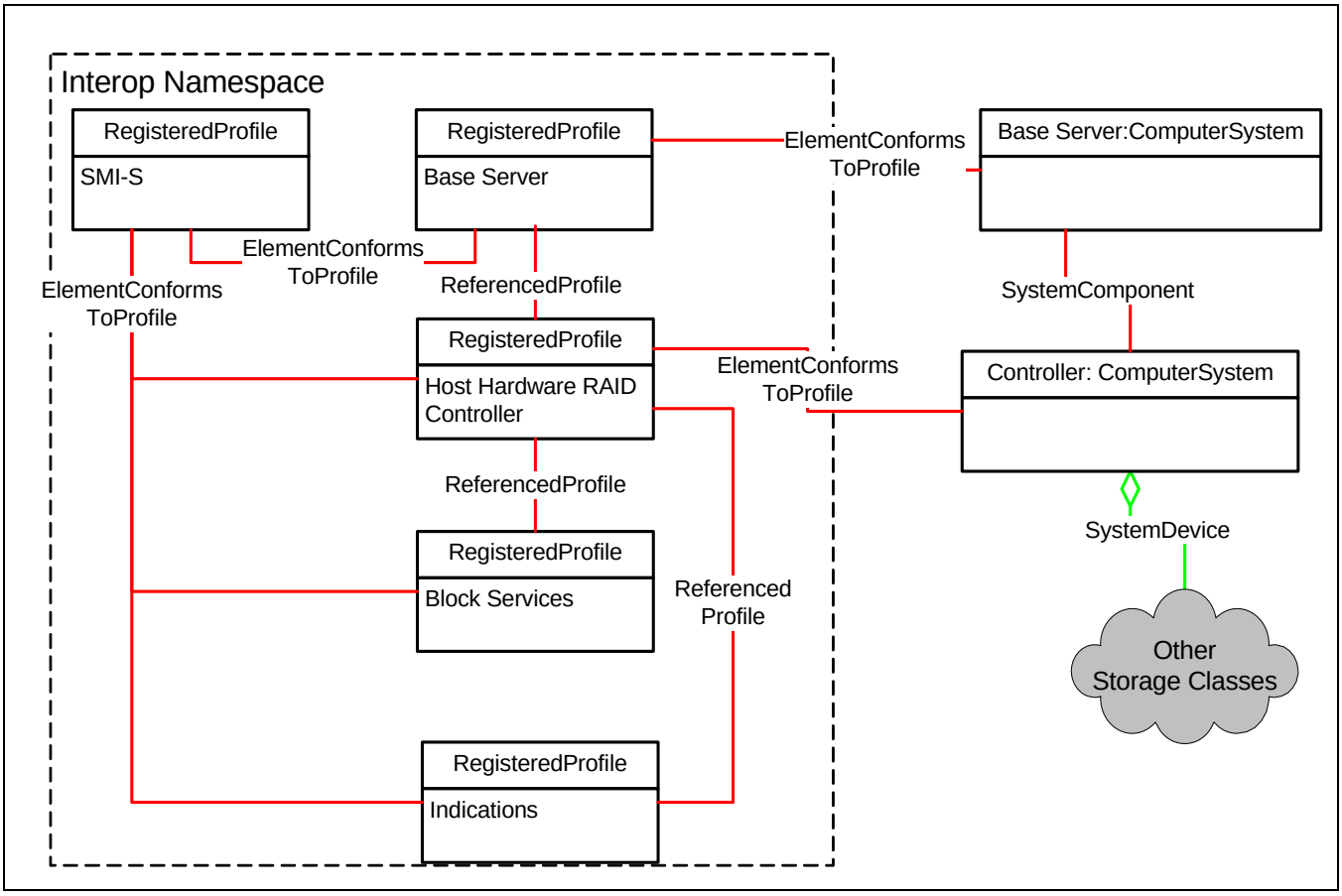

Figure B.3 - Profile Registration and Host Hardware RAID Controller Profiles

A.5.5 Other Host Storage Profiles

The Host Discovered Resources, SCSI Multipath Management, Disk Partition, and Host FIlesystem profiles are all defined to follow the shared ComputerSystem approach, similar to the Storage HBA Profile (see B.5.3 Storage HBA Profile).

A.6 Deployment Guidelines

A.6.1 Emulating the Monolithic Agent Approach

To emulate the monolithic agent approach using profiles designed for the shared ComputerSystem approach, implement the Base Server Profile and the appropriate host storage profiles. You also need to implement the Server and Profile Registration Profiles.

A.6.2 Platform Vendor Supporting Device Partners

Platform vendors may wish to provide an integrate CIM management environment for a variety of storage (and possibly non-storage) components.

A.6.3 How to Deploy FC HBA with New Host Profiles

To deploy the FC HBA Profile with profiles using the shared ComputerSystem approach, use the same ComputerSystem instance for FC HBA and Base Server Profiles. Figure B.4 depicts this approach.

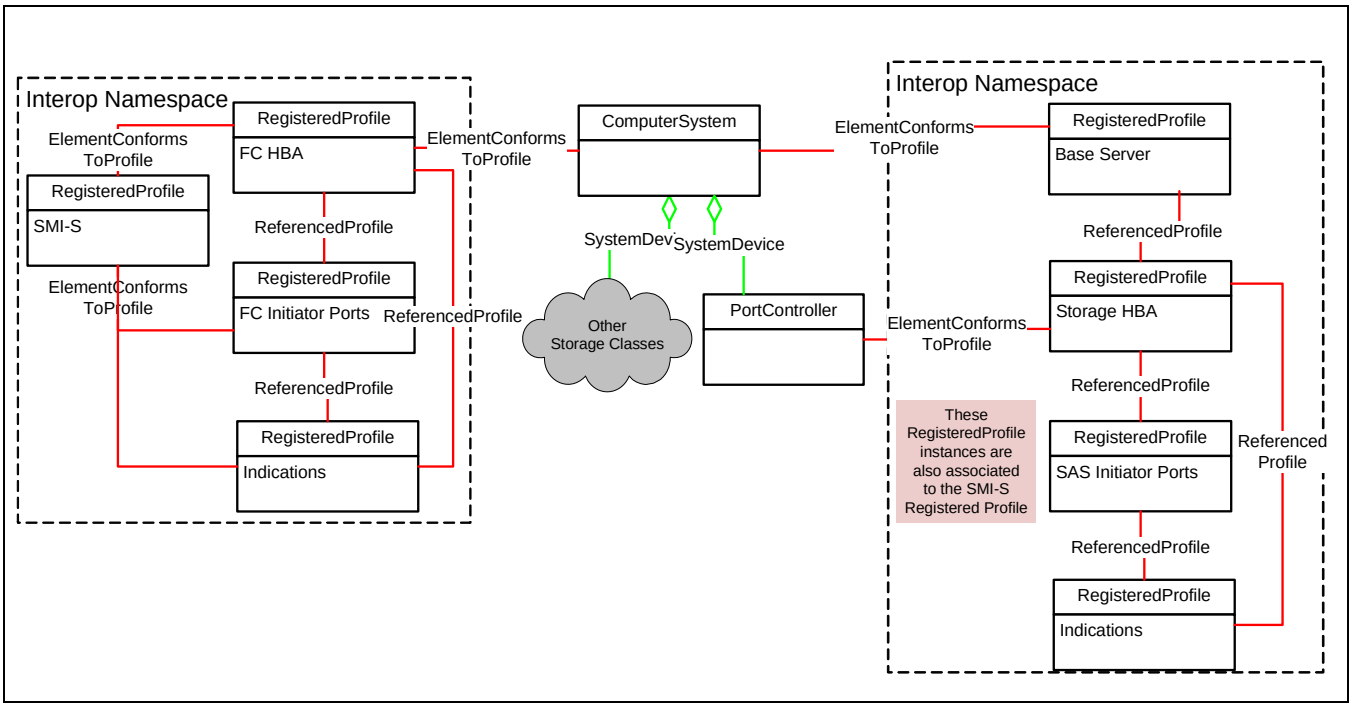

Figure B.4 - Deploying FC HBA with Storage HBA (SAS) Profiles

A.7 Client Discovery of Host Storage Profiles

A typical SMI-S client approach is to start discovering the resources provided by a CIMOM by first locating instances of RegisteredProfile for supported autonomous profiles, then following the ElementConformsToProfile association to instances of “top-level” Systems implementing these profiles. For external devices, autonomous profiles correspond to classes of devices - arrays, switches, tape libraries, etc.

For host profiles using the shared ComputerSystem approach, the autonomous profile is the Base Server profile; its component profiles model classes of host storage systems - HBAs, Host Filesystems, etc. To optimize this client discovery task, profiles using the shared ComputerSystem approach also include an ElementConformsToProfile instance between the RegisteredProfile for key component profiles and an instance from these profile. For example, the Storage HBA Profile includes ElementConformsToProfile between its RegisteredProfile and PortController instances.

The component profiles that follow this approach are advertised through SLP’s Supported Profiles list.

EXPERIMENTAL