



# **Storage Management Technical Specification, Part 6 Host Elements**

**Version 1.4.0, Revision 6**

*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***

***17 May 2010***



## Revision History

### Revision 1

**Date**

5 December 2007

**SCRs Incorporated and other changes**

None.

**Comments**

Editorial notes and DRAFT material are displayed.

### Revision 2

**Date**

29 February 2008

**SCRs Incorporated and other changes**

Clause 5: FC HBA Profile (SMIS-140-Draft-SCR00008)

- Figure in this is updated to identify FCPortStatistics as part of FCInitiatorPorts.

**Comments**

Editorial notes and DRAFT material are displayed.

### Revision 3

**Date**

23 May 2008

**SCRs Incorporated and other changes**

FC HBA Profile (SMIS-140-Draft-SCR00008)

- Updated FC HBA to remove statistics (now in FC Initiator Ports profile)

Storage HBA Profile (SMIS-140-Draft-SCR00014)

- Added alert indications to the StorageHBA profile

Host Hardware RAID Controller Profile (SMIS-140-Draft-SCR00014)

- Added alert indications to the Host Hardware RAID Controller profile

**Comments**

Editorial notes are displayed.

### Revision 4

**Date**

23 October 2008

**SCRs Incorporated and other changes**

Host Hardware RAID Controller Profile (SMIS-130-Errata-SCR00019)

- Fixed typo's in the CIM Elements tables
- Clarified that the hosting system is an HHRC system (as opposed to the Host System)
- Replaced CIM\_Realizes table with a CIM\_ComputerSystemPackage table

SCSI Multipath Management Subprofile (SMIS-130-Errata-SCR00014)

- Corrected a mislabeled association in the Multipath Management Class Diagram
- Corrected a misspelled association in the Four Path Instance Diagram
- Changed "SetOverRidePath" to "SetOverridePath"
- Changed "CencelOverRidePath" to "CancelOverridePath" and LogicalUnit to Path to match corresponding SetOverridePath method parameter.

#### **Comments**

Editorial notes and DRAFT material are not displayed.

## **Revision 5**

#### **Date**

9 December 2009

#### **SCRs Incorporated and other changes**

Host Hardware Raid Controller (HHRC) profile (SMIS-140-Errata-SCR00040)

- A number of fixes based on content review, including diagrams and HHRC\_TestAlarm recipe.

#### **Comments**

The errata SCRs listed above indicate changes made since this part of SMI-S 1.4 Revision 4 was reviewed and approved by SNIA membership and subsequently published as a SNIA Technical Position.

SMIS-140-Errata-SCR00001 through SMIS-140-Errata-SCR00060 (approved SCRs) have been incorporated into SMI-S 1.4 Revision 5.

## **Revision 6**

#### **Date**

17 May 2010

#### **SCRs Incorporated and other changes**

Host Hardware Raid Controller (HHRC) profile (SMIS-140-Errata-SCR00061)

- SystemComponent references were reversed in the CIM Elements Table
- The SystemDevice association referencing PortController were changed to reference the controller ComputerSystem.
- HHRC was spelled out in the CIM Elements tables.

(SMIS-140-Errata-SCR00066)

- Fixed incorrect and confusing descriptions in Host Hardware RAID Controller LogicalIdentity CIM Element Table.

#### **Comments**

The errata SCRs listed above indicate changes made since this part of SMI-S 1.4 Revision 4 was reviewed and approved by SNIA membership and subsequently published as a SNIA Technical Position.

SMIS-140-Errata-SCR00061 through SMIS-140-Errata-SCR00067 (approved SCRs) have been incorporated into SMI-S 1.4 Revision 6.

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/>.

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 must 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 must acknowledge the SNIA copyright on that material, and must 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](mailto: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.

Copyright © 2003-2010 Storage Networking Industry Association.



## 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.

## 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/>.

Copyright © 2003-2010 SNIA. All rights reserved. All other trademarks or registered trademarks are the property of their respective owners.

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.

## 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 1.2.0. 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

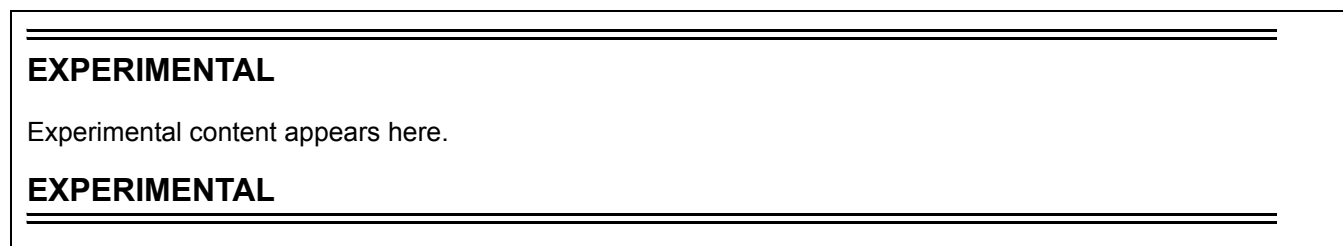
This specification has been structured to convey both the formal requirements and assumptions of the SMI-S API and its emerging implementation and deployment lifecycle. Over time, the intent is that all content in the specification will represent a mature and stable design, be verified by extensive implementation 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

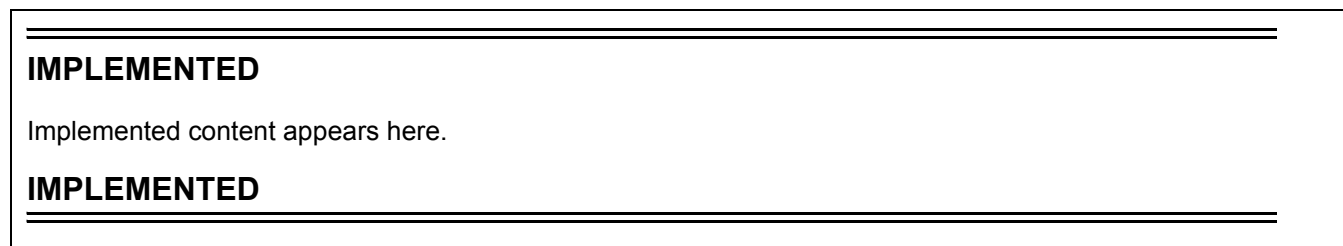
No material is included in this specification unless its initial architecture has been completed and reviewed. Some content included in this specification has complete and reviewed design, but lacks implementation experience and the maturity gained through implementation experience. This content is included 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.



**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.



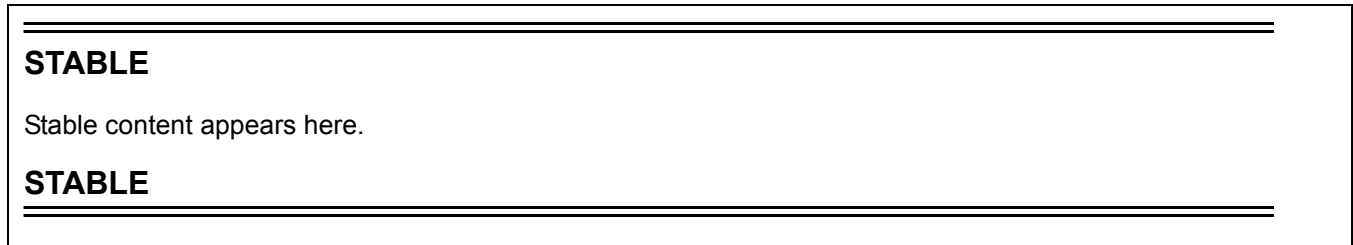
**Figure 2 - Implemented Maturity Level Tag**

### 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.



**Figure 3 - Stable Maturity Level Tag**

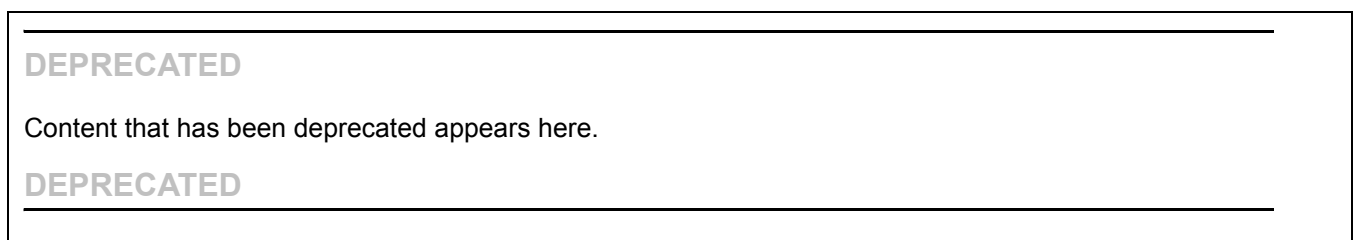
**Finalized Maturity Level**

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.



**Figure 4 - Deprecated Tag**

## 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.
- 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](mailto: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.

# Contents

Revision History.....	iii
List of Tables.....	xiii
List of Figures.....	xvii
Foreword.....	xix
<b>1. Scope.....</b>	<b>1</b>
<b>2. Normative References.....</b>	<b>3</b>
2.1 Approved references.....	3
2.2 References under development.....	3
2.3 Other references.....	3
<b>3. Terms and definitions.....</b>	<b>5</b>
3.1 General.....	5
3.2 Definitions.....	5
<b>4. Disk Partition Subprofile.....</b>	<b>7</b>
4.1 Description.....	7
4.2 Health and Fault Management Considerations.....	12
4.3 Supported Subprofiles and Packages.....	13
4.4 Methods of the Profile.....	13
4.5 Client Considerations and Recipes.....	14
4.6 Registered Name and Version.....	17
4.7 CIM Elements.....	17
<b>5. FC HBA Profile.....</b>	<b>27</b>
5.1 Synopsis.....	27
5.2 Description.....	27
5.3 Implementation.....	27
5.4 Health and Fault Management.....	30
5.5 Methods.....	31
5.6 Client Considerations and Recipes.....	33
5.7 CIM Elements.....	41
<b>6. Storage HBA Profile.....</b>	<b>57</b>
6.1 Synopsis.....	57
6.2 Description.....	57
6.3 Implementation.....	57
6.4 Methods of the Profile.....	63
6.5 Use Cases.....	63
6.6 CIM Elements.....	64
<b>7. Host Discovered Resources Profile.....</b>	<b>69</b>
7.1 Description.....	69
7.2 Health and Fault Management Considerations.....	74
7.3 Cascading Considerations.....	74
7.4 Supported Subprofiles and Packages.....	74
7.5 Extrinsic Methods of the Profile.....	74
7.6 Client Considerations and Recipes.....	75
7.7 Registered Name and Version.....	76
7.8 CIM Elements.....	76
<b>8. Host Hardware RAID Controller Profile.....</b>	<b>87</b>
8.1 Synopsis.....	87
8.2 Description.....	88
8.3 Implementation.....	88
8.4 Methods.....	100
8.5 Use Cases.....	100
8.6 CIM Elements.....	101
<b>9. iSCSI Initiator Profile.....</b>	<b>113</b>

9.1	Description .....	113
9.2	Health and Fault Management Considerations .....	116
9.3	Supported Subprofiles and Packages .....	116
9.4	Methods of the Profile .....	116
9.5	Client Considerations and Recipes .....	116
9.6	Registered Name and Version .....	117
9.7	CIM Elements .....	117
<b>10.</b>	<b>SCSI Multipath Management Subprofile .....</b>	<b>131</b>
10.1	Description .....	131
10.2	Health and Fault Management Considerations .....	133
10.3	Cascading Considerations .....	133
10.4	Supported Subprofiles and Packages .....	133
10.5	Methods of the Profile .....	133
10.6	Client Considerations and Recipes .....	134
10.7	Registered Name and Version .....	140
10.8	CIM Elements .....	140
<b>11.</b>	<b>SB Multipath Management Profile .....</b>	<b>149</b>
11.1	Description .....	149
11.2	Health and Fault Management Consideration .....	150
11.3	Cascading Considerations .....	150
11.4	Supported Profiles, Subprofiles, and Packages .....	150
11.5	Methods .....	150
11.6	Client Considerations and Recipes .....	151
11.7	Registered Name and Version .....	151
11.8	CIM Elements .....	151

## List of Tables

Table 1.	Capabilities Properties.....	11
Table 2.	CIM Elements for Disk Partition.....	17
Table 3.	SMI Referenced Properties/Methods for CIM_BasedOn (Partition to Extent).....	18
Table 4.	SMI Referenced Properties/Methods for CIM_BasedOn (Partition to Partition).....	19
Table 5.	SMI Referenced Properties/Methods for CIM_DiskPartitionConfigurationCapabilities .....	19
Table 6.	SMI Referenced Properties/Methods for CIM_DiskPartitionConfigurationService .....	20
Table 7.	SMI Referenced Properties/Methods for CIM_ElementCapabilities .....	20
Table 8.	SMI Referenced Properties/Methods for CIM_GenericDiskPartition .....	21
Table 9.	SMI Referenced Properties/Methods for CIM_HostedService .....	21
Table 10.	SMI Referenced Properties/Methods for CIM_InstalledPartitionTable (Capabilities to Extent).....	22
Table 11.	SMI Referenced Properties/Methods for CIM_InstalledPartitionTable (Capabilities to Partition).....	22
Table 12.	SMI Referenced Properties/Methods for CIM_LogicalDisk .....	22
Table 13.	SMI Referenced Properties/Methods for CIM_LogicalDiskBasedOnPartition .....	23
Table 14.	SMI Referenced Properties/Methods for CIM_StorageExtent.....	23
Table 15.	SMI Referenced Properties/Methods for CIM_SystemDevice (System to Extent) .....	24
Table 16.	SMI Referenced Properties/Methods for CIM_SystemDevice (System to LogicalDisk).....	24
Table 17.	SMI Referenced Properties/Methods for CIM_SystemDevice (System to Partition) .....	25
Table 18.	Related Profiles for FC HBA.....	27
Table 19.	CIM Elements for FC HBA.....	41
Table 20.	SMI Referenced Properties/Methods for CIM_AlarmDevice .....	42
Table 21.	SMI Referenced Properties/Methods for CIM_AssociatedAlarm.....	43
Table 22.	SMI Referenced Properties/Methods for CIM_ComputerSystem.....	43
Table 23.	SMI Referenced Properties/Methods for CIM_ControlledBy .....	44
Table 24.	SMI Referenced Properties/Methods for CIM_ElementCapabilities (Capabilities to FCPort) .....	44
Table 25.	SMI Referenced Properties/Methods for CIM_ElementCapabilities (Capabilities to System).....	45
Table 26.	SMI Referenced Properties/Methods for CIM_ElementSettingData.....	45
Table 27.	SMI Referenced Properties/Methods for CIM_ElementSoftwareIdentity (Driver).....	45
Table 28.	SMI Referenced Properties/Methods for CIM_ElementSoftwareIdentity (FCode/BIOS).....	46
Table 29.	SMI Referenced Properties/Methods for CIM_ElementSoftwareIdentity (Firmware) .....	46
Table 30.	SMI Referenced Properties/Methods for CIM_FCPort .....	46
Table 31.	SMI Referenced Properties/Methods for CIM_HostedCollection .....	47
Table 32.	SMI Referenced Properties/Methods for CIM_HostedService .....	47
Table 33.	SMI Referenced Properties/Methods for CIM_InstalledSoftwareIdentity (Driver) .....	48
Table 34.	SMI Referenced Properties/Methods for CIM_LogicalPortGroup.....	48
Table 35.	SMI Referenced Properties/Methods for CIM_MemberOfCollection.....	48
Table 36.	SMI Referenced Properties/Methods for CIM_OSSStorageNameBinding .....	49
Table 37.	SMI Referenced Properties/Methods for CIM_PhysicalPackage .....	50
Table 38.	SMI Referenced Properties/Methods for CIM_PortController .....	50
Table 39.	SMI Referenced Properties/Methods for CIM_Product .....	51
Table 40.	SMI Referenced Properties/Methods for CIM_ProductPhysicalComponent .....	51
Table 41.	SMI Referenced Properties/Methods for CIM_Realizes.....	51
Table 42.	SMI Referenced Properties/Methods for CIM_ServiceAvailableToElement.....	52
Table 43.	SMI Referenced Properties/Methods for CIM_SoftwareIdentity (Driver).....	52
Table 44.	SMI Referenced Properties/Methods for CIM_SoftwareIdentity (FCode/BIOS).....	53
Table 45.	SMI Referenced Properties/Methods for CIM_SoftwareIdentity (Firmware) .....	53
Table 46.	SMI Referenced Properties/Methods for CIM_StorageNameBinding .....	54
Table 47.	SMI Referenced Properties/Methods for CIM_StorageNameBindingCapabilities.....	54

Table 48.	SMI Referenced Properties/Methods for CIM_StorageNameBindingService .....	55
Table 49.	SMI Referenced Properties/Methods for CIM_SystemDevice (Associates System to PortController).....	55
Table 50.	Related Profiles for Storage HBA .....	57
Table 51.	CIM_PortController .....	63
Table 52.	CIM_SystemDevice .....	63
Table 53.	CIM Elements for Storage HBA .....	64
Table 54.	SMI Referenced Properties/Methods for CIM_ControlledBy .....	65
Table 55.	SMI Referenced Properties/Methods for CIM_PortController .....	65
Table 56.	SMI Referenced Properties/Methods for CIM_Product .....	66
Table 57.	SMI Referenced Properties/Methods for CIM_ProductElementComponent .....	66
Table 58.	SMI Referenced Properties/Methods for CIM_Realizes .....	66
Table 59.	SMI Referenced Properties/Methods for CIM_SystemDevice .....	67
Table 60.	Supported Profiles for Host Discovered Resources .....	74
Table 61.	CIM Elements for Host Discovered Resources .....	76
Table 62.	SMI Referenced Properties/Methods for CIM_ATAInitiatorTargetLogicalUnitPath .....	78
Table 63.	SMI Referenced Properties/Methods for CIM_ATAProtocolEndpoint .....	78
Table 64.	SMI Referenced Properties/Methods for CIM_ComputerSystem .....	79
Table 65.	SMI Referenced Properties/Methods for CIM_HostedAccessPoint .....	79
Table 66.	SMI Referenced Properties/Methods for CIM_LogicalDisk .....	80
Table 67.	SMI Referenced Properties/Methods for CIM_SCSIArbitraryLogicalUnit .....	80
Table 68.	SMI Referenced Properties/Methods for CIM_SCSIInitiatorTargetLogicalUnitPath .....	81
Table 69.	SMI Referenced Properties/Methods for CIM_SCSIProtocolEndpoint .....	82
Table 70.	SMI Referenced Properties/Methods for CIM_StorageExtent .....	82
Table 71.	SMI Referenced Properties/Methods for CIM_SystemDevice .....	83
Table 72.	SMI Referenced Properties/Methods for CIM_TapeDrive .....	83
Table 73.	SMI Referenced Properties/Methods for SNIA_SBInitiatorTargetLogicalUnitPath .....	84
Table 74.	SMI Referenced Properties/Methods for SNIA_SBProtocolEndpoint .....	84
Table 75.	Related Profiles for Host Hardware RAID Controller .....	87
Table 76.	CIM Elements for Host Hardware RAID Controller .....	101
Table 77.	SMI Referenced Properties/Methods for CIM_AlarmDevice .....	103
Table 78.	SMI Referenced Properties/Methods for CIM_AssociatedAlarm .....	103
Table 79.	SMI Referenced Properties/Methods for CIM_ComputerSystem (Host Hardware RAID Controller) .....	104
Table 80.	SMI Referenced Properties/Methods for CIM_ComputerSystemPackage (Associates PhysicalPackage to ComputerSystem) .....	104
Table 81.	SMI Referenced Properties/Methods for CIM_ControlledBy .....	105
Table 82.	SMI Referenced Properties/Methods for CIM_LogicalIdentity .....	105
Table 83.	SMI Referenced Properties/Methods for CIM_MediaAccessDevice .....	105
Table 84.	SMI Referenced Properties/Methods for CIM_PortController .....	106
Table 85.	SMI Referenced Properties/Methods for CIM_Product .....	106
Table 86.	SMI Referenced Properties/Methods for CIM_ProductPhysicalComponent .....	107
Table 87.	SMI Referenced Properties/Methods for CIM_ProtocolControllerForUnit (Extent or MediaAccessDevice) .....	107
Table 88.	SMI Referenced Properties/Methods for CIM_ProtocolControllerForUnit (Volume) .....	108
Table 89.	SMI Referenced Properties/Methods for CIM_Realizes (Associates PhysicalPackage to PortController) .....	108
Table 90.	SMI Referenced Properties/Methods for CIM_SAPAvailableForElement .....	109
Table 91.	SMI Referenced Properties/Methods for CIM_SCSIProtocolController .....	109
Table 92.	SMI Referenced Properties/Methods for CIM_SoftwareIdentity (Driver) .....	109
Table 93.	SMI Referenced Properties/Methods for CIM_SoftwareIdentity (FCode/BIOS) .....	110
Table 94.	SMI Referenced Properties/Methods for CIM_SoftwareIdentity (Firmware) .....	110
Table 95.	SMI Referenced Properties/Methods for CIM_StorageExtent .....	110

Table 96.	SMI Referenced Properties/Methods for CIM_SystemComponent .....	111
Table 97.	SMI Referenced Properties/Methods for CIM_SystemDevice (Associates System to AlarmDevice) .....	111
Table 98.	SMI Referenced Properties/Methods for CIM_SystemDevice (Associates controller system to PortController) ...	112
Table 99.	SMI Referenced Properties/Methods for CIM_SystemDevice (System to SCSIProtocolController) .....	112
Table 100.	iSCSI Terminology.....	113
Table 101.	OperationalStatus Values .....	116
Table 102.	Supported Profiles for iSCSI Initiator.....	116
Table 103.	CIM Elements for iSCSI Initiator .....	117
Table 104.	SMI Referenced Properties/Methods for CIM_BindsTo (TCPProtocolEndpoint to IPProtocolEndpoint).....	118
Table 105.	SMI Referenced Properties/Methods for CIM_BindsTo (iSCSIProtocolEndpoint to TCPProtocolEndpoint).....	118
Table 106.	SMI Referenced Properties/Methods for CIM_ComputerSystem.....	119
Table 107.	SMI Referenced Properties/Methods for CIM_ControlledBy .....	119
Table 108.	SMI Referenced Properties/Methods for CIM_DeviceSAPImplementation (EthernetPort to IPProtocolEndpoint)	120
Table 109.	SMI Referenced Properties/Methods for CIM_DeviceSAPImplementation (EthernetPort to iSCSIProtocolEndpoint). 120	
Table 110.	SMI Referenced Properties/Methods for CIM_ElementSoftwareIdentity.....	120
Table 111.	SMI Referenced Properties/Methods for CIM_EndpointOfNetworkPipe (Between iSCSI Session and iSCSIProtocolEndpoint).....	121
Table 112.	SMI Referenced Properties/Methods for CIM_EndpointOfNetworkPipe (Between iSCSI connection and TCPProtocolEndpoint) .....	121
Table 113.	SMI Referenced Properties/Methods for CIM_InstalledSoftwareIdentity .....	122
Table 114.	SMI Referenced Properties/Methods for CIM_NetworkPipeComposition .....	122
Table 115.	SMI Referenced Properties/Methods for CIM_PhysicalPackage .....	122
Table 116.	SMI Referenced Properties/Methods for CIM_PortController .....	123
Table 117.	SMI Referenced Properties/Methods for CIM_Product .....	123
Table 118.	SMI Referenced Properties/Methods for CIM_ProductPhysicalComponent .....	124
Table 119.	SMI Referenced Properties/Methods for CIM_ProtocolControllerForPort.....	124
Table 120.	SMI Referenced Properties/Methods for CIM_Realizes .....	124
Table 121.	SMI Referenced Properties/Methods for CIM_SAPAvailableForElement .....	125
Table 122.	SMI Referenced Properties/Methods for CIM_SCSIProtocolController .....	125
Table 123.	SMI Referenced Properties/Methods for CIM_SoftwareIdentity .....	126
Table 124.	SMI Referenced Properties/Methods for CIM_SystemDevice (to EthernetPort).....	126
Table 125.	SMI Referenced Properties/Methods for CIM_SystemDevice (to PortController) .....	126
Table 126.	SMI Referenced Properties/Methods for CIM_SystemDevice (to ProtocolController) .....	127
Table 127.	SMI Referenced Properties/Methods for CIM_iSCSIConnection .....	127
Table 128.	SMI Referenced Properties/Methods for CIM_iSCSISession.....	128
Table 129.	CIM Elements for SCSI Multipath Management.....	140
Table 130.	SMI Referenced Properties/Methods for CIM_ConcreteComponent .....	141
Table 131.	SMI Referenced Properties/Methods for CIM_ConcreteDependency.....	142
Table 132.	SMI Referenced Properties/Methods for CIM_ElementCapabilities .....	142
Table 133.	SMI Referenced Properties/Methods for CIM_ElementSettingData.....	143
Table 134.	SMI Referenced Properties/Methods for CIM_ElementSoftwareIdentity (Driver).....	143
Table 135.	SMI Referenced Properties/Methods for CIM_ElementSoftwareIdentity (MP API Plugin).....	143
Table 136.	SMI Referenced Properties/Methods for CIM_HostedService .....	144
Table 137.	SMI Referenced Properties/Methods for CIM_MemberOfCollection .....	144
Table 138.	SMI Referenced Properties/Methods for CIM_Product .....	144
Table 139.	SMI Referenced Properties/Methods for CIM_SCSIMultipathConfigurationCapabilities.....	145
Table 140.	SMI Referenced Properties/Methods for CIM_SCSIMultipathSettings.....	146
Table 141.	SMI Referenced Properties/Methods for CIM_SCSIPathConfigurationService .....	146
Table 142.	SMI Referenced Properties/Methods for CIM_SCSITargetPortGroup .....	147

Table 143. SMI Referenced Properties/Methods for CIM_ServiceAvailableToElement.....	147
Table 144. SMI Referenced Properties/Methods for CIM_SoftwareIdentity (Driver).....	148
Table 145. SMI Referenced Properties/Methods for CIM_SoftwareIdentity (MP API Plugin) .....	148
Table 146. CIM Elements for SB Multipath Management .....	151
Table 147. SMI Referenced Properties/Methods for SNIA_SBIInitiatorTargetLogicalUnitPath.....	152



## List of Figures

Figure 1. Experimental Maturity Level Tag .....	viii
Figure 2. Implemented Maturity Level Tag.....	viii
Figure 3. Stable Maturity Level Tag .....	ix
Figure 4. Deprecated Tag .....	ix
Figure 5. Disk Partition Class Hierarchy .....	8
Figure 6. Disk Partition Class Diagram .....	8
Figure 7. Disk MBR Partition Example.....	9
Figure 8. MBR Partition Instance Diagram.....	10
Figure 9. MBR and VTOC Partition Instance Diagram.....	11
Figure 10. Partition Instance Diagram for Size/Address Rules.....	12
Figure 11. FC HBA Instance Diagram.....	28
Figure 12. HBA Card with Two Ports .....	29
Figure 13. Persistent Binding Model .....	30
Figure 14. Model Overview .....	58
Figure 15. Profile Registration Profile .....	59
Figure 16. Software Inventory Profile in Storage HBA.....	61
Figure 17. HBA Card with Physical Classes .....	62
Figure 18. Host Discovered Resources Block Diagram .....	69
Figure 19. Host Discovered Resources Class Diagram.....	71
Figure 20. Single SPI Disk Model .....	72
Figure 21. Three FCP Logical Unit Instance Diagram .....	72
Figure 22. ATA Discovered Resource Model.....	73
Figure 23. SB Host Discovered Resources.....	73
Figure 24. Host Hardware RAID Controller Package Diagram .....	88
Figure 25. Alarms in Host Hardware RAID Controller.....	89
Figure 26. Profile Registration with Host Hardware RAID Controller and Base Server Profiles .....	90
Figure 27. Implementation of Physical Asset Profile.....	90
Figure 28. Block Services Package in Host Hardware RAID Controller .....	91
Figure 29. DAPort Subprofile in Host Hardware Controller.....	93
Figure 30. Software Inventory Profile in Host Hardware RAID Controller.....	93
Figure 31. Initiator Port profiles and Disk Drive Lite Subprofile.....	94
Figure 32. Model for Imported Disks .....	95
Figure 33. Imported Virtual Volumes.....	96
Figure 34. Device "Pass Through" Example .....	97
Figure 35. Example of Mutli-Function Controllers .....	99
Figure 36. iSCSI Product and Package Model.....	114
Figure 37. iSCSI Sessions and Connections Model .....	115
Figure 38. iSCSI Initiator Node .....	115
Figure 39. Multipath Management Class Diagram.....	131
Figure 40. Four Path Instance Diagram.....	132
Figure 41. Four SB Channel Instance Diagram .....	149
Figure 42. Two SB Channel, Three Volume Instance Diagram .....	150



## 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 subprofiles 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, Overview, 1.4.0 Rev 6*
- *Storage Management Technical Specification, Part 1 Common Architecture, 1.4.0 Rev 6*
- *Storage Management Technical Specification, Part 2 Common Profiles, 1.4.0 Rev 6*
- *Storage Management Technical Specification, Part 3 Block Devices, 1.4.0 Rev 6*
- *Storage Management Technical Specification, Part 4 File Systems, 1.4.0 Rev 6*
- *Storage Management Technical Specification, Part 5 Fabric, 1.4.0 Rev 6*
- *Storage Management Technical Specification, Part 6 Host Elements, 1.4.0 Rev 6*
- *Storage Management Technical Specification, Part 7 Media Libraries, 1.4.0 Rev 6*

### Acknowledgments

The SNIA SMI Technical Steering Group, which developed and reviewed this standard, would like to recognize the significant contributions made by the following members:

<i>Organization Represented</i>	<i>Name of Representative</i>
Brocade .....	John Crandall
EMC .....	Mike Hadavi
.....	Mike Thompson
Hewlett Packard.....	Steve Peters
Hitachi Data Systems.....	Eric Hibbard
.....	Steve Quinn
IBM .....	Krishna Harathi
.....	Mike Walker
Olocity.....	Scott Baker
Pillar.....	Gary Steffens
Symantec.....	Steve Hand
.....	Paul von Behren

### 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, 425 Market Street, Suite 1020, San Francisco, CA 94105, U.S.A.



## Clause 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 (or subprofiles) 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

- 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.

- Fibre Channel HBA Profile

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

- iSCSI Initiator Subprofile

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 (subprofiles) 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 Subprofile

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

- SB Multipath Management Subprofile

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

- SCSI Multipath Management Subprofile

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



## Clause 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

DMTF DSP0200, CIM Operations Over HTTP 1.2.0

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

### 2.2 References under development

*Storage Management Technical Specification, Part 1 Common Architecture, 1.4.0 Rev 6*

*Storage Management Technical Specification, Part 2 Common Profiles, 1.4.0 Rev 6*

### 2.3 Other references

ISO/IEC Directives, Part 2, Rules for the structure and drafting of International Standards,  
<http://isotc.iso.org/livelink/livelink.exe?func=ll&objId=4230456&objAction=browse&sort=subtype>

Unified Modeling Language™ (UML®) Specification,  
[http://www.omg.org/technology/documents/modeling\\_spec\\_catalog.htm#UML](http://www.omg.org/technology/documents/modeling_spec_catalog.htm#UML)





## Clause 3: Terms and definitions

### 3.1 General

For the purposes of this document, the terms and definitions given in *Storage Management Technical Specification, Part 1 Common Architecture, 1.4.0 Rev 6* and the following apply.

### 3.2 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™<sup>1</sup> 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 Disk

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. 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 specification and does not constitute an endorsement of this product by SNIA or any standards organization.



---

---

## EXPERIMENTAL

### Clause 4: Disk Partition Subprofile

#### 4.1 Description

This subprofile 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 subprofile are MBR (used on all operating systems running on X86 hardware), vtoc (Solaris and other operating systems with a BSD heritage), and EFI (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. EFI is a new set of interfaces for x86 64-bit environments and includes a partitioning style. Of particular note is that EFI partitions can exceed the two-terabyte limit associated with other partition styles. So many vendors are migrating towards EFI as an option for supporting larger volumes. This profile includes separate specialized subclasses for MBR, VTOC, and EFI partitions. Their relationship is summarized in Figure 5.

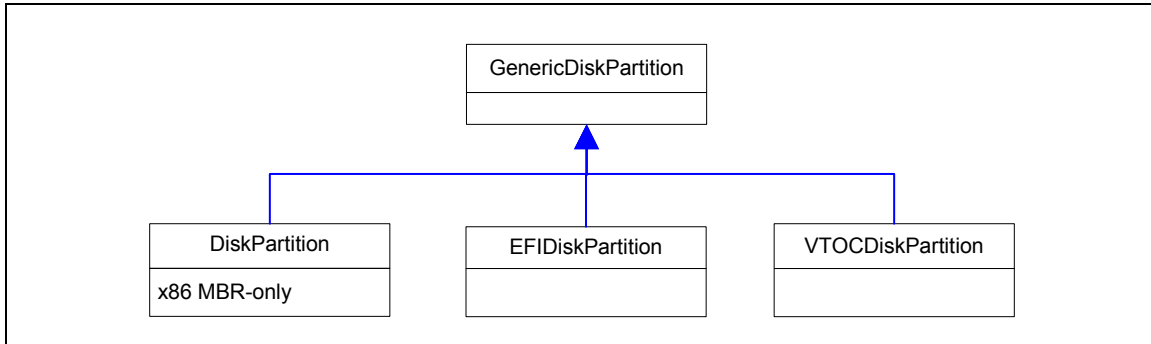


Figure 5 - Disk Partition Class Hierarchy

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 should be at most one instance of DiskPartitionConfigurationService, as shown in Figure 6.

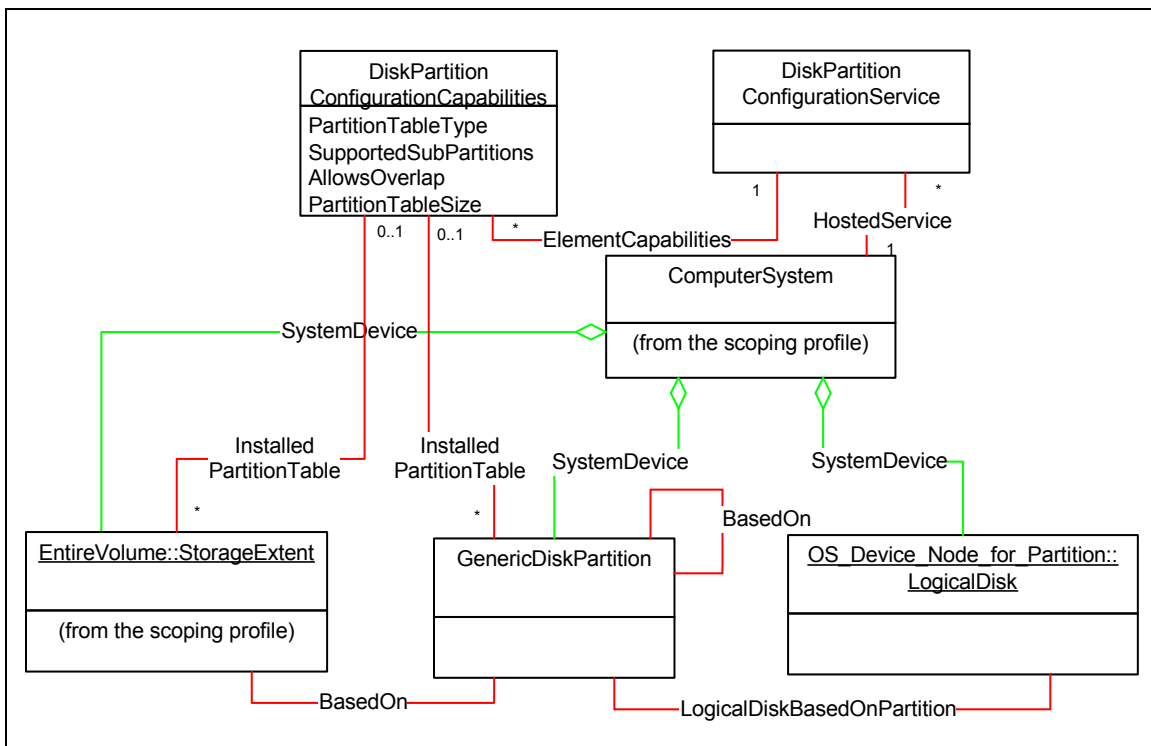


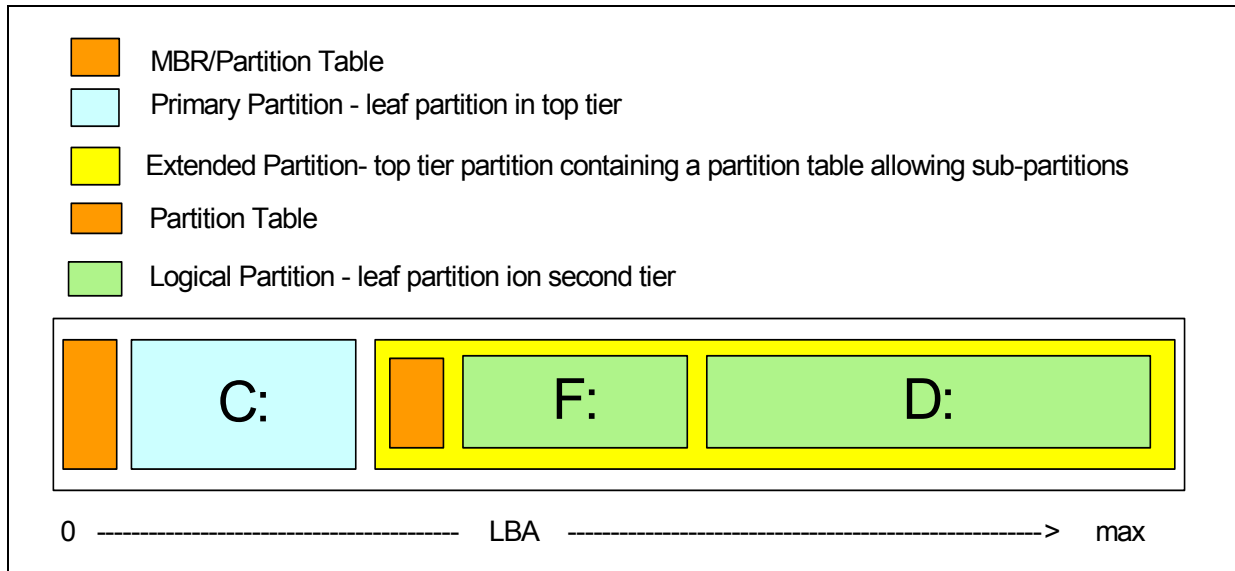
Figure 6 - Disk Partition Class Diagram

#### 4.1.1 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.

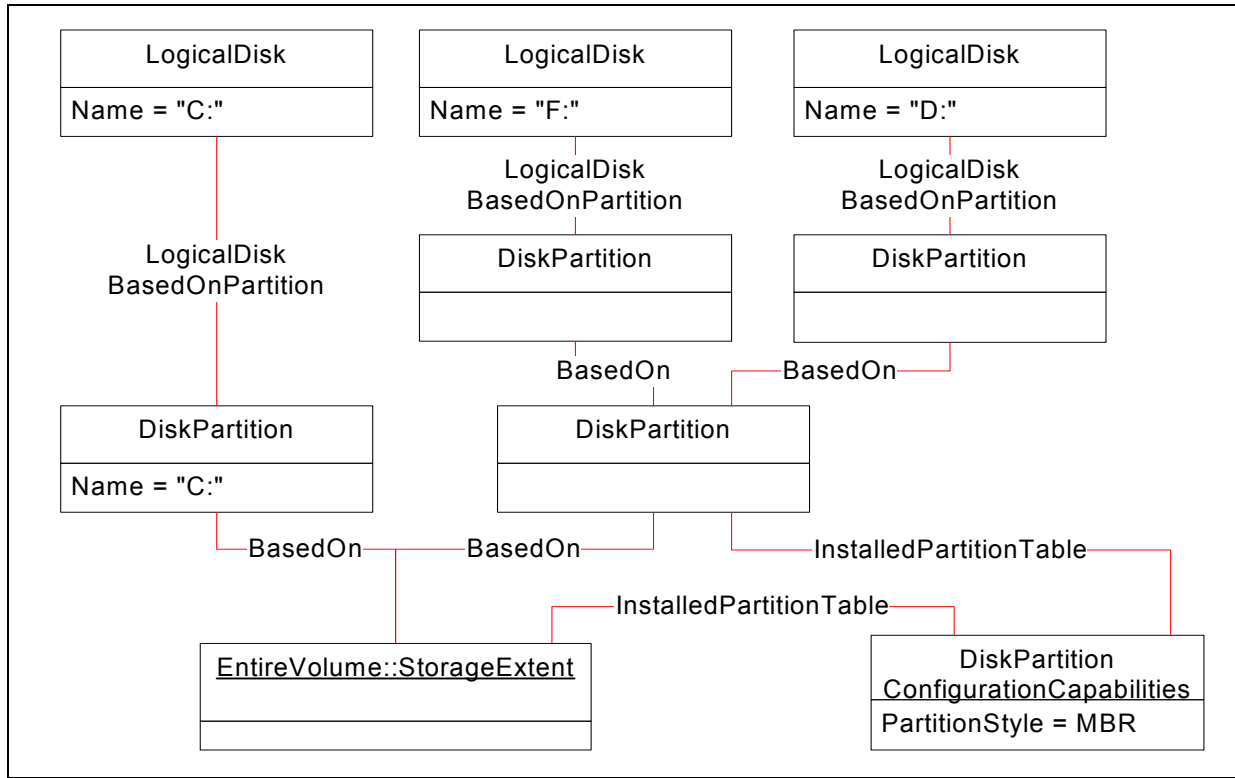


**Figure 7 - Disk MBR Partition Example**

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.

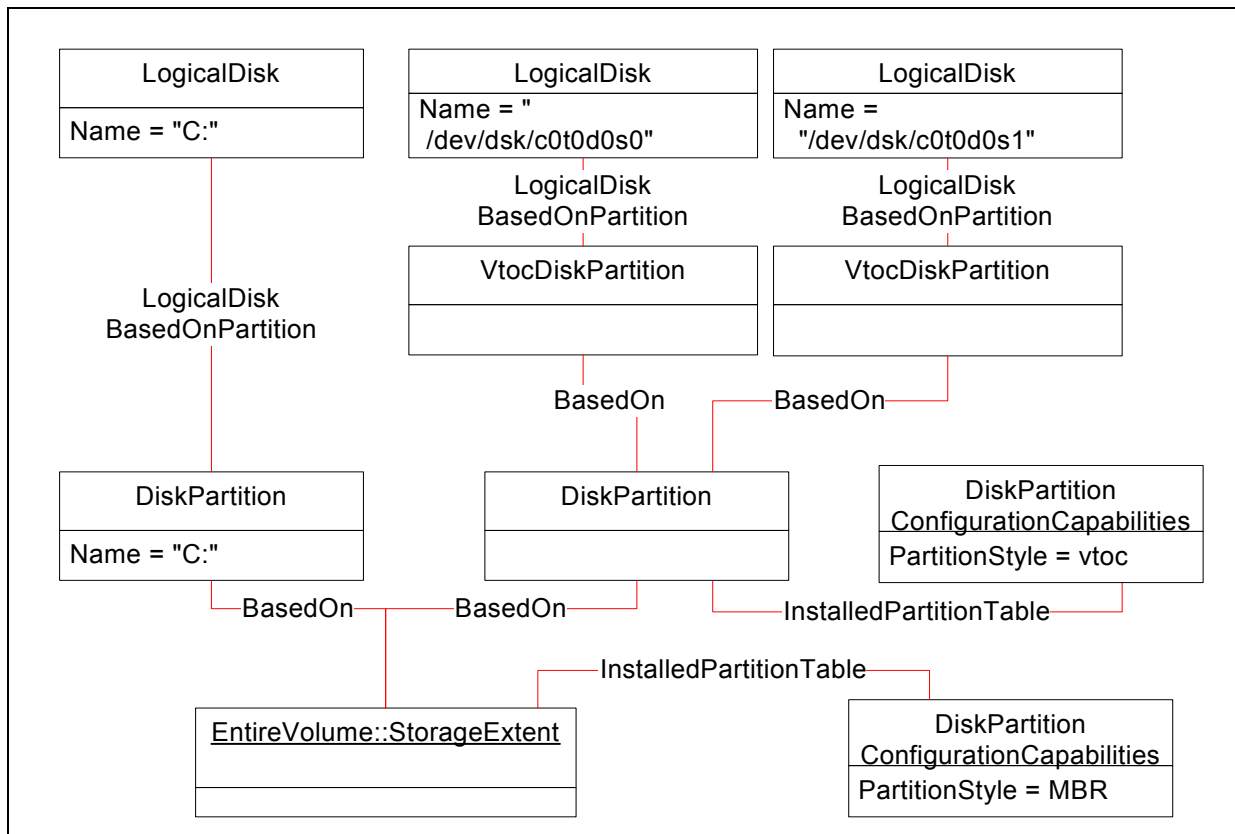
Figure 8 is an instance diagram of the SMI-S classes describing this configuration. Technically, the MBR/Partition tables could be considered to be small partitions. operating systems generally hide these sectors and treat the effective disk volume as starting just after the MBR. Rather than complicate 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.



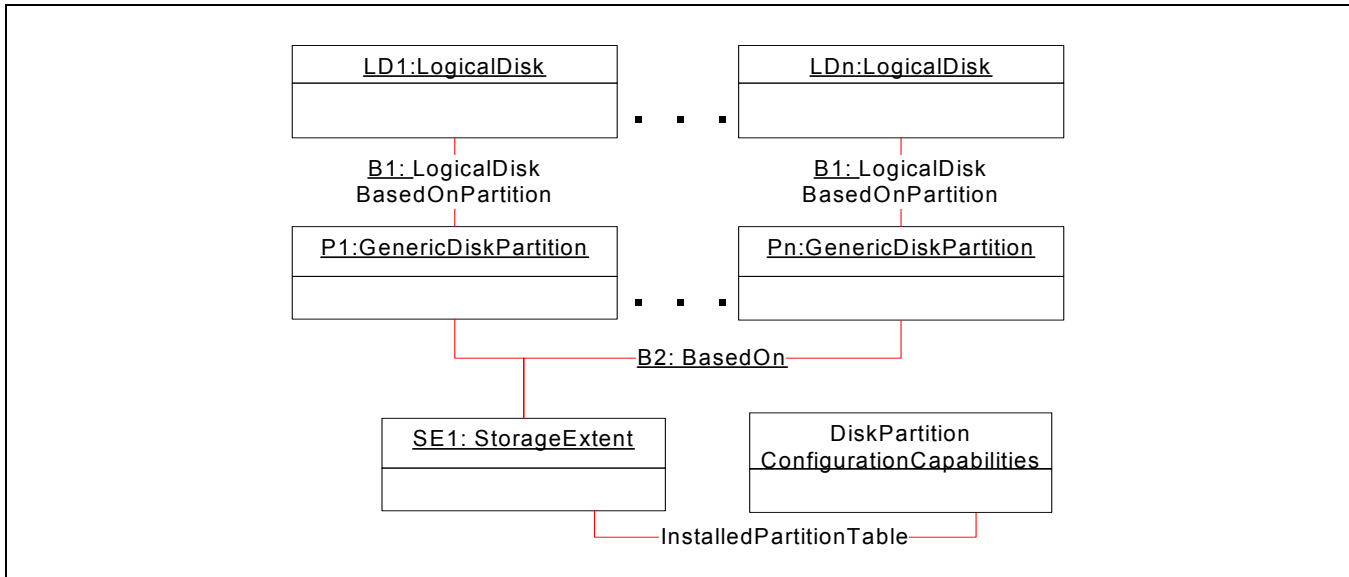
**Figure 9 - MBR and VTOC Partition Instance Diagram**

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

**Table 1 - Capabilities Properties**

Property	X86 MBR	vtoc	EFI			
			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 will 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 LogicalDiskBasedOnPartition 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 EFI, 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 Subprofiles and Packages

None

### 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 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);

```

Delete Instance to delete DiskPartition (and everything south)

## 4.5 Client Considerations and Recipes

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 table type. If multiple capabilities instances are discovered, the client should look at the SupportedExtentTypes property to determine the services that apply to entire disk volumes and those that apply to partitions.

### 4.5.1 Create New Partition Using All Available Space at End of Volume

```

//
// Description:
// Create New Partition Using All Available Space at End of Volume
//
// Preconditions:
// $Host holds a ref to the (top-level) ComputerSystem
// $Disk holds a reference to the LogicalDisk (or StorageExtent) instance
// representing the disk or disk volume. $Disk must either be "raw"
// (no volume label), or have some partitioned space at the end.

// Locate instances of CIM_DiskPartitionConfigurationService.
// Note that HDR does not support the multiple system SP,
// so all services must be hosted on $Host.

$Services = AssociatorNames($Host,
    "CIM_HostedService",
    "CIM_DiskPartitionConfigurationService",
    "Antecedent", // Role
    "Dependent") // Result Role
// If no service instances are found, then this platform does
// not support partitioning - so exit.
if ($Service->[].size == 0) {
    <EXIT This system does not support SMI-S disk partitioning>
}

```

```

// Look for CIM_DiskPartitionConfigurationCapabilities
// associated to $Disk.
$Capabilities->[] = AssociatorNames($Host->, // ObjectName
    "CIM_ElementCapabilities", // AssocClass
    "CIM_DiskPartitionConfigurationCapabilities", // ResultClass
    "ManagedElement", // Role
    "Capabilities") // ResultRole

if ($Capabilities != null && $Capabilities->[].size > 1) {
    <ERROR - must not be more than 1
        CIM_DiskPartitionConfigurationCapabilities
        associated with an extent>

#CreateOneBigPartition = false
if ($Capabilities == null || $Capabilities->[].size == 0 ) {
    // No Capabilities instance found associated to $Disk, this disk has no
    // volume label, create a label with SetPartitionStyle() using
    // the first service instance found.

    // Locate the first Capabilities instance associated with the
    // service. If none, then error.
    $Capabilities->[] =
    // If no capabilities associated to service, then error exit

    %InArguments["Extent"] = $Disk
    %InArguments["Capabilities"] = $Capabilities->[0]
    #MethodReturn = $Services[0]->InvokeMethod(
        $Service->[0],
        "SetPartitionStyle",
        %InArguments)
    if (#MethodReturn != 0) {
        <ERROR - SetPartitionStyle non-zero method return>
    }
    #CreateOneBigPartition = true;
}

// locate partitons based on this disk
$BasedOns[] = References(
    $Disk->[],
    "CIM_BasedOn", // Assoc class
    "Antecedent", // my role
    false,
    false,
    {"StartingAddress", "EndingAddress"})
if ($BasedOns[] == null || $BasedOns->[].size == 0) {
    // If $Disk has no associated partitions, create one using

```

```

    // entire disk with CreateOrModifyPartition()
    #CreateOneBigPartition = true;
}

if (#CreateOneBigPartition == true) {
    // null starting and ending address parameters mean
    // "use entire disk".
    // null Partition REF parameter means Create
    %InArguments["Extent"] = $Disk
    // all other parms default to "use entire extent"
    #MethodReturn = $Services[0]->InvokeMethod(
        $Services->[0],
        "CreateOrModifyPartition",
        %InArguments)
    if(#MethodReturn != 0) {
        <ERROR! CreateOrModifyPartition full disk method Failed >
    }
}

// Look for available space at end of disk
// Note that the order of partitions in $BasedOns is not necessarily
// the same as the order of the addresses in the partitions.
#CreatePartPossible = true;
// LastBlockInParts in the highest block address in any partition
#LastBlockInParts = $Capabilities.PartitionTableSize
$Capabilities = <get capabilities instance associated with this disk>
for (#i in $BasedOns->[]) {
    // if this partition goes to the end of the underlying extent ...
    if ($BasedOns[#i].EndingAddress == $Disk.NumberOfBlocks-1) {
        // if OverlapAllowed and this partitions takes up entire
        // consumable disk space, then this is a special backup
        // partition - the condition below is the opposite...
        if (!(($Capabilities.OverlapAllowed) ||
            ($BasedOns->[#i].StartingAddress >
                $Capabilities.PartitionTableSize))) {
            #CreatePartPossible = false
        }
    } else {
        // This partition ends after others we've seen (LastBlockInParts)
        // Update LastBlockInParts with the new address
        if ($BasedOns[#i].EndingAddress > #LastBlockInParts) {
            #LastBlockInParts = $BasedOns[#i].EndingAddress
        }
    }
}
if (#CreatePartPossible) {

```

```

if ($BasedOns->[].size() >= $Capabilities.MaxNumberOfPartitions) {
    // then we can't create any more partitions - exit
    exit
} else {
    // Get the service associated with $Capabilities
    $Service = ..
    $Services = AssociatorNames($Host,
        "CIM_InstalledPartitionTable",
        "CIM_DiskPartitionConfigurationService",
        "Antecedent", // Role
        "Dependent") // Result Role
    %InArguments["Extent"] = $Disk;
    %InArguments["StartingBlock"] = #LastBlockInParts + 1
    // EndingBlock will default to end of disk
    #MethodReturn = InvokeMethod(
        $Services->[0],
        "CreateOrModifyPartition",
        %InArguments)

    if(#MethodReturn != 0) {
        <ERROR! CreateOrModifyPartition park disk method Failed >
    }
}
} else {
    <EXIT - no space at end of disk>
}

```

## 4.6 Registered Name and Version

Disk Partition version 1.3.0

## 4.7 CIM Elements

Table 2 describes the CIM elements for Disk Partition.

**Table 2 - CIM Elements for Disk Partition**

Element Name	Requirement	Description
4.7.1 CIM_BasedOn (Partition to Extent)	Mandatory	
4.7.2 CIM_BasedOn (Partition to Partition)	Mandatory	
4.7.3 CIM_DiskPartitionConfigurationCapabilities	Mandatory	
4.7.4 CIM_DiskPartitionConfigurationService	Mandatory	
4.7.5 CIM_ElementCapabilities	Mandatory	

**Table 2 - CIM Elements for Disk Partition**

Element Name	Requirement	Description
4.7.6 CIM_GenericDiskPartition	Mandatory	
4.7.7 CIM_HostedService	Mandatory	
4.7.8 CIM_InstalledPartitionTable (Capabilities to Extent)	Mandatory	
4.7.9 CIM_InstalledPartitionTable (Capabilities to Partition)	Mandatory	
4.7.10 CIM_LogicalDisk	Mandatory	
4.7.11 CIM_LogicalDiskBasedOnPartition	Mandatory	
4.7.12 CIM_StorageExtent	Mandatory	
4.7.13 CIM_SystemDevice (System to Extent)	Mandatory	
4.7.14 CIM_SystemDevice (System to LogicalDisk)	Mandatory	
4.7.15 CIM_SystemDevice (System to Partition)	Mandatory	
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_GenericDiskPartition	Mandatory	Partition Creation.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_GenericDiskPartition	Mandatory	Partition Deletion.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_GenericDiskPartition	Mandatory	Partition Modification.

**4.7.1 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	

**Table 3 - SMI Referenced Properties/Methods for CIM\_BasedOn (Partition to Extent)**

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

**4.7.2 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	

**4.7.3 CIM\_DiskPartitionConfigurationCapabilities**

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 5 describes class CIM\_DiskPartitionConfigurationCapabilities.

**Table 5 - SMI Referenced Properties/Methods for CIM\_DiskPartitionConfigurationCapabilities**

Properties	Flags	Requirement	Description & Notes
PartitionStyle		Mandatory	
ValidSubPartitionStyles		Mandatory	
MaxNumberOfPartitions		Mandatory	
MaxCapacity		Mandatory	

**Table 5 - SMI Referenced Properties/Methods for CIM\_DiskPartitionConfigurationCapabilities**

Properties	Flags	Requirement	Description & Notes
OverlapAllowed		Mandatory	
PartitionTableSize		Mandatory	

**4.7.4 CIM\_DiskPartitionConfigurationService**

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 6 describes class CIM\_DiskPartitionConfigurationService.

**Table 6 - SMI Referenced Properties/Methods for CIM\_DiskPartitionConfigurationService**

Properties	Flags	Requirement	Description & Notes
SetPartitionStyle()		Mandatory	
CreateOrModifyPartition()		Mandatory	

**4.7.5 CIM\_ElementCapabilities**

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 7 describes class CIM\_ElementCapabilities.

**Table 7 - SMI Referenced Properties/Methods for CIM\_ElementCapabilities**

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	
Capabilities		Mandatory	

**4.7.6 CIM\_GenericDiskPartition**

Created By: Extrinsic: CreateOrModifyPartition

Modified By: Static

Deleted By: Static

Requirement: Mandatory



Table 8 describes class CIM\_GenericDiskPartition.

**Table 8 - SMI Referenced Properties/Methods for CIM\_GenericDiskPartition**

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

**4.7.7 CIM\_HostedService**

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

Table 9 describes class CIM\_HostedService.

**Table 9 - SMI Referenced Properties/Methods for CIM\_HostedService**

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

**4.7.8 CIM\_InstalledPartitionTable (Capabilities to Extent)**

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

Table 10 describes class CIM\_InstalledPartitionTable (Capabilities to Extent).

**Table 10 - SMI Referenced Properties/Methods for CIM\_InstalledPartitionTable (Capabilities to Extent)**

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

#### 4.7.9 CIM\_InstalledPartitionTable (Capabilities to Partition)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 11 describes class CIM\_InstalledPartitionTable (Capabilities to Partition).

**Table 11 - SMI Referenced Properties/Methods for CIM\_InstalledPartitionTable (Capabilities to Partition)**

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

#### 4.7.10 CIM\_LogicalDisk

Created By: Extrinsic: CreateOrModifyPartition

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 12 describes class CIM\_LogicalDisk.

**Table 12 - SMI Referenced Properties/Methods for CIM\_LogicalDisk**

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

**Table 12 - SMI Referenced Properties/Methods for CIM\_LogicalDisk**

Properties	Flags	Requirement	Description & Notes
NameFormat		Mandatory	OS Device Name.
NameNamespace		Mandatory	OS Device Namespace.
OperationalStatus		Mandatory	
NumberOfBlocks		Optional	

**4.7.11 CIM\_LogicalDiskBasedOnPartition**

Created By: Extrinsic: CreateOrModifyPartition

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 13 describes class CIM\_LogicalDiskBasedOnPartition.

**Table 13 - SMI Referenced Properties/Methods for CIM\_LogicalDiskBasedOnPartition**

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

**4.7.12 CIM\_StorageExtent**

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 14 describes class CIM\_StorageExtent.

**Table 14 - SMI Referenced Properties/Methods for CIM\_StorageExtent**

Properties	Flags	Requirement	Description & Notes
SystemCreationClass sName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	
Name		Mandatory	

**Table 14 - SMI Referenced Properties/Methods for CIM\_StorageExtent**

Properties	Flags	Requirement	Description & Notes
OperationalStatus		Mandatory	
NumberOfBlocks		Optional	

**4.7.13 CIM\_SystemDevice (System to Extent)**

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 15 describes class CIM\_SystemDevice (System to Extent).

**Table 15 - SMI Referenced Properties/Methods for CIM\_SystemDevice (System to Extent)**

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	
PartComponent		Mandatory	

**4.7.14 CIM\_SystemDevice (System to LogicalDisk)**

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 16 describes class CIM\_SystemDevice (System to LogicalDisk).

**Table 16 - SMI Referenced Properties/Methods for CIM\_SystemDevice (System to LogicalDisk)**

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	
PartComponent		Mandatory	

**4.7.15 CIM\_SystemDevice (System to Partition)**

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 17 describes class CIM\_SystemDevice (System to Partition).

**Table 17 - SMI Referenced Properties/Methods for CIM\_SystemDevice (System to Partition)**

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	
PartComponent		Mandatory	

## **EXPERIMENTAL**

---

---



---



---

## IMPLEMENTED

### Clause 5: FC HBA Profile

#### 5.1 Synopsis

**Profile Name:** FC HBA

**Version:** 1.4.0

**Organization:** SNIA

**CIM Schema Version:** 2.9.0

Table 18 describes the related profiles for FC HBA.

**Table 18 - Related Profiles for FC HBA**

Profile Name	Organization	Version	Requirement	Description
FC Initiator Ports	SNIA	1.4.0	Mandatory	
Software Update	DMTF	1.0.0	Optional	
Indication	SNIA	1.4.0	Optional	

Central Class Instance: CIM\_ComputerSystem

Scoping Class Instance: CIM\_ComputerSystem

The FC HBA profile describes behavior of Fibre Channel host adapters supporting the SCSI (FCP) command set.

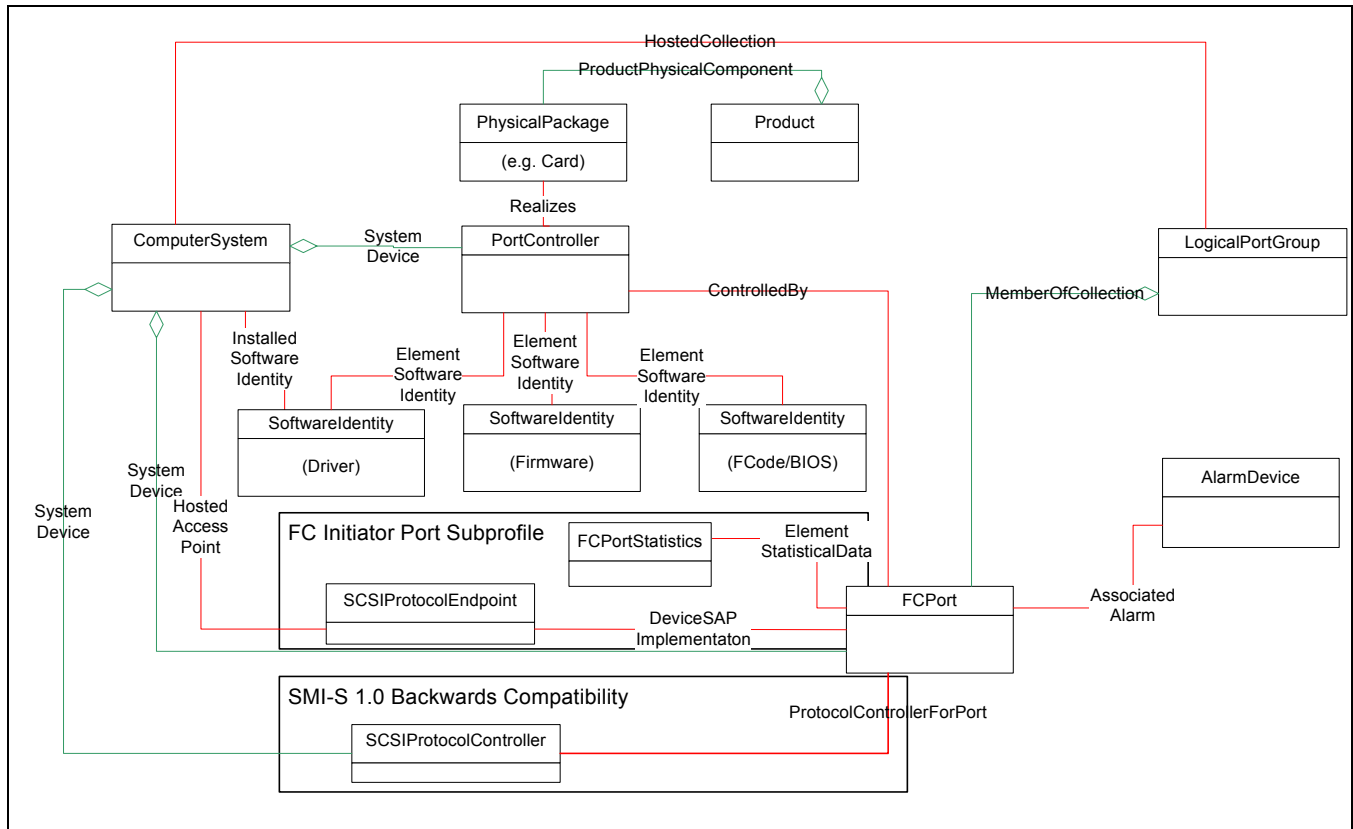
#### 5.2 Description

The FC HBA profile describes behavior of Fibre Channel host adapters supporting the SCSI (FCP) command set. A Fibre Channel adapter used in a host system is called a Host Bus Adapter (HBA). An HBA is a physical device that contains one or more Fibre Channel ports. A single system contains one or more HBAs.

#### 5.3 Implementation

An HBA is represented in CIM by FCPorts associated to a ComputerSystem through the SystemDevice association. To understand the containment to the HBAs physical implementation, the FCPorts are associated to PhysicalPackage through the Realizes association. The PortController represents the logical behavior of the HBA card. It is associated to the ComputerSystem through the SystemDevice association and associated to the FCPorts through the ControlledBy association. PortController's PhysicalPackage is associated with Product - which holds information about the HBA (including vendor and model names).

If the FCPorts reside on a mainboard (rather than a separate card), the same model is used - PortController and PhysicalPackage represent the mainboard. Attributes of Product refer the vendor and model names of the FCPorts, not the mainboard or system.



**Figure 11 - FC HBA Instance Diagram**

Separate instances of SoftwareIdentity represent driver, firmware, and FCODE/BIOS associated with the HBA and includes properties for the vendor, product, and version names (see 5.7 CIM Elements for details). The Classifications property identifies the type (driver, firmware,...). The SoftwareIdentity instance for the driver is mandatory; the others are optional. Note that a separate instance of SoftwareIdentity representing the SMI-S/CIM instrumentation is required by the Profile Registration Profile.

**DEPRECATED**

**5.3.1 Modeling SCSI Protocol Support**

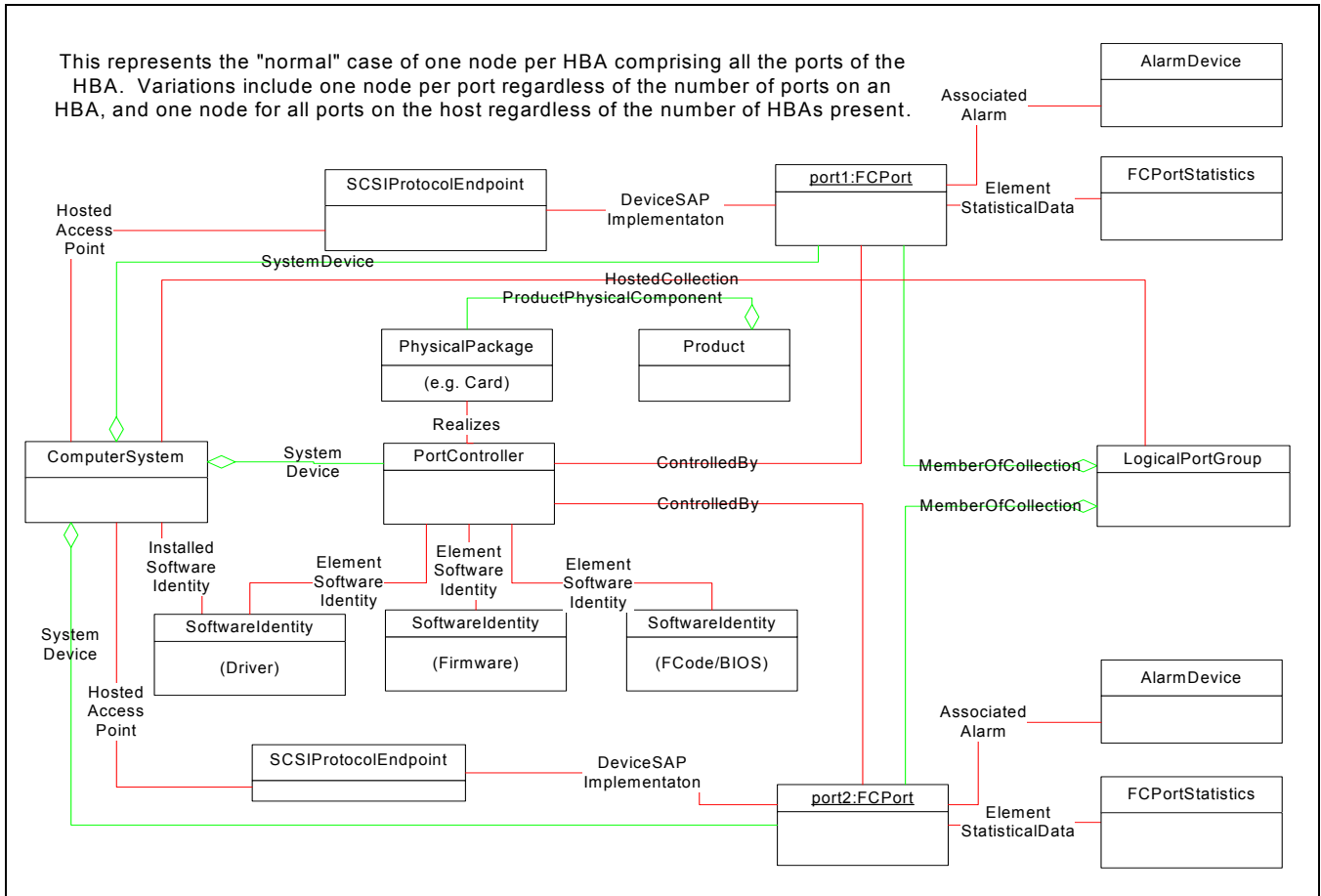
The SMI-S 1.0 model for ports and protocols addressed FCP (SCSI over Fibre Channel). As other configurations were considered, the general pattern of initiator port subprofiles (see *Storage Management Technical Specification, Part 2 Common Profiles, 1.4.0 Rev 6,*) emerged. For SMI-S 1.1.0, any initiator port that is configured for SCSI protocol shall use the model in the instance diagram above (ComputerSystem-Hosted Access Point-SCSIProtocolEndpoint-DeviceSAPImplementation-FCPort).

For backwards compatibility, the FC HBA Profile also exposes the SMI-S 1.0 classes (SCSIProtocolController and ProtocolControllerForPort association). In the future, SCSIProtocolController and ProtocolControllerForPort will not be part of the profile; client applications are encouraged to migrate to the new model.

**DEPRECATED**



Figure 12 depicts the model for an HBA card with two ports. The LogicalPortGroup represents the collection of ports that shared a Node WWN.



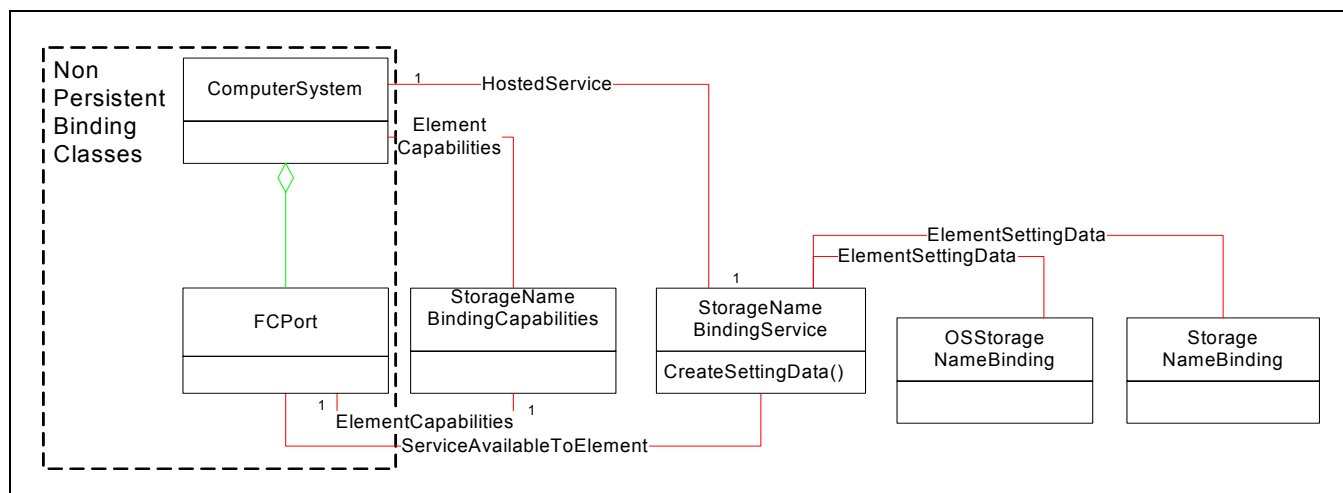
**Figure 12 - HBA Card with Two Ports**

**5.3.2 Persistent Binding**

Persistent Binding describes the capability of host adapters to persist user preferences regarding which target logical units are mapped to which OS device names. Persistent Binding for Fibre Channel HBAs is documented in detail in the FC API specification (see <ftp://ftp.t11.org/t11/pub/fc/hba/04-137v0.pdf>).

The term "Persistent Binding" technically refers to the data structure that maps the association from target device correlatable IDs to an OS device name. The collection of these bindings is persisted by the HBA and/or drivers. A persistent binding structure can be defined while the referenced hardware is offline or uninstalled. When the drivers discover attached hardware that matches a persistent binding, the mapping takes place. In many cases, a newly defined persistent binding has no impact until the system is rebooted. The impact will cause target logical units to be attached to initiator SCSIProtocolEndpoints. These associations and target objects are modeled with the Host Discovered Resources Profile.

The persistent binding data structure for bindings that specify OS device names is modeled as OSStorageNameBinding. StorageNameBinding is used when a persistent binding of a device name is determined by the OS. StorageNameBindingService includes methods to create instances of the setting data subclasses, and StorageNameBindingCapabilities provides information about the capabilities of the implementation.



**Figure 13 - Persistent Binding Model**

Persistent Binding is optional. An implementation that does not support persistent binding (and any of the classes in the diagram above) shall not instantiate an instance of StorageNameBindingService. An implementation that does support persistent binding shall:

- Instantiate a single instance of StorageNameBindingService and associate it to the ComputerSystem.
- Instantiate an instance of StorageNameBindingCapabilities for each FCPort instance, associated via ElementCapabilities.
- At initialization, the implementation shall instantiate instances of OSStorageNameBinding or StorageNameBinding for each previously defined binding.
- Implement the CreateOSStorageNameBinding Method if any StorageNameBindingCapabilities exists with CanSetOSDeviceName set to true.
- Implement the CreateStorageNameBinding Method if StorageNameBindingCapabilities instance exists with CanSetOSDeviceName set to false.
- Support DeleteInstance for StorageNameBinding and OSStorageNameBinding.
- Support ModifyInstance of StorageNameBindingCapabilities.

### 5.3.3 LED Blink

Implementations may optionally support LED blinking by instantiating a AlarmDevice instance and associating it via AssociatedAlarm to Port instances.

AlarmDevice.VisibleAlarm sound be set to true.

AlarmDevice.Urgency should be set to 3 (Informational).

The instrumentation shall provide the SetAlarmState method on AlarmDevice. This method has a single parameter RequestedAlarmState. The only value for this parameter shall be 3 (Alternating).

## 5.4 Health and Fault Management

Not defined in this standard.

## 5.5 Methods

### 5.5.1 Extrinsic Methods of this Profile

The following extrinsic methods are available, but only required if the specific capability (persistent binding or LED blink) is supported.

#### 5.5.2 StorageNameBindingService.CreateStorageNameBinding

This method requests that the driver create a name binding from a target (and optional logical unit) and lets the OS assign the name.

```
uint32 CreateStorageNameBinding (
    [IN, Description ("The value to assign to BindingType."),
    uint16 BindingType,

    [IN, Description ("The value to assign to BindAllLogicalUnits.")]
    boolean BindAllLogicalUnits,

    [IN, Description ("The value to assign to Hide.")]
    boolean Hide,

    [IN, Description ("The value to assign to TargetName.")]
    string TargetName,

    [IN, Description ("The value to assign to LogicalUnitNumber.")]
    string LogicalUnitNumber,

    [IN, Description ("The type of the ports in LocalPortNames."),
    // shall be "2" "FC Port WWN"
    uint16 LocalPortNameType,

    [IN, Description ("The values to assign to LocalPortNames.")]
    string LocalPortName,

    [IN (false), OUT, Description ("A reference to the created name binding
    instance.")]
    StorageNameBinding REF Binding);
```

#### 5.5.3 StorageNameBindingService.CreateOSStorageNameBinding

This method requests that the driver create a name binding from a target (and option logical unit) to a specified OS Device Name or addresses."

```
uint32 CreateOSStorageNameBinding (
    [IN, Description ("The value to assign to BindingType."),
    uint16 BindingType,

    [IN, Description ("The value to assign to BindAllLogicalUnits.")]
    boolean BindAllLogicalUnits,

    [IN, Description ("The value to assign to Hide.")]
```

```

boolean Hide,

    [IN, Description ("The value to assign to TargetName.")]
string TargetName,

    [IN, Description ("The value to assign to LogicalUnitNumber.")]
string LogicalUnitNumber,

    [IN, Description ("The value to assign to OSDeviceName.")]
string OSDeviceName,

    [IN, Description ("The value to assign to OSAddressesValid.")]
boolean OSAddressesValid,

    [IN, Description ("The value to assign to OSBusNumber.")]
uint32 OSBusNumber,

    [IN, Description ("The value to assign to OSTargetNumber.")]
uint32 OSTargetNumber,

    [IN, Description ("The value to assign to OSLUN.")]
uint32 OSLUN,

    [IN, Description ("The type of the ports in LocalPortNames."),
    // shall be "2" "FC Port WWN"
uint16 LocalPortNameType,

    [IN, Description ("The values to assign to LocalPortNames.")]
string LocalPortName,

    [IN (false), OUT, Description ("A reference to the created name binding
    instance.")]
CIM_StorageNameBinding REF Binding);

```

### 5.5.3.1 CIM\_AlarmDevice.SetAlarmState

The instrumentation shall provide the SetAlarmState method on AlarmDevice. This method has a single parameter RequestedAlarmState. The instrumentation shall support a value of 3 (Alternating) for this parameter.

### 5.5.4 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

## 5.6 Client Considerations and Recipes

Different HBA vendors may have separate implementations of this profile installed on the same server; the instrumentation may be running under the same or different CIM servers.

### 5.6.1 Discovery HBA Topology and Attributes

```
// DESCRIPTION
//
// This recipe discovers the topology of an FC HBA. Noteworthy information
// such as installed firmware/software and port information is retrieved.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
//
// 1. A reference to the top-level ComputerSystem in the FC HBA Profile,
//    which represents the system hosting the HBA, is known as $Host->
//
// Step 1. Get name(s) of the HBA's on the host system. Note that there
// MAY be more than one HBA on the host.
//
$HBA->[] = AssociatorNames($Host->, // ObjectName
    "CIM_SystemDevice", // AssocClass
    "CIM_PortController", // ResultClass
    "GroupComponent", // Role
    "PartComponent") // ResultRole

if ($HBA->[] == null || $HBA->[].length == 0) {
    <EXIT: No HBAs on the host system!>
}

// Determine the topology and retrieve noteworthy information for each HBA.
//
for (#i in $HBA->[]) {
    // Step 2. Determine the vendor and product information of the HBA.
    //
    $PhysicalPackage[] = Associators(
        $HBA->[#i], // ObjectName
        "CIM_Realizes", // AssocClass
        "CIM_PhysicalPackage", // ResultClass
        "Antecedent", // ResultRole
        "Dependent", // Role
        false, // IncludeQualifiers
        false, // IncludeClassOrigin
        {"Manufacturer", "Model"}) // PropertyList
    // Exactly one PhysicalPackage shall be returned
}
```

```

if ($PhysicalPackage[] == null || $PhysicalPackage[].length == 0) {
    <ERROR! Improper Physical Package information!>
}
// NOTE: The Product properties of interest are all Key qualified
// properties, thus the instance name rather than the instance
// itself is retrieved.
//
$Product->[] = AssociatorNames(
    $PhysicalPackage[0], // ObjectName
    "CIM_ProductPhysicalComponent", // AssocClass
    "CIM_Product", // ResultClass
    "GroupComponent", // ResultRole
    "PartComponent") // Role
// Exactly one Product shall be returned
if ($Product->[] == null || $Product->[].length == 0) {
    <ERROR! Improper Product information!>
}
// Step 3. Determine the software (e.g. firmware, driver(s), BIOS,
// FCode) installed on the HBA.
//
#PropList = {"VersionString", "Manufacturer", "Classifications"}
$Software[] = Associators(
    $HBA->[#i], // ObjectName
    "CIM_ElementSoftwareIdentity", // AssocClass
    "CIM_SoftwareIdentity", // ResultClass
    "Antecedent", // ResultRole
    "Dependent", // Role
    false, // IncludeQualifiers
    false, // IncludeClassOrigin
    #PropList) // PropertyList
if ($Software[] != null && $Software[].length > 0) {
    for (#j in $Software[]) {
        // Retrieve relevant property instance data
        // These properties are not used in the recipe,
        // this just demonstrates how to locate this
        // information
        #VersionString = $Software[#j].VersionString
        #Manufacturer = $Software[#j].Manufacturer
        #Classifications[] = $Software[#j].Classifications
    }
}
// Step 4. Locate the Fibre Channel ports on the HBA and determine
// each port's speed and WWN.
#PropList = {"Speed", "PermanentAddress"}
$Ports[] = Associators(
    $HBA->[#i], // ObjectName
    "CIM_ControlledBy", // AssocClass

```

```

    "CIM_FCPort", // ResultClass
    "Dependent", // ResultRole
    "Antecedent", // Role
    false,      // IncludeQualifiers
    false,      // IncludeClassOrigin
    #PropList) // PropertyList
if ($Ports[] != null && $Ports[].length > 0) {
    for (#j in $Ports[]) {
        // Retrieve relevant Port instance data
        #Speed = $Ports[#j].Speed
        #PermanentAddress[] = $Ports[#j].PermanentAddress
        // Step 5. Determine the Node WWN of the port.
        $PortGroup[] = Associators(
            $Ports[#j].getObjectPath(), // ObjectName
            "CIM_MemberOfCollection", // AssocClass
            "CIM_LogicalPortGroup", // ResultClass
            "Collection", // ResultRole
            "Member", // Role
            false,      // IncludeQualifiers
            false,      // IncludeClassOrigin
            {"Name"}) // PropertyList
        // Exactly one PhysicalPackage MUST be returned
        if ($PortGroup[] == null || $PortGroup[].length == 0) {
            <ERROR! Improper Port Group information!>
        }
        #NodeWWN = $PortGroup[0].Name
    }
}
}

```

### 5.6.2 Get the statistics for each port

```

//
// DESCRIPTION
//
// Find the FCPortStatistics associated with FC ports
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
//
// 1. A reference to the top-level ComputerSystem in the FC HBA Profile,
//    which represents the system hosting the HBA, is known as $Host->
//
// Get a list of all the ports
$Ports->[] = AssociatorNames($Host->, // ObjectName
    "CIM_SystemDevice", // AssocClass
    "CIM_FCPort", // ResultClass
    "GroupComponent", // Role
    "PartComponent") // ResultRole

```

```

if ($Ports->[] == null || $Ports->[].length == 0) {
    <ERROR! No FC Ports on the host system!>
}

for (#i in $Ports->[] ) {
    // Get a list of FCPortStatistics associated with each port
    // Should only be exactly one FCPortStatistics instance
    $Stats->[] = AssociatorNames($Ports->[#i],// ObjectName
        "CIM_ElementStatisticalData",// AssocClass
        "CIM_FCPortStatistics",// ResultClass
        "ManagedElement",// Role
        "Stats") // ResultRole
    if ($Stats->[] == null || $Ports->[].length == 0) {
        <ERROR! Each FCPort shall have an associated FCPortStatistics>
    } else {
        if ( $Stats->[].length > 1) {
            <ERROR: More than 1 FCPortStatistics associated with a port>
        }
    }
    // $Stats[0]-> holds that stats
}

```

### 5.6.3 Define a persistent binding to a target PWWN

```

// DESCRIPTION
//
// This recipe creates a persistent binding based on a PWWN
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
//
// 1. A reference to the top-level ComputerSystem in the FC HBA Profile,
//    which represents the system hosting the HBA, is known as $Host->
//
// 2. The name of the target port WWN is known as #TargetWWN. The
//    easiest way to discover this is to use an FC Switch or Fabric
//    client application. The host should have a single HBA Profile
//    implementation running and the target MUST be connected to an
//    HBA supported by this profile implementation.
//
// 3. A reference to an FCPort on the local system - $LocalPort->
//
//
// Get a list of initiator ports
// First get all the initiator ports
$Ports->[] = AssociatorNames($Host->,// ObjectName
    "CIM_SystemDevice",// AssocClass
    "CIM_FCPort",// ResultClass
    "GroupComponent", // Role

```



```

    "PartComponent") // ResultRole

if ($Ports->[] == null || $Ports->[].length == 0) {
    <ERROR! No FC Ports on the host system!>
}
if (!contains($LocalPort->, $Ports->[]) {
    <ERROR! The input local port is not on the host system!>
}

$Services->[] = AssociatorNames($Host->, // ObjectName
    "CIM_HostedService", // AssocClass
    "CIM_StorageNameBindingService", // ResultClass
    null, null)

if ($Services == null || $Services[].length == 0) {
    <ERROR: HBA Instrumentation does not instantiate StorageNameBindingService>
}
if ($Services[].length != 1) {
    <ERROR! Must be exactly one StorageNameBindingService>
}

$Capabilities->[] = AssociatorNames($FCPort->, // ObjectName
    "CIM_ElementCapabilities", // AssocClass
    "CIM_StorageNameBindingCapabilities", // ResultClass
    null, null)

If ($Capabilities == null || $Capabilities[].length != 1) {
    <ERROR! must be exactly one StorageNameBindingCapabilities per FCPort>
}

If ($Capabilities->[0].CanBindAllLuns != true) {
    <EXIT: HBA Instrumentation does not support CanBindAllLuns>
}

If contains("FcApiBindToWWN", $Capabilities->[0].ValidBindingTypes) {
    // All checks done, perform the binding
    // set up the arguments and invoke CreateStorageNameBinding
    %InArguments["BindingType"] = "FcApiBindToWWPN"
    %InArguments["BindAllLogicalUnits"]=true
    %InArguments["Hide"]=false
    %InArguments["TargetName"]=#TargetPWWN
    %InArguments["LocalPortNameType"]="2"// FC Port WWN
    %InArguments["LocalPortName"]=$LocalPort->[].PermanentAddress
    #MethodReturn = InvokeMethod(
        $Services->[0],
        "CreateStorageNameBinding",
        %InArguments,

```

```

    %OutArguments)

    if(#MethodReturn != 0) {
        <ERROR! CreateStorageNameBinding method Failed >
    }

    If ($Capabilities->[0].ActivateBindingRequiresReset) {
        <EXIT: Persistent Binding request okay; Reboot Required>
    }

} else {
    <EXIT: HBA instrumentation does not support BindtoWWPN>
}

```

#### 5.6.4 Define a persistent binding to an LUID

```

// DESCRIPTION
//
// This recipe creates a persistent binding based on a LUID
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
//
//
// 1. A reference to the top-level ComputerSystem in the FC HBA Profile,
//     which represents the system hosting the HBA, is known as $Host->
//
// 2. The name of the logical unit (VPD pg 83 ID) is known as #LUID.
//     The easiest way to discover this is to use an array management
//     client application. The host should have a single HBA Profile
//     implementation running and the LU shall be in a target connected
//     to an HBA supported by this profile implementation.
//
// 3. A reference to an FCPort on the local system - $LocalPort->
//
//
// Get a list of initiator ports
// First get all the initiator ports
$Ports->[] = AssociatorNames($Host->, // ObjectName
    "CIM_SystemDevice", // AssocClass
    "CIM_FCPort", // ResultClass
    "GroupComponent", // Role
    "PartComponent") // ResultRole

if ($Ports->[] == null || $Ports->[].length == 0) {
    <ERROR! No FC Ports on the host system!>
}

if (!contains($LocalPort->, $Ports->[])) {
    <ERROR: The input local port is not on the host system!>
}

```

```

}

$Services->[] = AssociatorNames($Host->, // ObjectName
    "CIM_HostedService", // AssocClass
    "CIM_StorageNameBindingService", // ResultClass
    null, null)

if ($Services == null || $Services[].length == 0) {
    <ERROR: HBA Instrumentation does not instantiate StorageNameBindingService>
}
if ($Services[].length != 1) {
    <ERROR! Must be exactly one StorageNameBindingService>
}

$Capabilities->[] = AssociatorNames($FCPort->, // ObjectName
    "CIM_ElementCapabilities", // AssocClass
    "CIM_StorageNameBindingCapabilities", // ResultClass
    null, null)

If ($Capabilities == null || $Capabilities[].length != 1) {
    <ERROR! must be exactly one StorageNameBindingCapabilities per FCPort>
}

If contains("BindToLUID", $Capabilities->[0].ValidBindingTypes) {
    // All checks done, perform the binding
    // set up the arguments and invoke CreateStorageNameBinding
    %InArguments["BindingType"]="BindToLUID"
    %InArguments["BindAllLogicalUnits"]=true
    %InArguments["Hide"]=false
    %InArguments["TargetName">#LUID
    %InArguments["LocalPortNameType"]="2" // FC Port WWN
    %InArguments["LocalPortName"]=$LocalPort->[].PermanentAddress
    #MethodReturn = InvokeMethod(
        $Services->[0],
        "CreateStorageNameBinding",
        %InArguments,
        %OutArguments)
    if(#MethodReturn != 0) {
        <ERROR! CreateStorageNameBinding method Failed>
    }

    If ($Capabilities->[0].ActivateBindingRequiresReset) {
        <EXIT: Persistent Binding request okay; Reboot Required>
    }
} else {
    <EXIT: HBA instrumentation does not support BindtoLUID>
}

```

```
}

```

### 5.6.5 Blink the LED

```
//
// DESCRIPTION
//
// Blink LEDs associated with FC ports
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
//
// 1. A reference to the top-level ComputerSystem in the FC HBA Profile,
//    which represents the system hosting the HBA, is known as $Host->
//
//    The host should have a single HBA Profile implementation running
//
// Get a list of all the ports
$Ports->[] = AssociatorNames($Host->, // ObjectName
    "CIM_SystemDevice", // AssocClass
    "CIM_FCPort", // ResultClass
    "GroupComponent", // Role
    "PartComponent") // ResultRole

if ($Ports->[] == null || $Ports->[].length == 0) {
    <ERROR! No FC Ports on the host system!>
}

for (#i in $Ports->[] ) {
    // Get a list of Alarms associated with each port
    // Should only be one (or zero) alarms
    $Alarms->[] = AssociatorNames($Ports->[#i], // ObjectName
        "CIM_AssociatedAlarm", // AssocClass
        "CIM_Alarm", // ResultClass
        "Antecedent", // Role
        "Dependent") // ResultRole
    if ($Alarms->[] == null || $Ports->[].length == 0) {
        <EXIT: HBA Instrumentation does not support LED blink>
    } else {
        if ( $Alarms->[].length > 1) {
            <ERROR! More than 1 alarm associated with a port>
        }
    }

    // invoke the method to blink the alarm
    %InArguments["RequestedAlarmState"] = "Alternating"
    #MethodReturn = InvokeMethod(
        $Alarms->[0],
        "SetAlarmState",

```

```

    %InArguments)

    if(#MethodReturn != 0)
    {
        <ERROR! SetAlarmState (blink LED) method Failed >
    }
}

```

## 5.7 CIM Elements

Table 19 describes the CIM elements for FC HBA.

**Table 19 - CIM Elements for FC HBA**

Element Name	Requirement	Description
5.7.1 CIM_AlarmDevice	Optional	optional.
5.7.2 CIM_AssociatedAlarm	Optional	
5.7.3 CIM_ComputerSystem	Mandatory	The host system containing the HBAs.
5.7.4 CIM_ControlledBy	Mandatory	
5.7.5 CIM_ElementCapabilities (Capabilities to FCPort)	Optional	
5.7.6 CIM_ElementCapabilities (Capabilities to System)	Optional	
5.7.7 CIM_ElementSettingData	Optional	
5.7.8 CIM_ElementSoftwareIdentity (Driver)	Mandatory	
5.7.9 CIM_ElementSoftwareIdentity (FCode/ BIOS)	Optional	
5.7.10 CIM_ElementSoftwareIdentity (Firmware)	Optional	
5.7.11 CIM_FCPort	Mandatory	
5.7.12 CIM_HostedCollection	Optional	Associates the LogicalPortGroup (Fibre Channel Node) to the hosting System.
5.7.13 CIM_HostedService	Optional	
5.7.14 CIM_InstalledSoftwareIdentity (Driver)	Optional	
5.7.15 CIM_LogicalPortGroup	Optional	Collection of Fibre Channel ports that share a Node WWN.
5.7.16 CIM_MemberOfCollection	Optional	Associates FCPort to the LogicalPortGroup.
5.7.17 CIM_OSStorageNameBinding	Optional	
5.7.18 CIM_PhysicalPackage	Mandatory	

**Table 19 - CIM Elements for FC HBA**

Element Name	Requirement	Description
5.7.19 CIM_PortController	Mandatory	
5.7.20 CIM_Product	Mandatory	
5.7.21 CIM_ProductPhysicalComponent	Mandatory	
5.7.22 CIM_Realizes	Mandatory	
5.7.23 CIM_ServiceAvailableToElement	Optional	
5.7.24 CIM_SoftwareIdentity (Driver)	Mandatory	Driver.
5.7.25 CIM_SoftwareIdentity (FCode/BIOS)	Optional	FCode/BIOS.
5.7.26 CIM_SoftwareIdentity (Firmware)	Optional	Firmware.
5.7.27 CIM_StorageNameBinding	Optional	
5.7.28 CIM_StorageNameBindingCapabilities	Optional	
5.7.29 CIM_StorageNameBindingService	Optional	
5.7.30 CIM_SystemDevice (Associates System to PortController)	Mandatory	
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_PortController	Optional	PortController (HBA) Creation. See <i>Storage Management Technical Specification, Part 6 Host Elements, 1.4.0 Rev 6</i> 5.6.1 Discovery HBA Topology and Attributes
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_PortController	Optional	PortController (HBA) Removal.

**5.7.1 CIM\_AlarmDevice**

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 20 describes class CIM\_AlarmDevice.

**Table 20 - SMI Referenced Properties/Methods for CIM\_AlarmDevice**

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

**Table 20 - SMI Referenced Properties/Methods for CIM\_AlarmDevice**

Properties	Requirement	Description & Notes
VisibleAlarm	Mandatory	Shall be 'true'.
Urgency	Mandatory	Shall be 3 (Alternating).
SetAlarmState()	Mandatory	

**5.7.2 CIM\_AssociatedAlarm**

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 21 describes class CIM\_AssociatedAlarm.

**Table 21 - SMI Referenced Properties/Methods for CIM\_AssociatedAlarm**

Properties	Requirement	Description & Notes
Antecedent	Mandatory	Reference to AlarmDevice.
Dependent	Mandatory	Reference to FCPort (defined in FC Initiator Ports profile).

**5.7.3 CIM\_ComputerSystem**

The host system containing the HBAs.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 22 describes class CIM\_ComputerSystem.

**Table 22 - SMI Referenced Properties/Methods for CIM\_ComputerSystem**

Properties	Requirement	Description & Notes
CreationClassName	Mandatory	
Name	Mandatory	The name of the host.
ElementName	Mandatory	
NameFormat	Mandatory	
OtherIdentifyingInfo	Mandatory	
IdentifyingDescriptions	Mandatory	Shall include 'Ipv4 Address', 'Ipv6 Address', or 'Fully Qualified Domain Name'.

**Table 22 - SMI Referenced Properties/Methods for CIM\_ComputerSystem**

Properties	Requirement	Description & Notes
Dedicated	Mandatory	Shall be 0 (Not Dedicated).
OtherDedicatedDescriptions	Optional	
OperationalStatus	Mandatory	

**5.7.4 CIM\_ControlledBy**

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 23 describes class CIM\_ControlledBy.

**Table 23 - SMI Referenced Properties/Methods for CIM\_ControlledBy**

Properties	Requirement	Description & Notes
Dependent	Mandatory	Reference to FCPort (defined in FC Initiator Ports profile).
Antecedent	Mandatory	Reference to PortController.

**5.7.5 CIM\_ElementCapabilities (Capabilities to FCPort)**

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 24 describes class CIM\_ElementCapabilities (Capabilities to FCPort).

**Table 24 - SMI Referenced Properties/Methods for CIM\_ElementCapabilities (Capabilities to FCPort)**

Properties	Requirement	Description & Notes
Capabilities	Mandatory	Reference to CIM_StorageNameBindingCapabilities.
ManagedElement	Mandatory	Reference to FCPort (defined in FC Initiator Ports profile).

**5.7.6 CIM\_ElementCapabilities (Capabilities to System)**

Created By: Static

Modified By: Static



Deleted By: Static  
Requirement: Optional

Table 25 describes class CIM\_ElementCapabilities (Capabilities to System).

**Table 25 - SMI Referenced Properties/Methods for CIM\_ElementCapabilities (Capabilities to System)**

Properties	Requirement	Description & Notes
Capabilities	Mandatory	Reference to CIM_StorageNameBindingCapabilities.
ManagedElement	Mandatory	Reference to ComputerSystem.

### 5.7.7 CIM\_ElementSettingData

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

Table 26 describes class CIM\_ElementSettingData.

**Table 26 - SMI Referenced Properties/Methods for CIM\_ElementSettingData**

Properties	Requirement	Description & Notes
ManagedElement	Mandatory	Reference to CIM_StorageNameBindingService.
SettingData	Mandatory	Reference to StorageNameBinding or OSStorageNameBinding.

### 5.7.8 CIM\_ElementSoftwareIdentity (Driver)

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

Table 27 describes class CIM\_ElementSoftwareIdentity (Driver).

**Table 27 - SMI Referenced Properties/Methods for CIM\_ElementSoftwareIdentity (Driver)**

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

### 5.7.9 CIM\_ElementSoftwareIdentity (FCode/BIOS)

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

Table 28 describes class CIM\_ElementSoftwareIdentity (FCode/BIOS).

**Table 28 - SMI Referenced Properties/Methods for CIM\_ElementSoftwareIdentity (FCode/BIOS)**

Properties	Requirement	Description & Notes
Antecedent	Mandatory	Reference to FCode/BIOS SoftwareIdentity.
Dependent	Mandatory	Reference to the PortController.

### 5.7.10 CIM\_ElementSoftwareIdentity (Firmware)

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

Table 29 describes class CIM\_ElementSoftwareIdentity (Firmware).

**Table 29 - SMI Referenced Properties/Methods for CIM\_ElementSoftwareIdentity (Firmware)**

Properties	Requirement	Description & Notes
Antecedent	Mandatory	Reference to Firmware SoftwareIdentity.
Dependent	Mandatory	Reference to the PortController.

### 5.7.11 CIM\_FCPort

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

Table 30 describes class CIM\_FCPort.

**Table 30 - SMI Referenced Properties/Methods for CIM\_FCPort**

Properties	Requirement	Description & Notes
PermanentAddress	Mandatory	Override PermanentAddress to be mandatory in this profile.

**5.7.12 CIM\_HostedCollection**

Associates the LogicalPortGroup (Fibre Channel Node) to the hosting System.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 31 describes class CIM\_HostedCollection.

**Table 31 - SMI Referenced Properties/Methods for CIM\_HostedCollection**

Properties	Requirement	Description & Notes
Antecedent	Mandatory	Reference to ComputerSystem.
Dependent	Mandatory	

**5.7.13 CIM\_HostedService**

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 32 describes class CIM\_HostedService.

**Table 32 - SMI Referenced Properties/Methods for CIM\_HostedService**

Properties	Requirement	Description & Notes
Antecedent	Mandatory	Reference to ComputerSystem.
Dependent	Mandatory	Reference to CIM_StorageNameBindingService.

**5.7.14 CIM\_InstalledSoftwareIdentity (Driver)**

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 33 describes class CIM\_InstalledSoftwareIdentity (Driver).

**Table 33 - SMI Referenced Properties/Methods for CIM\_InstalledSoftwareIdentity (Driver)**

Properties	Requirement	Description & Notes
InstalledSoftware	Mandatory	Reference to Driver SoftwareIdentity.
System	Mandatory	Reference to ComputerSystem.

### 5.7.15 CIM\_LogicalPortGroup

Represents the Fibre Channel Node. Associated to the host system by the HostedCollection Association.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 34 describes class CIM\_LogicalPortGroup.

**Table 34 - SMI Referenced Properties/Methods for CIM\_LogicalPortGroup**

Properties	Requirement	Description & Notes
InstanceID	Mandatory	Opaque.
Name	Mandatory	Fibre Channel Node WWN.
NameFormat	Mandatory	"WWN".
ElementName	Mandatory	Node Symbolic Name if available. Otherwise NULL. If the underlying implementation includes characters that are illegal in CIM strings, then truncate before the first of those characters.

### 5.7.16 CIM\_MemberOfCollection

Associates FCPort to the LogicalPortGroup.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 35 describes class CIM\_MemberOfCollection.

**Table 35 - SMI Referenced Properties/Methods for CIM\_MemberOfCollection**

Properties	Requirement	Description & Notes
Collection	Mandatory	Reference to LogicalPortGroup.
Member	Mandatory	Reference to FCPort (defined in FC Initiator Ports profile).

### 5.7.17 CIM\_OSStorageNameBinding

The structure representing an FC persistent binding when the caller specifies the OS Device name. Description column includes mapping to FC API properties.

Created By: Extrinsic: CIM\_StorageNameBindingService.CreateOSStorageNameBinding

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 36 describes class CIM\_OSStorageNameBinding.

**Table 36 - SMI Referenced Properties/Methods for CIM\_OSStorageNameBinding**

Properties	Requirement	Description & Notes
BindingType	Mandatory	API HBA_BIND_TYPE. 2=FCApiBindToDID, 3=FCApiBindToWWPN, 4=FCApiBindToWWNN, 5=BindToLUID.
BindAllLogicalUnits	Mandatory	API HBA_BIND_TARGETS. If true, then all target logical units are bound to the OS. Not valid to set this if BindingType is BindToLUID.
Hide	Mandatory	Shall be false.
TargetName	Mandatory	API FCID, NodeWWN, PortWWN or HBA_LUID. If BindingType is FcApiBindToDID, TargetName holds a hexadecimal-encoded representation of the 32-bit D_ID and corresponds to FC API HBA_FCPID.FcId. If BindingType is FcApiBindToWWPN or FcApiBindToWWNN, TargetName holds a hexadecimal-encoded representation of the 64-bit FC Port or Node World Wide Name. If BindingType is BindToLUID, TargetName holds a SCSI Logical Unit Name from Inquiry VPD page 83, Association 0 as defined in SCSI Primary Commands. If the identifier descriptor (in the SCSI response) has Code Set Binary, then TargetName is its hexadecimal-encoded value.
Status	Mandatory	HBA_FCPBINDING2.Status.
OSDeviceName	Mandatory	
OSAddressesValid	Mandatory	Indicates whether OSBusNumber, OSTargetNumber, and OSLUN properties are valid.
OSBusNumber	Mandatory	API SCSIBusNumber.
OSTargetNumber	Mandatory	API osTargetId.
OSLUN	Mandatory	API osLUN.
LocalPortNameType	Mandatory	Shall be 2 - FC Port WWN.
LocalPortName	Mandatory	initiator port WWN.

### 5.7.18 CIM\_PhysicalPackage

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

Table 37 describes class CIM\_PhysicalPackage.

**Table 37 - SMI Referenced Properties/Methods for CIM\_PhysicalPackage**

Properties	Requirement	Description & Notes
Manufacturer	Mandatory	
Model	Mandatory	
Tag	Mandatory	
CreationClassName	Mandatory	
SerialNumber	Optional	

#### 5.7.19 CIM\_PortController

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

Table 38 describes class CIM\_PortController.

**Table 38 - SMI Referenced Properties/Methods for CIM\_PortController**

Properties	Requirement	Description & Notes
SystemCreationClassName	Mandatory	
SystemName	Mandatory	
CreationClassName	Mandatory	
DeviceID	Mandatory	
ControllerType	Mandatory	Shall be 4 (FC).

#### 5.7.20 CIM\_Product

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

Table 39 describes class CIM\_Product.

**Table 39 - SMI Referenced Properties/Methods for CIM\_Product**

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

**5.7.21 CIM\_ProductPhysicalComponent**

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 40 describes class CIM\_ProductPhysicalComponent.

**Table 40 - SMI Referenced Properties/Methods for CIM\_ProductPhysicalComponent**

Properties	Requirement	Description & Notes
GroupComponent	Mandatory	Reference to Product.
PartComponent	Mandatory	Reference to PhysicalPackage.

**5.7.22 CIM\_Realizes**

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 41 describes class CIM\_Realizes.

**Table 41 - SMI Referenced Properties/Methods for CIM\_Realizes**

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

### 5.7.23 CIM\_ServiceAvailableToElement

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

Table 42 describes class CIM\_ServiceAvailableToElement.

**Table 42 - SMI Referenced Properties/Methods for CIM\_ServiceAvailableToElement**

Properties	Requirement	Description & Notes
ServiceProvided	Mandatory	Reference to CIM_StorageNameBindingService.
UserOfService	Mandatory	Reference to FCPort (defined in FC Initiator Ports profile).

### 5.7.24 CIM\_SoftwareIdentity (Driver)

SoftwareIdentity representing the Driver software. This SoftwareIdentity is mandatory and may be associated to the hosting system using InstalledSoftwareIdentity in addition to being associated to the PortController via ElementSoftwareIdentity.

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

Table 43 describes class CIM\_SoftwareIdentity (Driver).

**Table 43 - SMI Referenced Properties/Methods for CIM\_SoftwareIdentity (Driver)**

Properties	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).

### 5.7.25 CIM\_SoftwareIdentity (FCode/BIOS)

Provides information about the HBA FCode/BIOS.

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



Table 44 describes class CIM\_SoftwareIdentity (FCode/BIOS).

**Table 44 - SMI Referenced Properties/Methods for CIM\_SoftwareIdentity (FCode/BIOS)**

Properties	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 11 (FCODE/BIOS).

### 5.7.26 CIM\_SoftwareIdentity (Firmware)

Provides information about the HBA Firmware.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 45 describes class CIM\_SoftwareIdentity (Firmware).

**Table 45 - SMI Referenced Properties/Methods for CIM\_SoftwareIdentity (Firmware)**

Properties	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 10 (Firmware).

### 5.7.27 CIM\_StorageNameBinding

The structure representing an FC persistent binding when the driver/platform implicitly creates the OS device name. Description column includes mapping to FC API properties.

Created By: Extrinsic: CIM\_StorageNameBindingService.CreateStorageNameBinding

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 46 describes class CIM\_StorageNameBinding.

**Table 46 - SMI Referenced Properties/Methods for CIM\_StorageNameBinding**

Properties	Requirement	Description & Notes
BindingType	Mandatory	API HBA_BIND_TYPE. 2=FCApiBindToDID, 3=FCApiBindToWWPN, 4=FCApiBindToWWNN, 5=BindToLUID.
BindAllLogicalUnits	Mandatory	API HBA_BIND_TARGETS. If true, then all target logical units are bound to the OS. Not valid to set this if BindingType is BindToLUID.
Hide	Mandatory	Shall be false.
TargetName	Mandatory	API FCID, NodeWWN, PortWWN or HBA_LUID. If BindingType is FcApiBindToDID, TargetName holds a hexadecimal-encoded representation of the 32-bit D_ID and corresponds to FC API HBA_FCPIID.FcId. If BindingType is FcApiBindToWWPN or FcApiBindToWWNN, TargetName holds a hexadecimal-encoded representation of the 64-bit FC Port or Node World Wide Name. If BindingType is BindToLUID, TargetName holds a SCSI Logical Unit Name from Inquiry VPD page 83, Association 0 as defined in SCSI Primary Commands. If the identifier descriptor (in the SCSI response) has Code Set Binary, then TargetName is its hexadecimal-encoded value.
Status	Mandatory	HBA_FCPBINDING2.Status.
LocalPortNameType	Mandatory	Shall be 2 - FC Port WWN.
LocalPortName	Mandatory	initiator port WWN.

### 5.7.28 CIM\_StorageNameBindingCapabilities

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 47 describes class CIM\_StorageNameBindingCapabilities.

**Table 47 - SMI Referenced Properties/Methods for CIM\_StorageNameBindingCapabilities**

Properties	Requirement	Description & Notes
ValidBindingTypes	Mandatory	API HBA_BIND_TYPE. Shall include a subset of 2(FcApiBindToDID), 3(FcApiBindToWWPN), 4(FcApiBindToWWNN), or 5(BindToLUID).
ActivateBindingRequiresReset	Mandatory	True if creating a binding requires a system reboot.

**Table 47 - SMI Referenced Properties/Methods for CIM\_StorageNameBindingCapabilities**

Properties	Requirement	Description & Notes
CanMapAddresses	Mandatory	True if the implementation allows overriding OS bus/target/LUN numbers.
CanBindAllLuns	Mandatory	
AutoDiscovery	Mandatory	
CanSetOSDeviceName	Mandatory	

**5.7.29 CIM\_StorageNameBindingService**

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

Table 48 describes class CIM\_StorageNameBindingService.

**Table 48 - SMI Referenced Properties/Methods for CIM\_StorageNameBindingService**

Properties	Requirement	Description & Notes
CreateStorageNameBinding()	Mandatory	
CreateOSStorageNameBinding()	Mandatory	

**5.7.30 CIM\_SystemDevice (Associates System to PortController)**

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

Table 49 describes class CIM\_SystemDevice (Associates System to PortController).

**Table 49 - SMI Referenced Properties/Methods for CIM\_SystemDevice (Associates System to PortController)**

Properties	Requirement	Description & Notes
GroupComponent	Mandatory	Reference to ComputerSystem.
PartComponent	Mandatory	Reference to PortController.

**IMPLEMENTED**

---

---

---

---

## EXPERIMENTAL

### Clause 6: Storage HBA Profile

#### 6.1 Synopsis

**Profile Name:** Storage HBA

**Version:** 1.4.0

**Organization:** SNIA

**CIM Schema Version:** 2.15.0

Table 50 describes the related profiles for Storage HBA.

**Table 50 - 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	
Indication	SNIA	1.4.0	Mandatory	
SAS Initiator Ports	SNIA	1.4.0	Support for at least one is mandatory.	
FC Initiator Ports	SNIA	1.4.0		

#### 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 specification, DMTF specifications, or the DMTF CIM schema specification.

#### 6.3 Implementation

##### 6.3.1 Health and Fault Management Consideration

None in this version of the standard.

##### 6.3.2 Cascading Considerations

None in this version of the 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 PhysicalPackage (or a subclass such as Card), as shown in Figure 14. Physical Package shall be associated to PortController via Realizes and Product via ProductPhysicalComponent. If the instrumentation is running in a guest virtual machine and representing a virtual HBA, PhysicalPackage, Realizes, and ProductPhysicalComponent shall not be instantiated. See 6.3.12 for additional virtual system considerations.

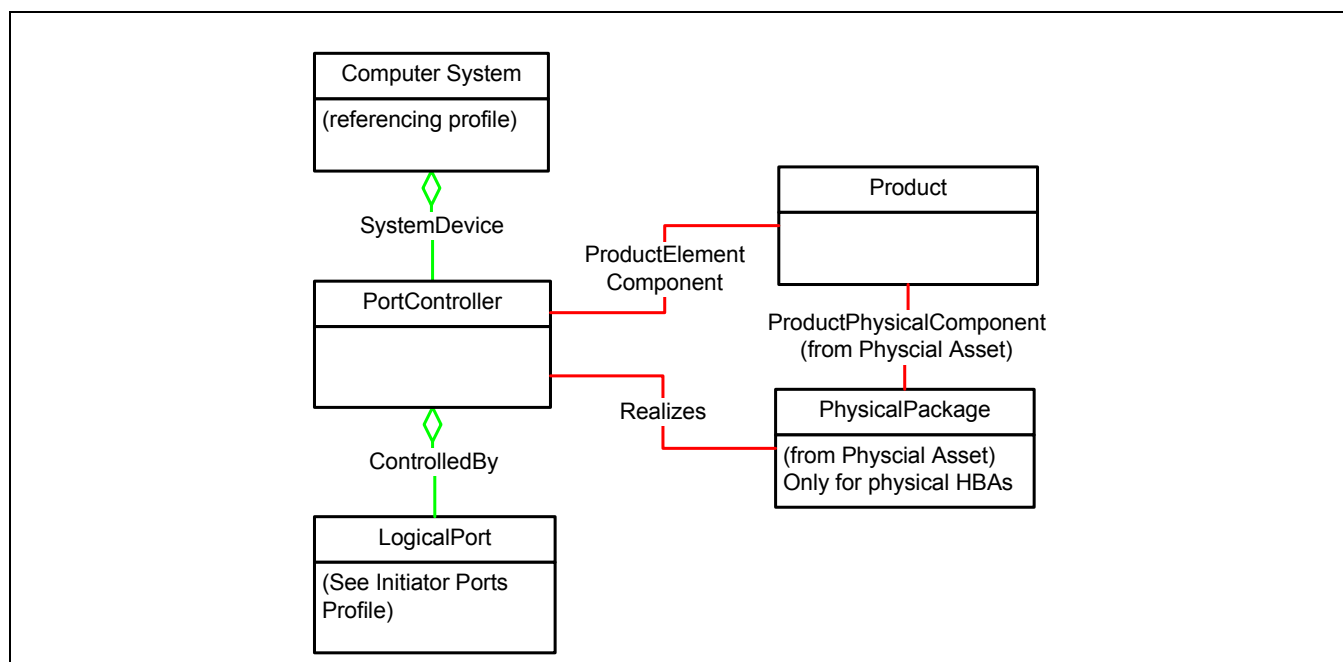


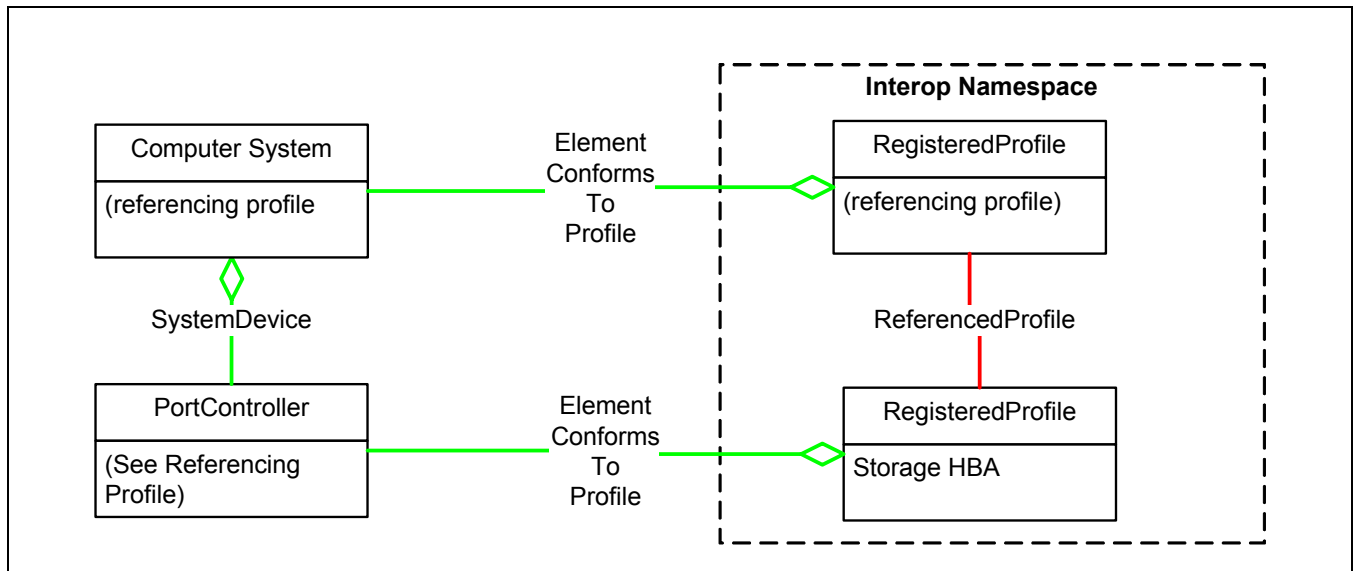
Figure 14 - Model Overview

#### 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 15.



**Figure 15 - 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 is 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 16, 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.

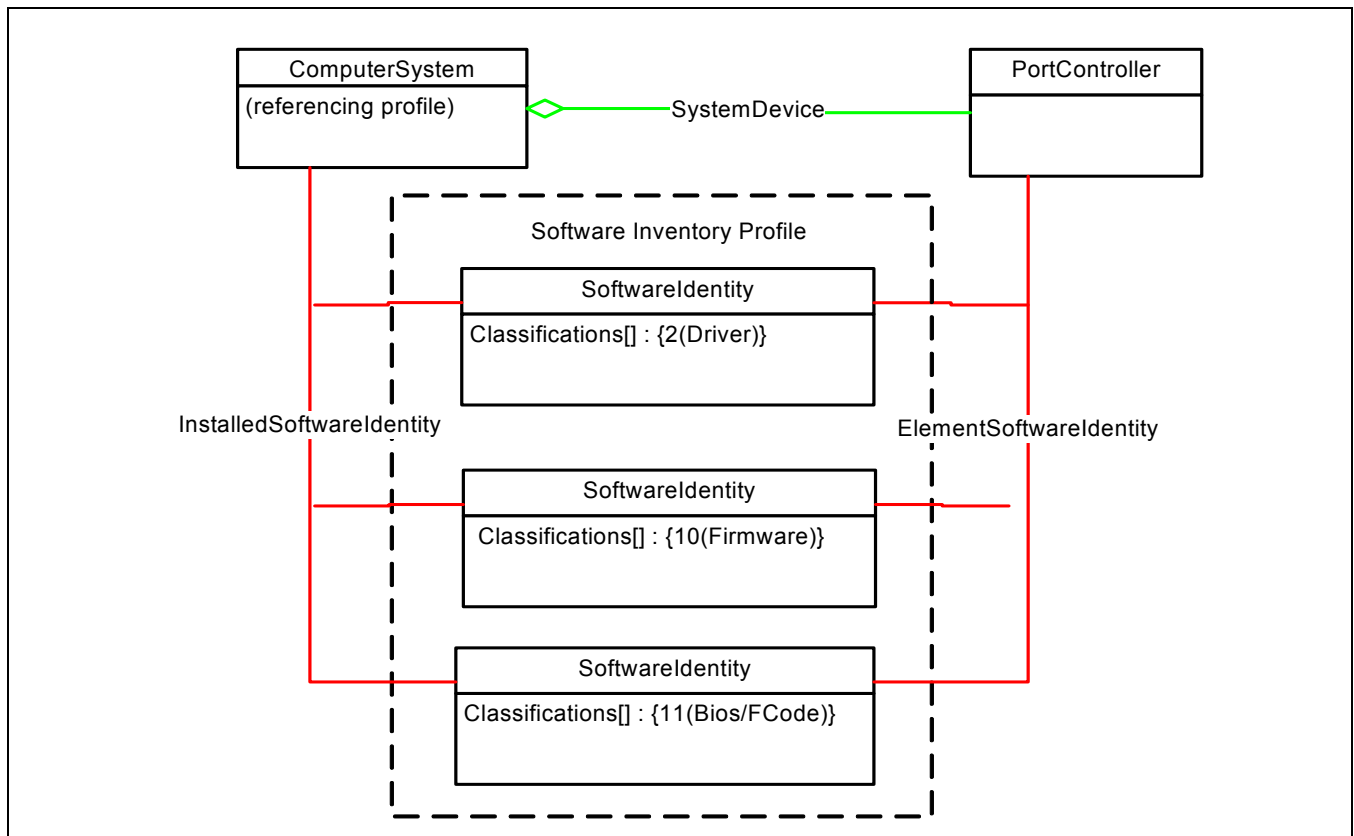


Figure 16 - Software Inventory Profile in Storage HBA

### 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.

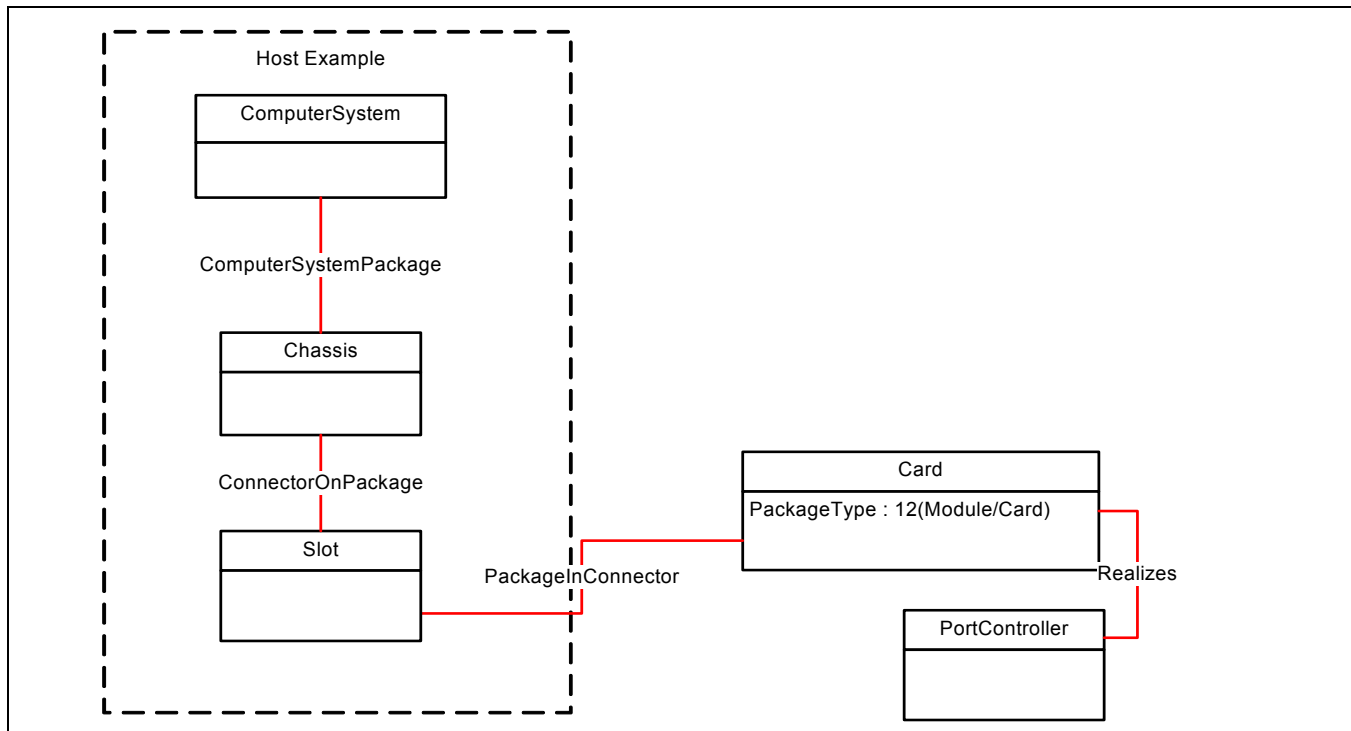
### 6.3.9 HBA Hot Swap Events

The implementation may optionally support asynchronous notification of HBA inserts and removals using the InstCreation and InstDeletion indications (see Table 53, “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 17 illustrates an HBA Card with Physical Classes.



**Figure 17 - 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 14.3.1 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.

## 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 51 - 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 52 - CIM\_SystemDevice**

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

## 6.5 Use Cases

None in this version of the standard.

## 6.6 CIM Elements

Table 53 describes the CIM elements for Storage HBA.

**Table 53 - 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.
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 54 describes class CIM\_ControlledBy.

**Table 54 - SMI Referenced Properties/Methods for CIM\_ControlledBy**

Properties	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

Table 55 describes class CIM\_PortController.

**Table 55 - SMI Referenced Properties/Methods for CIM\_PortController**

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

### 6.6.3 CIM\_Product

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

Table 56 describes class CIM\_Product.

**Table 56 - SMI Referenced Properties/Methods for CIM\_Product**

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

#### 6.6.4 CIM\_ProductElementComponent

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 57 describes class CIM\_ProductElementComponent.

**Table 57 - SMI Referenced Properties/Methods for CIM\_ProductElementComponent**

Properties	Requirement	Description & Notes
GroupComponent	Mandatory	
PartComponent	Mandatory	

#### 6.6.5 CIM\_Realizes

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 58 describes class CIM\_Realizes.

**Table 58 - SMI Referenced Properties/Methods for CIM\_Realizes**

Properties	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 59 describes class CIM\_SystemDevice.

**Table 59 - SMI Referenced Properties/Methods for CIM\_SystemDevice**

Properties	Requirement	Description & Notes
GroupComponent	Mandatory	
PartComponent	Mandatory	

## **EXPERIMENTAL**

---

---





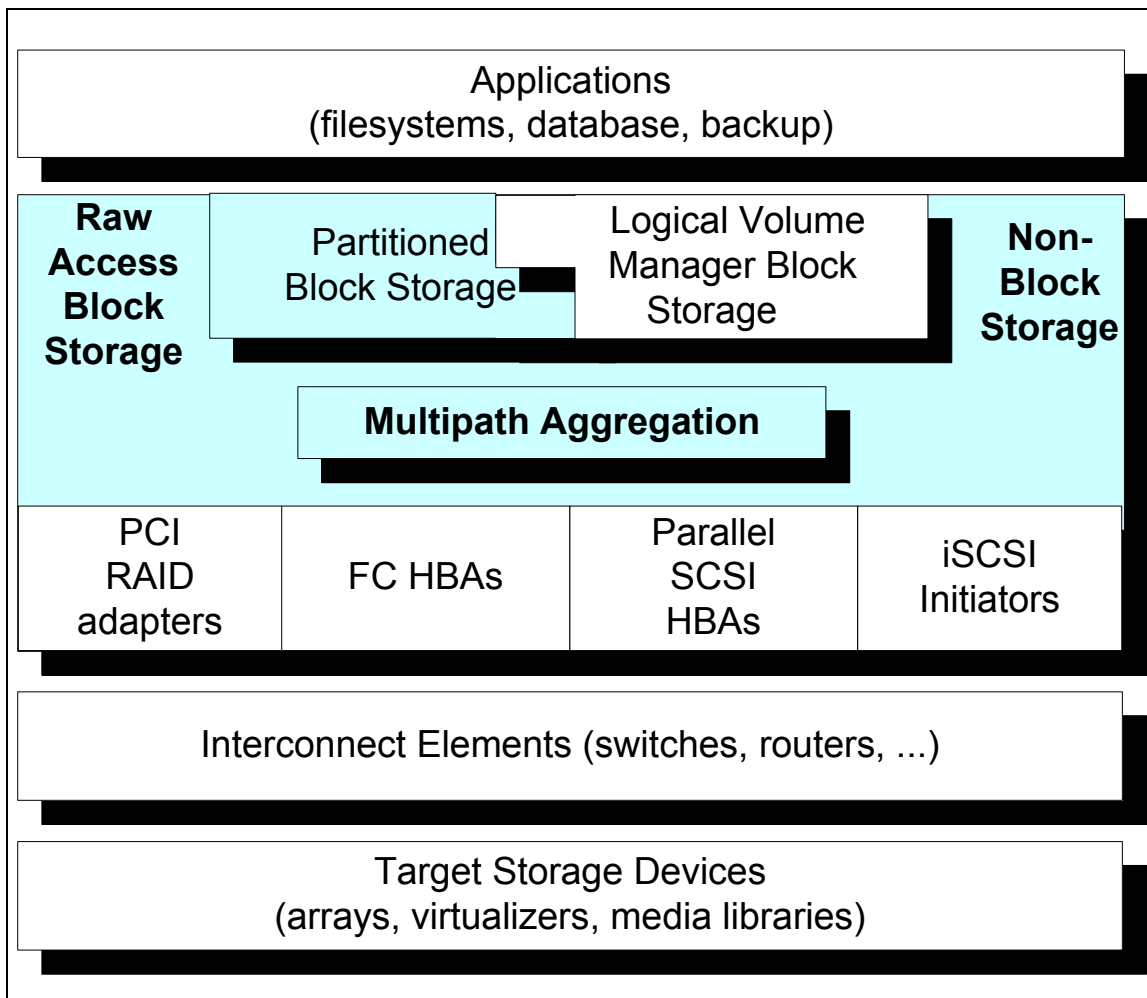
**EXPERIMENTAL**

**Clause 7: Host Discovered Resources Profile**

**7.1 Description**

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 18 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 18 - 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.

### 7.1.1 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 not 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 Clause 4: Disk Partition Subprofile.
- Multipath Management - see Clause 10: SCSI Multipath Management Subprofile.
- Discovery of Hardware Resources - see 7.1.2.

### 7.1.2 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 Clause 7: Correlatable and Durable Names, Section 7.6.6 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 Clause 7: Correlatable and Durable Names, Section 7.6.2 applies, but instead of using the Name and NameNamespace properties, the information is contained in the OtherIdentifyingInfo and IdentifyingDescriptions array properties. The valid strings

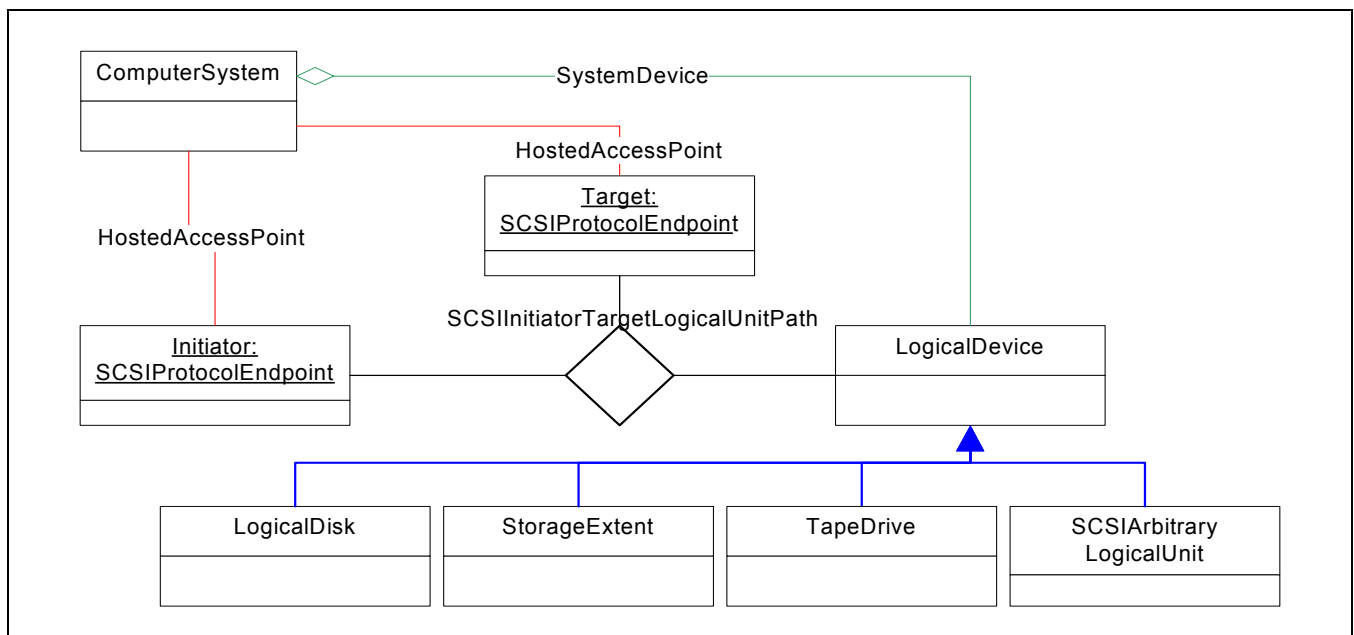
for IdentifyingDescriptions are exactly those described for NameNamespace in Clause 7: Correlatable and Durable Names, Section 7.6.2.

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.

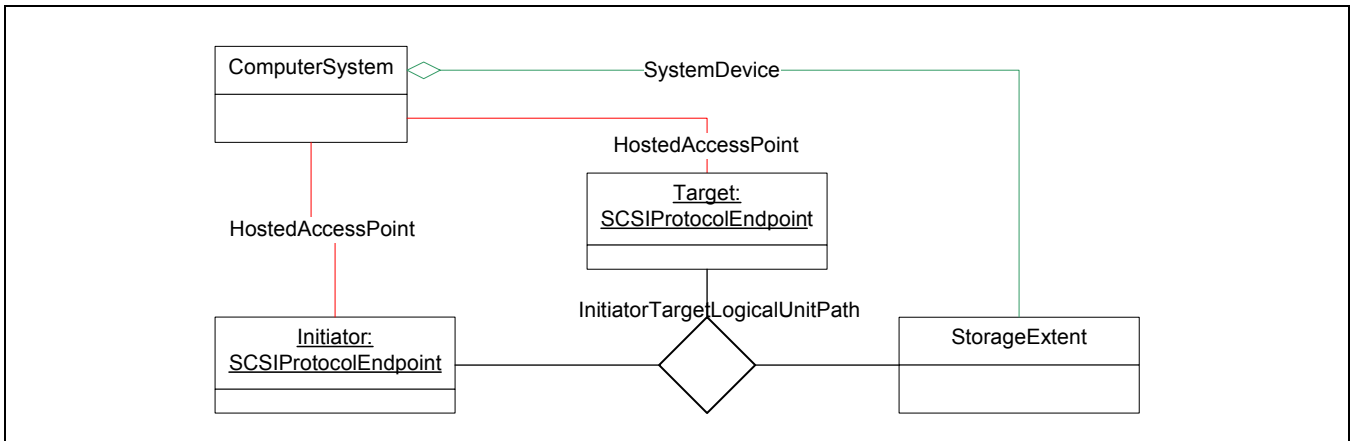
Figure 19 is a class diagram for Host Discovered Resources. SCSIProtocolEndpoint represents the SCSI logical port, either initiator or target. The transports type (e.g., FC, iSCSI) is specified in SCSIProtocolEndpoint ConnectionType property.



**Figure 19 - Host Discovered Resources Class Diagram**

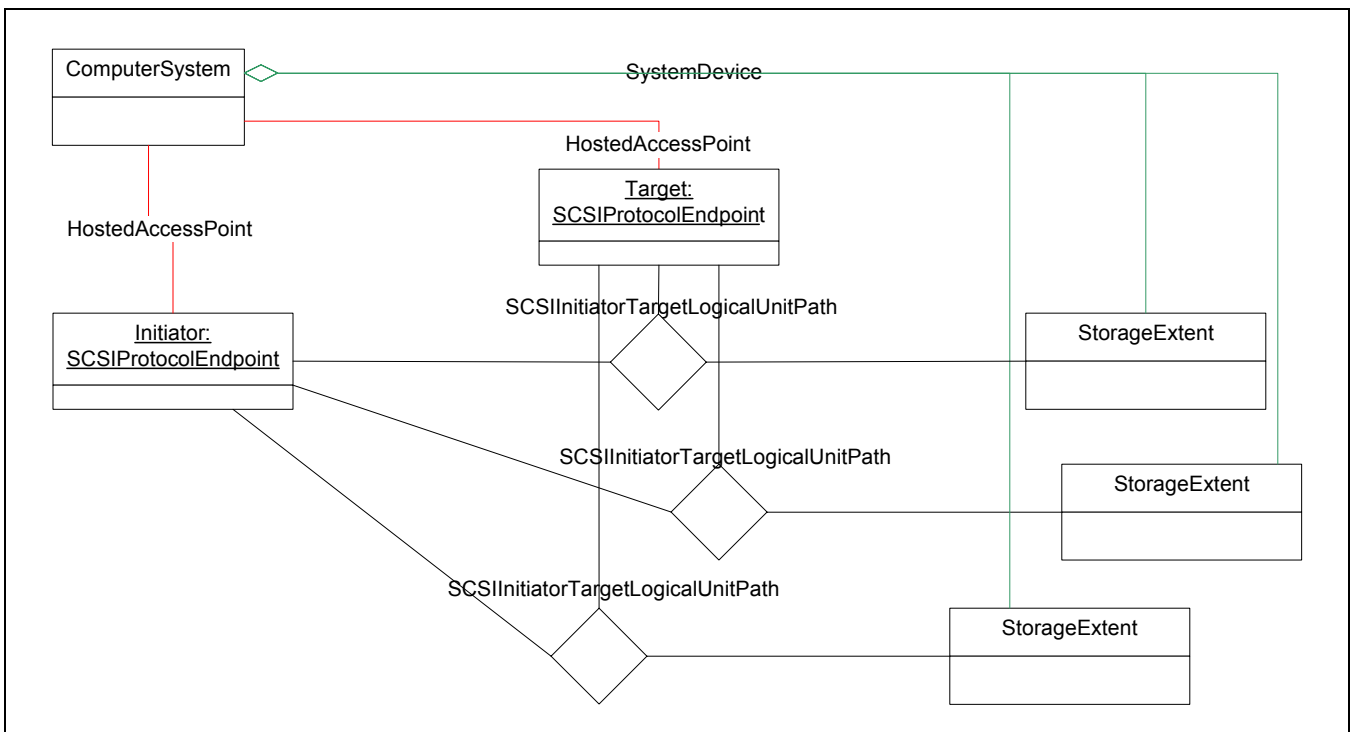
The initiator ProtocolEndpoint and each target ProtocolEndpoint and LogicalDevice are associated by SCSIIInitiatorTargetLogicalUnitPath.

Consider a few concrete cases. Figure 20 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 20 - Single SPI Disk Model**

Figure 21 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.

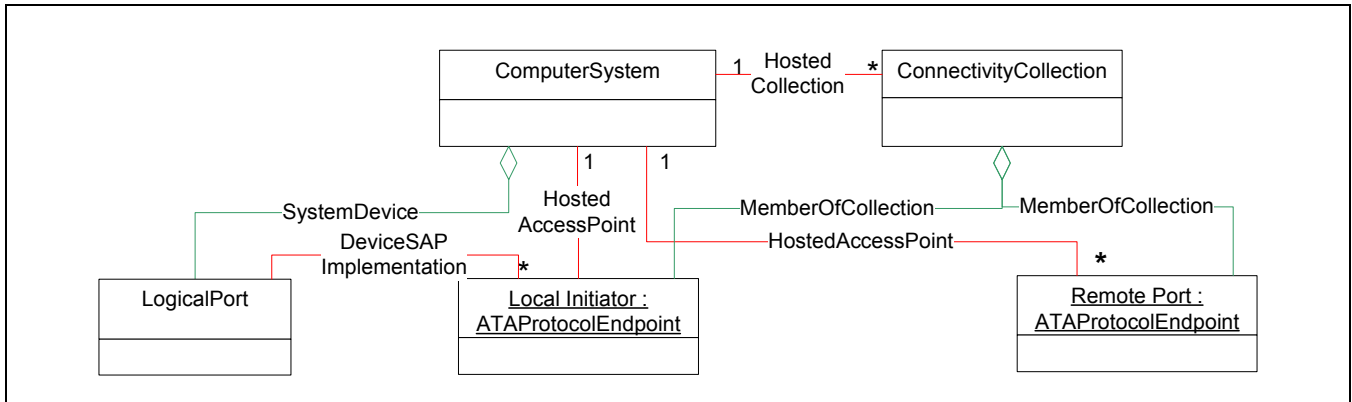


**Figure 21 - Three FCP Logical Unit Instance Diagram**

The Multipath Subprofile describes more complicated multipath configurations. See Figure 40.

**Model for ATA Protocol Resources**

The model for ATA, shown in Figure 22, uses a ConnectivityCollection of ATAProtocolEndpoints.

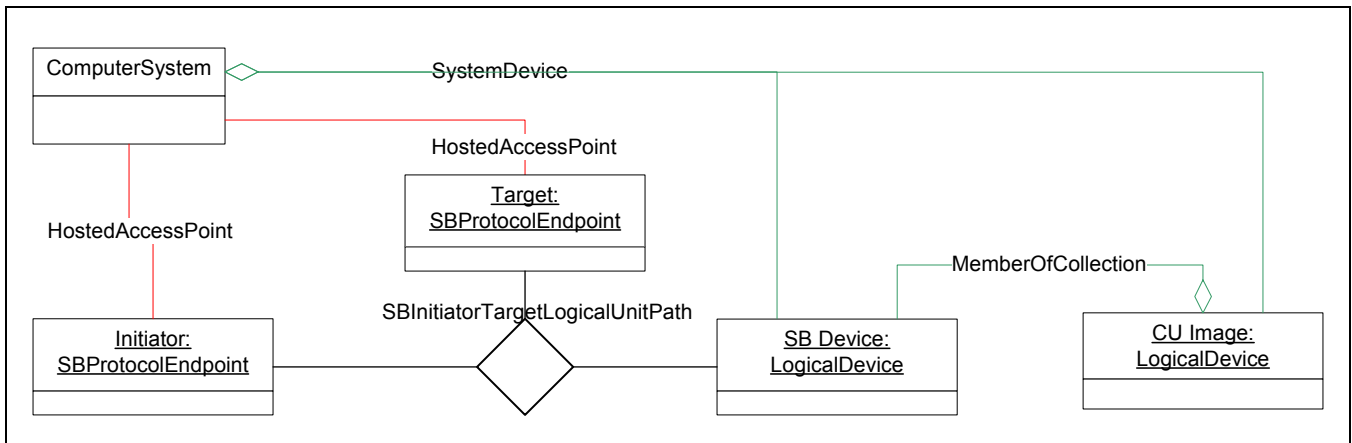


**Figure 22 - ATA Discovered Resource Model**

**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 23 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 SBCUIImage.

Figure 23 depicts SB Host Discovered Resources.



**Figure 23 - 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 mod-

eled between the StorageExtents and LogicalDisks. For more details, see Clause 4: Disk Partition Subprofile. The requirement for disk partitions is reflected by the presence of DiskPartitionConfigurationCapabilities.

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 Supported Subprofiles and Packages

Table 60 describes the supported profiles for Host Discovered Resources.

**Table 60 - Supported Profiles for Host Discovered Resources**

Profile Name	Organization	Version	Requirement	Description
SCSI Multipath Management	SNIA	1.3.0	Optional	
SB Multipath Management	SNIA	1.0.0	Optional	
Disk Partition	SNIA	1.3.0	Optional	

## 7.5 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.

## 7.6 Client Considerations and Recipes

### 7.6.1 Determine which exported extents are impacted by removal of a physical extent

```
//
// Description:
// Determine which exported extents are impacted by removal of a
// physical extent. Note that in this Profile, "exported extent"
// is synonymous with LogicalDisk.
//
// Pre-Conditions:
// $Host holds a ref to the (top-level) ComputerSystem
// $Disk holds a reference to the StorageExtent to be removed.
//
// In SMI-S, anything exposed to applications (or LVMs) as an OS
// disk is modeled as LogicalDisk. On platforms that support partitions,
// if a disk is partitioned, the disk itself is modeled as StorageExtent.
// Each partition that is exposed is modeled as LogicalDisk Based on a
// GenericDiskPartition BasedOn StorageExtent (the disk). Some platforms
// allow a partition to be sub-partitioned; this is modeled as
// LogicalDisk (exposed) BasedOn DiskPartition (top-tier) BasedOn
// DiskPartition (bottom tier) BasedOn StorageExtent (the disk).
// On systems without disk partitions, a LogicalDisk instance models
// the entire usable disk capacity.
//
// CIM models each exposed partition as a LogicalDisk BasedOn a
// DiskPartition (mapped 1-1). Many DiskPartitions can be based
// on the same underlying StorageExtent (either a disk or another
// partition). The valid configurations
// are
// 1 - $Disk is actually exposed as a LogicalDisk (LD)
// 2 - Single-tier partitioning, LD based on DiskPartition (DP) BasedOn SE
//    (StorageExtent)
// 3 - Two-tier partitioning - LD BasedOn DP BasedOn DP BasedOn SE
//
// The recipe below uses recursion to find all StorageExtents (the
// super-class of LogicalDisk and DiskPartition) based on $Disk, then
// follows BasedOn associations untill it hits LogicalDisks.

sub REF[] GetImpactedExtents($Extent)
{
    // A logical disk can't contain any partitions - if $Extent
```

```

// is a LogicalDisk, add it to the $ImpactedExtents list
// and return.
if ($Extent ISA CIM_LogicalDisk) {
    push ($ImpactedExtents[], $Extent)
    return ($ImpactedExtents[]->)
}

// For non LogicalDisks, get the list of all extents based on $Extent
$SuperExtents[] = AssociatorNames(
    $Disk,
    "CIM_BasedOn",
    null, // ResultClass
    "Antecedent" // Role
    "Dependent") // ResultRole

// For each extent that depends on $Extent, recurse
for #i in $SuperExtents[] {
    $ImpactedExtents = &GetImpactedExtents($SuperExtents[#i])
}
return $ImpactedExtents[]->
}

$ImpactedLDs = &GetImpactedExtents($Disk)

```

## 7.7 Registered Name and Version

Host Discovered Resources version 1.2.0

## 7.8 CIM Elements

Table 61 describes the CIM elements for Host Discovered Resources.

**Table 61 - CIM Elements for Host Discovered Resources**

Element Name	Requirement	Description
7.8.1 CIM_ATAInitiatorTargetLogicalUnitPath	Optional	Associates initiator and target ATAProtocolEndpoints to a logical unit.
7.8.2 CIM_ATAProtocolEndpoint	Optional	
7.8.3 CIM_ComputerSystem	Mandatory	'Top level' system that hosts the resources.
7.8.4 CIM_HostedAccessPoint	Mandatory	This association links all ProtocolEndpoints to the scoping system.
7.8.5 CIM_LogicalDisk	Optional	Represents a block logical unit that is exposed to applications such as file systems without being partitioned.
7.8.6 CIM_SCSIArbitraryLogicalUnit	Optional	A SCSI Logical Unit that exists only for management.



**Table 61 - CIM Elements for Host Discovered Resources**

Element Name	Requirement	Description
7.8.7 CIM_SCSIIInitiatorTargetLogicalUnitPath	Optional	Associates initiator and target SCSIProtocolEndpoints to a logical unit.
7.8.8 CIM_SCSIProtocolEndpoint	Optional	
7.8.9 CIM_StorageExtent	Optional	Represents a block logical unit in the host that is partitioned before being exposed to applications.
7.8.10 CIM_SystemDevice	Mandatory	This association links LogicalDevices to the scoping system.
7.8.11 CIM_TapeDrive	Optional	Represents a tape drive logical unit in the host.
7.8.12 SNIA_SBInitiatorTargetLogicalUnitPath	Optional	Associates initiator and target SBProtocolEndpoints to a logical unit.
7.8.13 SNIA_SBProtocolEndpoint	Optional	
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_SCSIIInitiatorTargetLogicalUnitPath	Mandatory	Path creation.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_SCSIIInitiatorTargetLogicalUnitPath	Mandatory	Path deletion.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_SCSIIInitiatorTargetLogicalUnitPath AND SourceInstance.CIM_SCSIIInitiatorTargetLogicalUnitPath::State <> PreviousInstance.CIM_SCSIIInitiatorTargetLogicalUnitPath::State	Mandatory	CQL -Path State change.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_SCSIIInitiatorTargetLogicalUnitPath AND SourceInstance.CIM_SCSIIInitiatorTargetLogicalUnitPath::AdministrativeWeight <> PreviousInstance.CIM_SCSIIInitiatorTargetLogicalUnitPath::AdministrativeWeight	Mandatory	CQL -Path AdministrativeWeight change.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_SCSIIInitiatorTargetLogicalUnitPath AND SourceInstance.CIM_SCSIIInitiatorTargetLogicalUnitPath::AdministrativeOverride <> PreviousInstance.CIM_SCSIIInitiatorTargetLogicalUnitPath::AdministrativeOverride	Mandatory	CQL -Path AdministrativeOverride change.

**7.8.1 CIM\_ATAInitiatorTargetLogicalUnitPath**

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

Table 62 describes class CIM\_ATAInitiatorTargetLogicalUnitPath.

**Table 62 - 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.

### 7.8.2 CIM\_ATAProtocolEndpoint

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

Table 63 describes class CIM\_ATAProtocolEndpoint.

**Table 63 - SMI Referenced Properties/Methods for CIM\_ATAProtocolEndpoint**

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

### 7.8.3 CIM\_ComputerSystem

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

Requirement: Mandatory

Table 64 describes class CIM\_ComputerSystem.

**Table 64 - SMI Referenced Properties/Methods for CIM\_ComputerSystem**

Properties	Flags	Requirement	Description & Notes
CreationClassName		Mandatory	
Name	C	Mandatory	Unique identifier for the system. E.g., IP address.
ElementName		Mandatory	User friendly name.
NameFormat		Mandatory	Format for Name property.
Dedicated		Mandatory	Indicates that this computer system is not a dedicated storage system.

#### 7.8.4 CIM\_HostedAccessPoint

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 65 describes class CIM\_HostedAccessPoint.

**Table 65 - SMI Referenced Properties/Methods for CIM\_HostedAccessPoint**

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

#### 7.8.5 CIM\_LogicalDisk

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 66 describes class CIM\_LogicalDisk.

**Table 66 - SMI Referenced Properties/Methods for CIM\_LogicalDisk**

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	Opaque identifier.
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	

### 7.8.6 CIM\_SCSIArbitraryLogicalUnit

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 67 describes class CIM\_SCSIArbitraryLogicalUnit.

**Table 67 - SMI Referenced Properties/Methods for CIM\_SCSIArbitraryLogicalUnit**

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

**Table 67 - SMI Referenced Properties/Methods for CIM\_SCSIArbitraryLogicalUnit**

Properties	Flags	Requirement	Description & Notes
IdentifyingDescriptions	C	Mandatory	
OperationalStatus		Mandatory	

**7.8.7 CIM\_SCSIInitiatorTargetLogicalUnitPath**

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

Table 68 describes class CIM\_SCSIInitiatorTargetLogicalUnitPath.

**Table 68 - 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.

**7.8.8 CIM\_SCSIProtocolEndpoint**

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

Table 69 describes class CIM\_SCSIProtocolEndpoint.

**Table 69 - SMI Referenced Properties/Methods for CIM\_SCSIProtocolEndpoint**

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

### 7.8.9 CIM\_StorageExtent

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 70 describes class CIM\_StorageExtent.

**Table 70 - SMI Referenced Properties/Methods for CIM\_StorageExtent**

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	Opaque identifier.
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	

### 7.8.10 CIM\_SystemDevice

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 71 describes class CIM\_SystemDevice.

**Table 71 - SMI Referenced Properties/Methods for CIM\_SystemDevice**

Properties	Flags	Requirement	Description & Notes
PartComponent		Mandatory	
GroupComponent		Mandatory	

### 7.8.11 CIM\_TapeDrive

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 72 describes class CIM\_TapeDrive.

**Table 72 - SMI Referenced Properties/Methods for CIM\_TapeDrive**

Properties	Flags	Requirement	Description & Notes
SystemCreationClass Name		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	Opaque identifier.
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	

### 7.8.12 SNIA\_SBInitiatorTargetLogicalUnitPath

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

Table 73 describes class SNIA\_SBInitiatorTargetLogicalUnitPath.

**Table 73 - SMI Referenced Properties/Methods for SNIA\_SBInitiatorTargetLogicalUnitPath**

Properties	Flags	Requirement	Description & Notes
OSDeviceName		Optional	
UsePreferredPath		Optional	Boolean indicating whether preferred path processing is required.
PreferredPath		Optional	Boolean indicating whether 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 valid 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.

### 7.8.13 SNIA\_SBProtocolEndpoint

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

Table 74 describes class SNIA\_SBProtocolEndpoint.

**Table 74 - SMI Referenced Properties/Methods for SNIA\_SBProtocolEndpoint**

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



**Table 74 - SMI Referenced Properties/Methods for SNIA\_SBProtocolEndpoint**

Properties	Flags	Requirement	Description & Notes
ConnectionType		Mandatory	
Role		Mandatory	

**EXPERIMENTAL**

---

---



---

---

## EXPERIMENTAL

### Clause 8: Host Hardware RAID Controller Profile

#### 8.1 Synopsis

**Profile Name:** Host Hardware RAID Controller

**Version:** 1.3.0

**Organization:** SNIA

**CIM Schema Version:** 2.11.0

Table 75 describes the related profiles for Host Hardware RAID Controller.

**Table 75 - Related Profiles for Host Hardware RAID Controller**

Profile Name	Organization	Version	Requirement	Description
Physical Asset	DMTF	1.0.0a	Mandatory	
Block Services	SNIA	1.4.0	Mandatory	
Disk Drive Lite	SNIA	1.4.0	Optional	
Software Inventory	SNIA	1.0.0	Mandatory	
Software Update	DMTF	1.0.0	Optional	
Extent Composition	SNIA	1.2.0	Optional	
Disk Sparing	SNIA	1.4.0	Optional	
DA Target Ports	SNIA	1.4.0	Mandatory	
Indication	SNIA	1.4.0	Mandatory	
Erasure	SNIA	1.2.0	Optional	
Copy Services	SNIA	1.4.0	Optional	
Storage Enclosure	SNIA	1.3.0	Optional	Model for External Disk Enclosures.
Thin Provisioning	SNIA	1.4.0	Optional	
SPI Initiator Ports	SNIA	1.4.0	Support for at least one is mandatory.	
FC Initiator Ports	SNIA	1.4.0		
ATA Initiator Ports	SNIA	1.4.0		
SAS Initiator Ports	SNIA	1.4.0		

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 24, "Host Hardware RAID Controller Package Diagram" shows the relationship between the Host Hardware RAID Profile and its key component profiles.

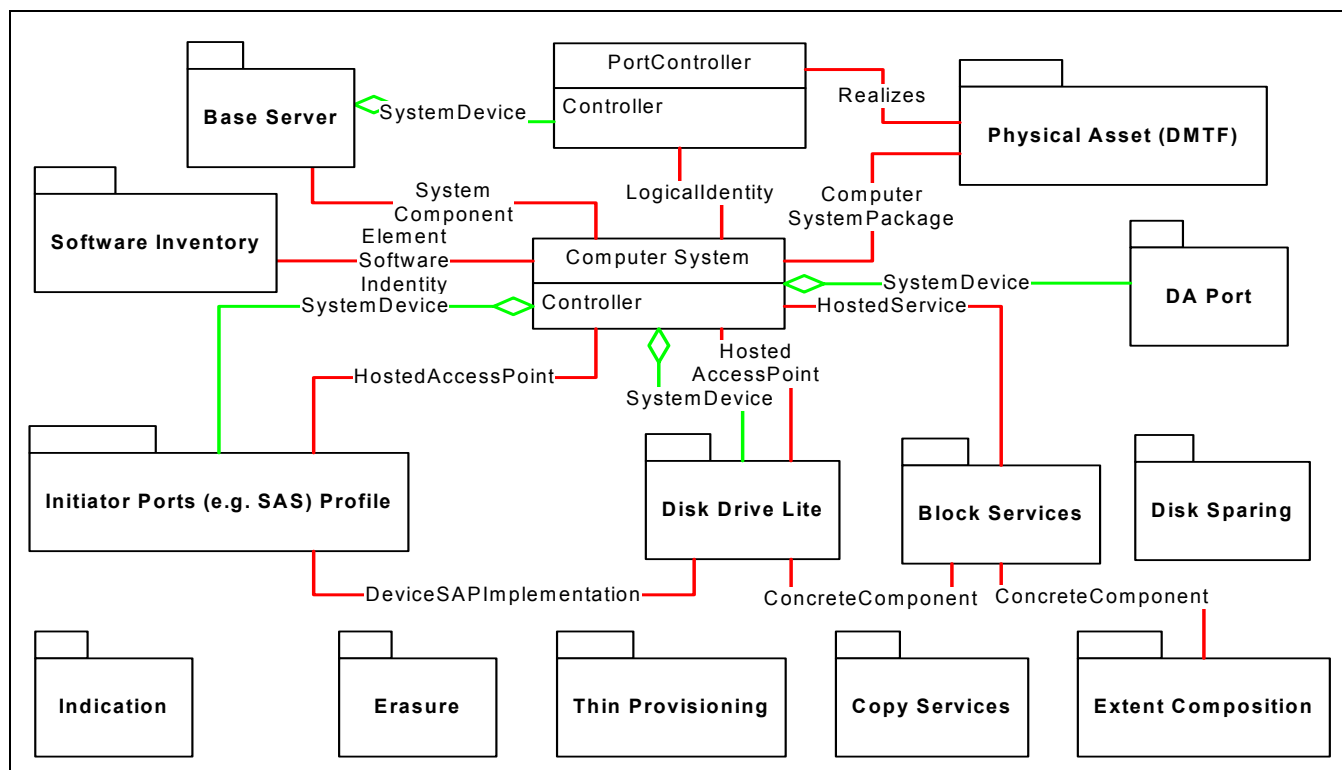


Figure 24 - Host Hardware RAID Controller Package Diagram

## 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 2 Common Profiles, 1.4.0 Rev 6* Clause 35: 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.

### 8.3.2 CIM\_PortController

The PortController class represents an instance of a RAID controller and controls 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

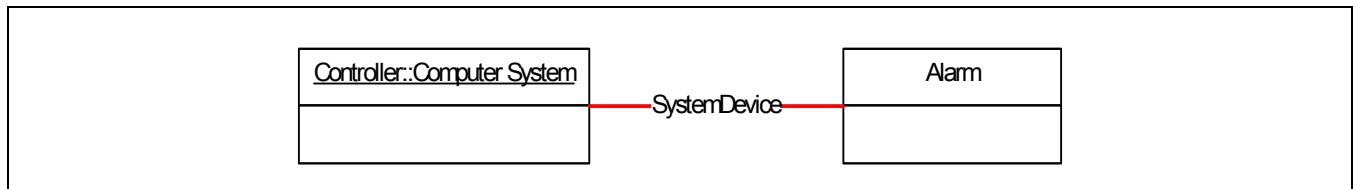
In the Host Hardware RAID Controller profile, the ComputerSystem Class within this profile represents the RAID controller itself.

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”.

### 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 25 - Alarms in Host Hardware RAID Controller**

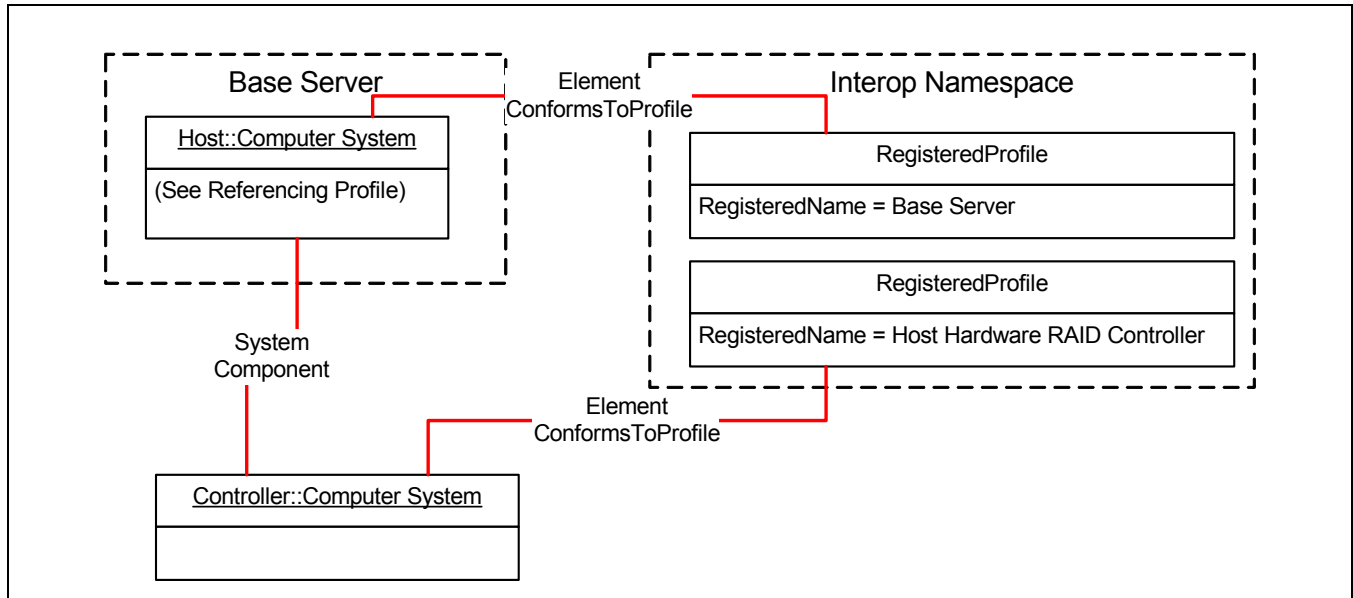
### 8.3.5 Server Profile

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 2 Common Profiles, 1.4.0 Rev 6* Clause 41: 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 26 - Profile Registration with Host Hardware RAID Controller and Base Server Profiles**

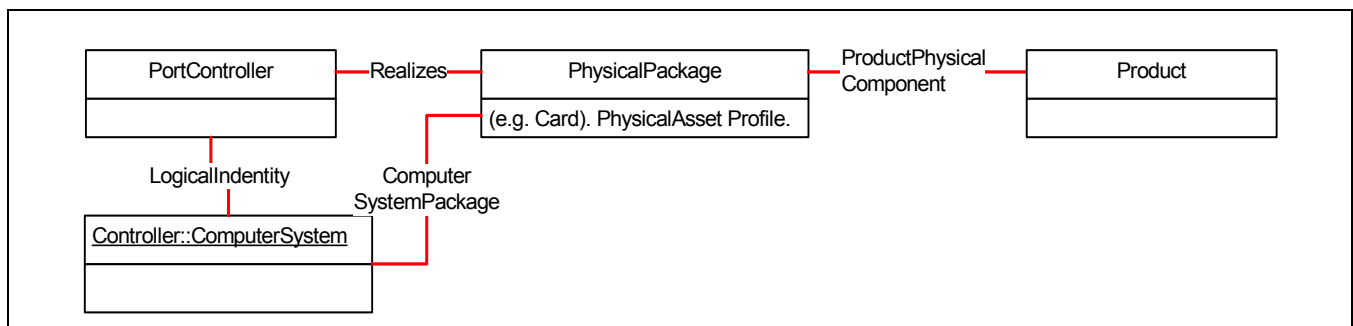
**8.3.7 Profile Discovery and Advertisement**

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 27 - 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 28 - 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.

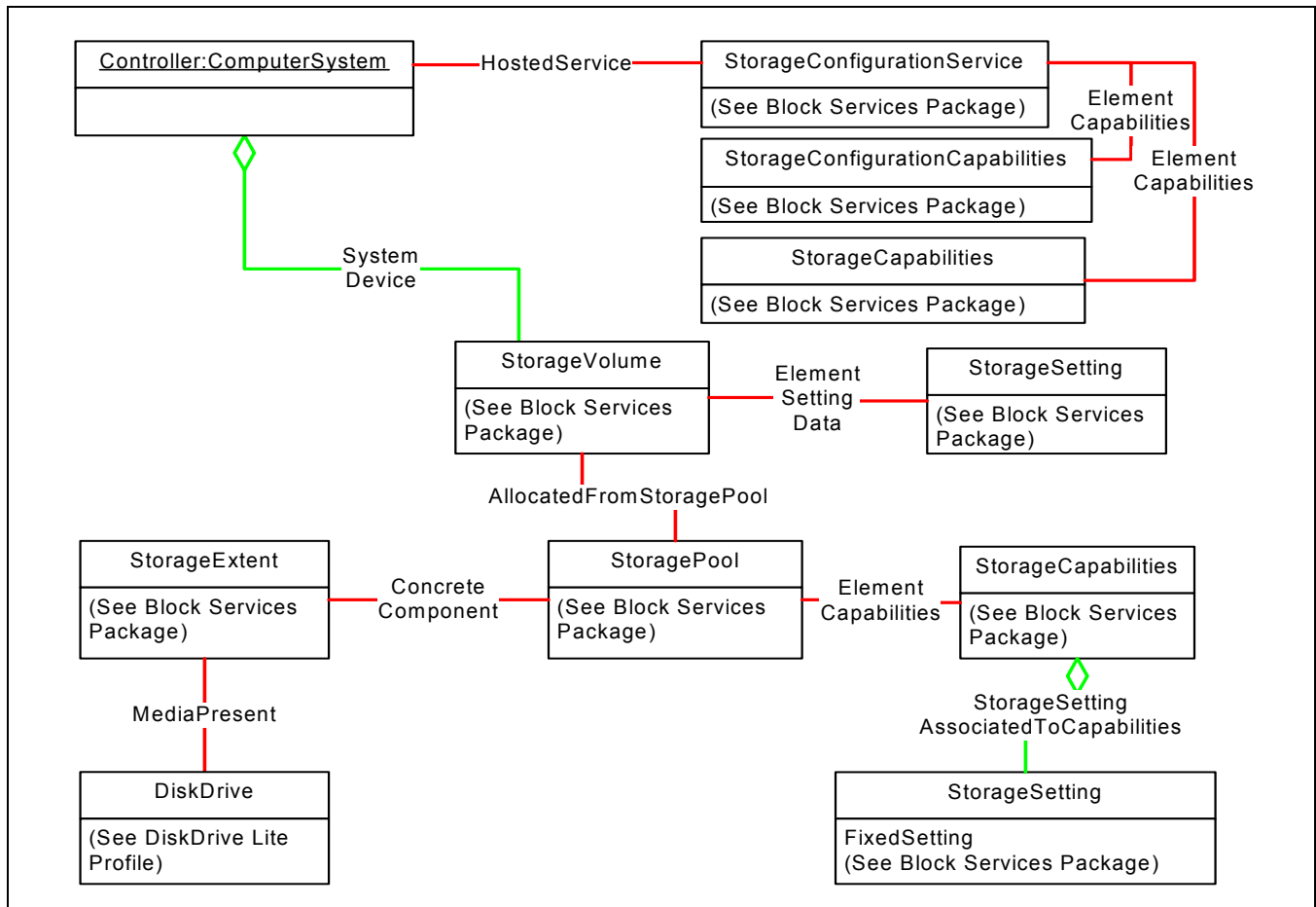


Figure 28 - Block Services Package in Host Hardware RAID Controller

#### 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 (InElements).

### 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.

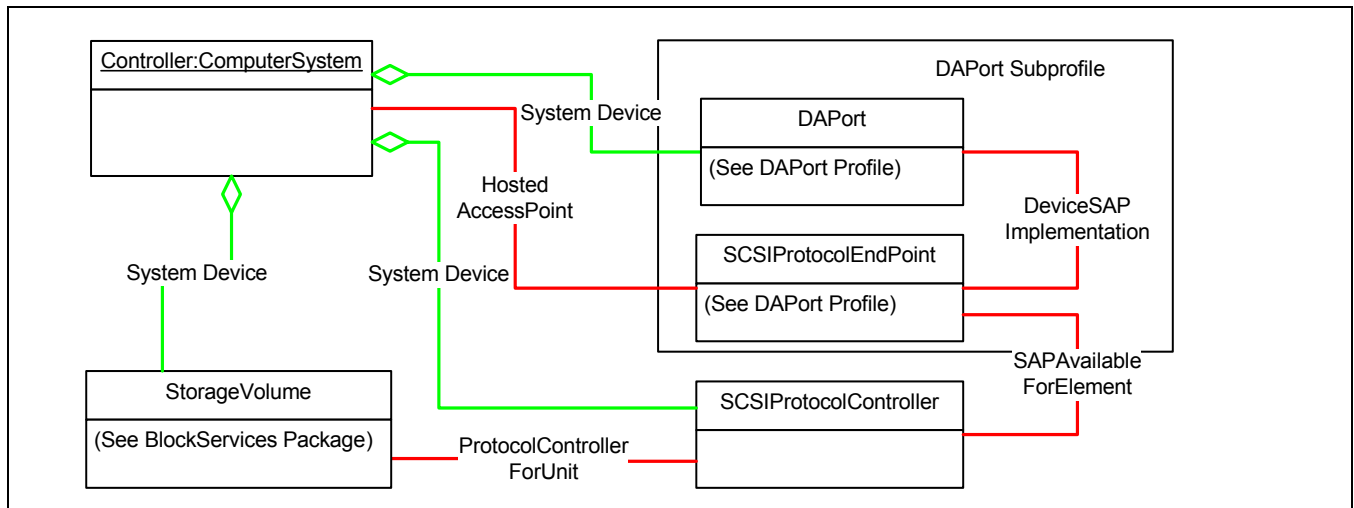
### 8.3.11 Implementation of DAPort and SCSIProtocolController

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 an 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 29, "DAPort Subprofile in Host Hardware Controller".

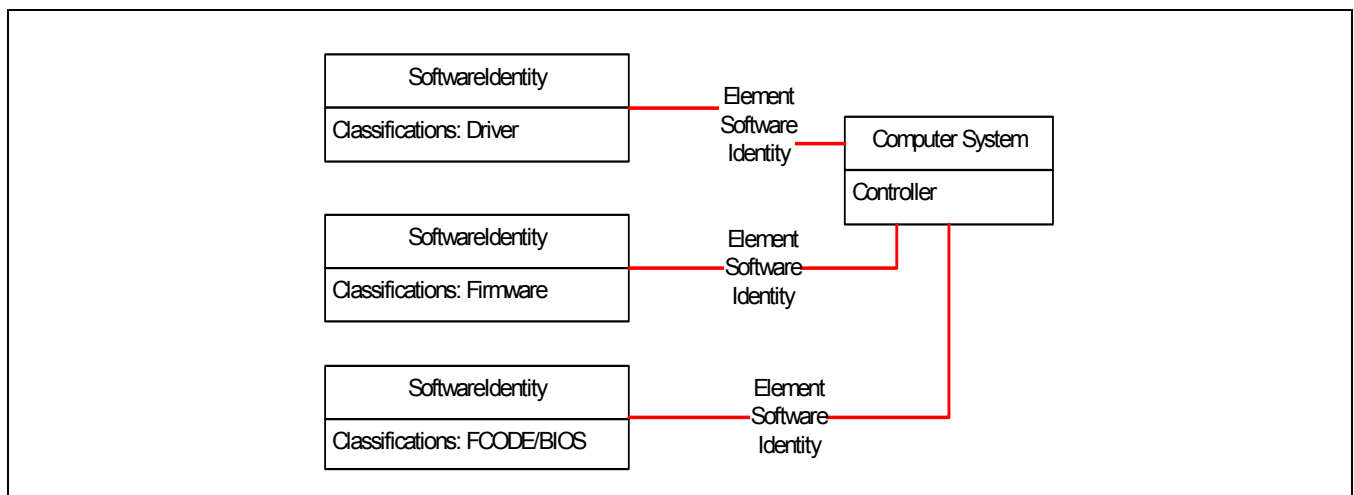


**Figure 29 - DAPort Subprofile in Host Hardware Controller**

**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.

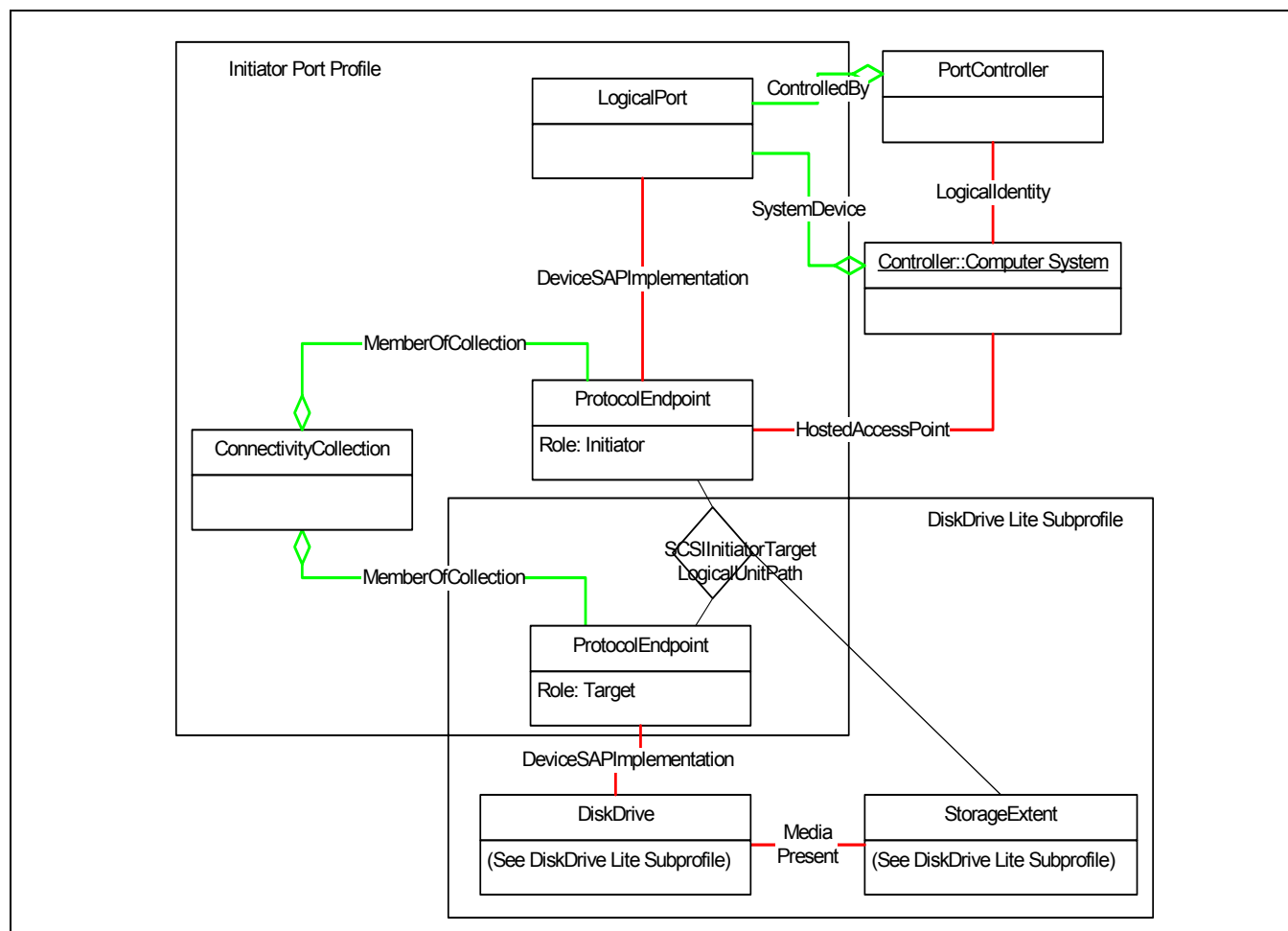
To model the driver, firmware and BIOS software for the controller, the implementation shall instantiate an instance of SoftwareIdentity class associated to the top level ComputerSystem that represents the RAID controller, using the ElementSoftwareIdentity association. The SoftwareIdentity instances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



**Figure 30 - 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 31 - Initiator Port profiles and Disk Drive Lite Subprofile**

#### 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.

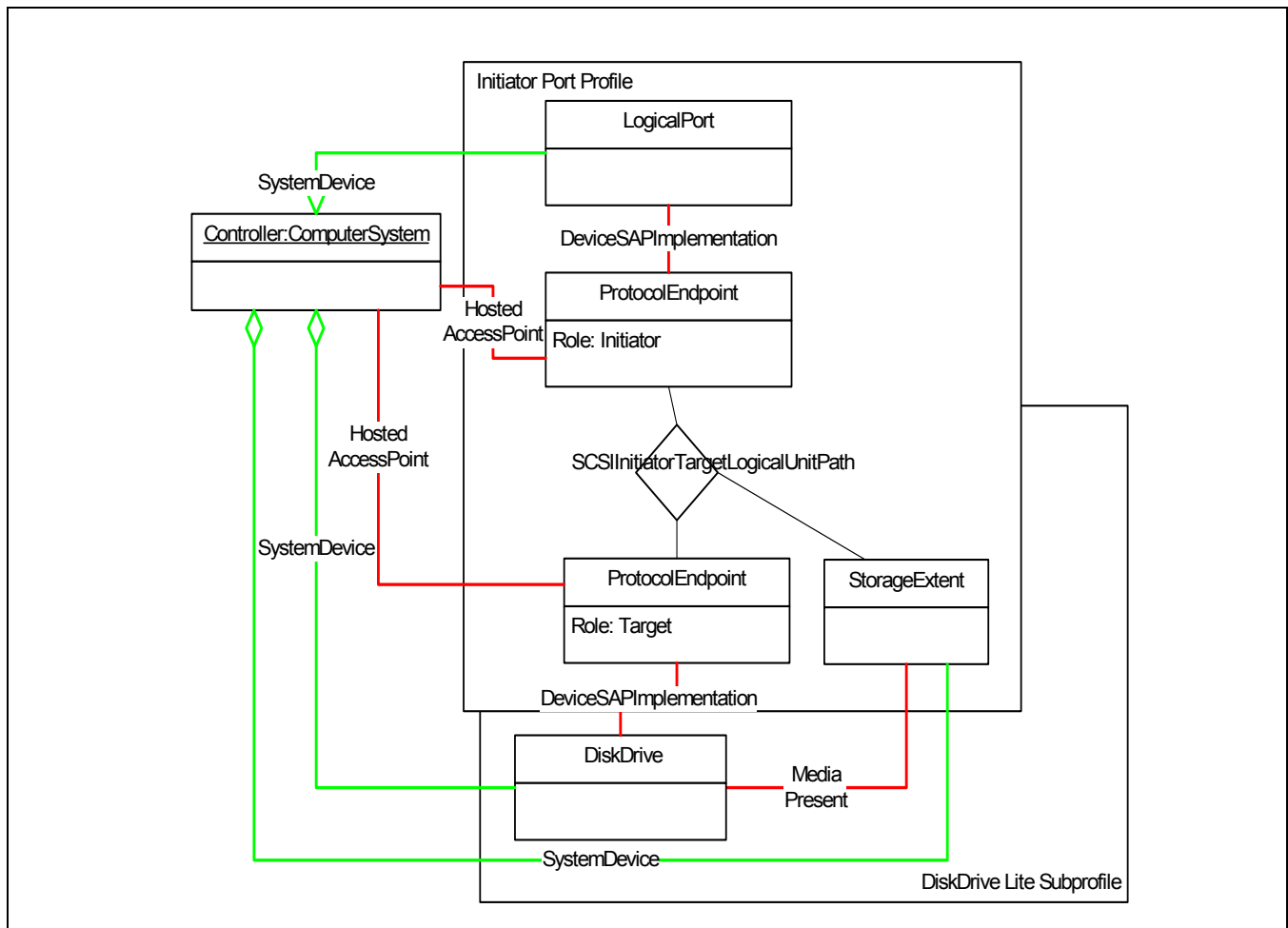
Like LogicalPort, the ProtocolEndpoint class is intended to model the storage protocol in an abstract manner. For example, the ProtocolEndpoint could represent a SCSI or ATA protocol. Thus, the instance of the ProtocolEndpoint shall be subclassed to SCSIProtocolEndpoint or ATAProtocolEndpoint.

### 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 Subprofile (see *Storage Management Technical Specification, Part 3 Block Devices, 1.4.0 Rev 6 Clause 11: Disk Drive Lite Subprofile*).



**Figure 32 - Model for Imported Disks**

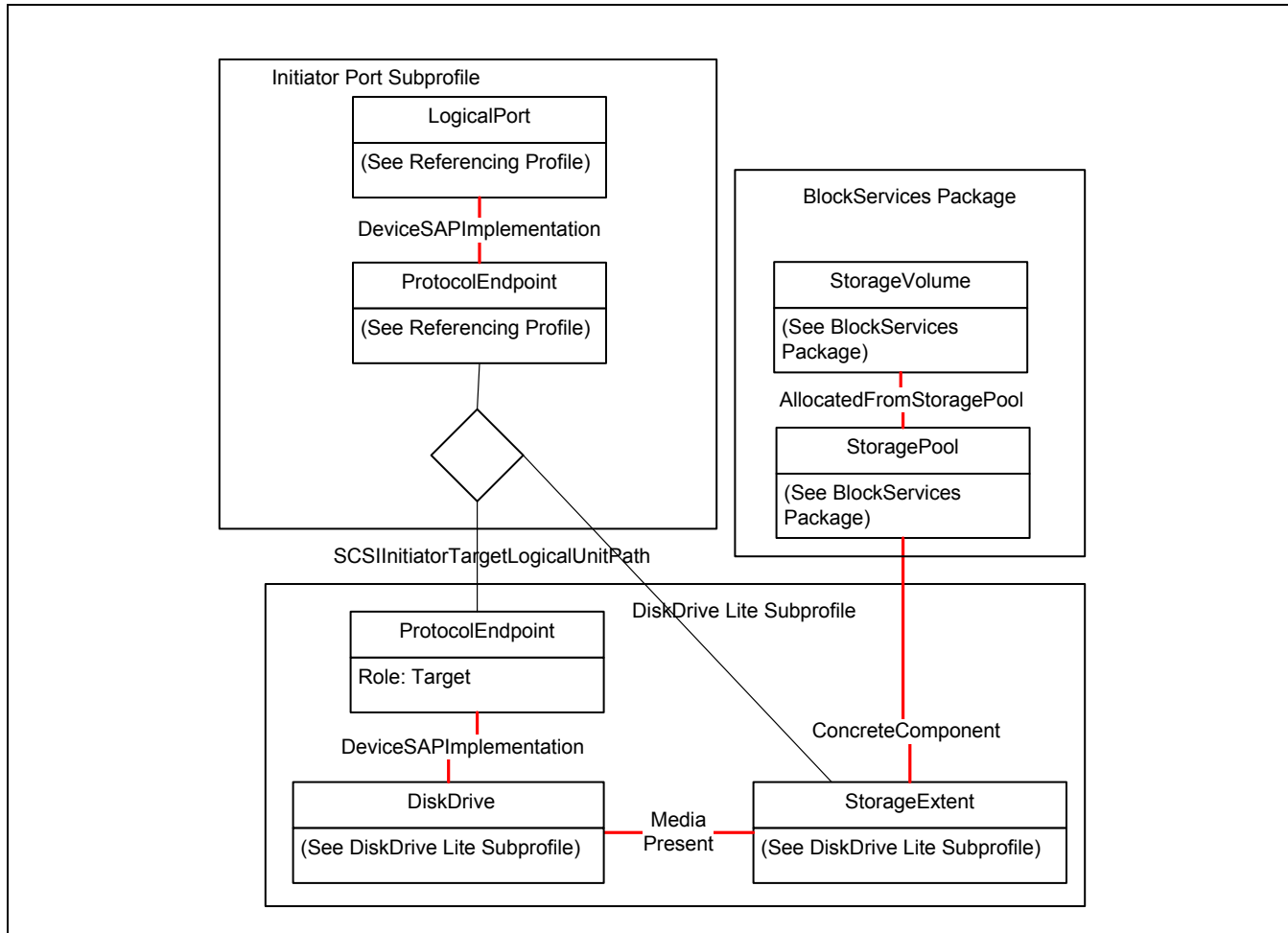
To model one disk drive:

- 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 Subprofile and instantiating instances of StorageExtent to represent the imported volumes. Figure 33, "Imported Virtual Volumes" shows this model.

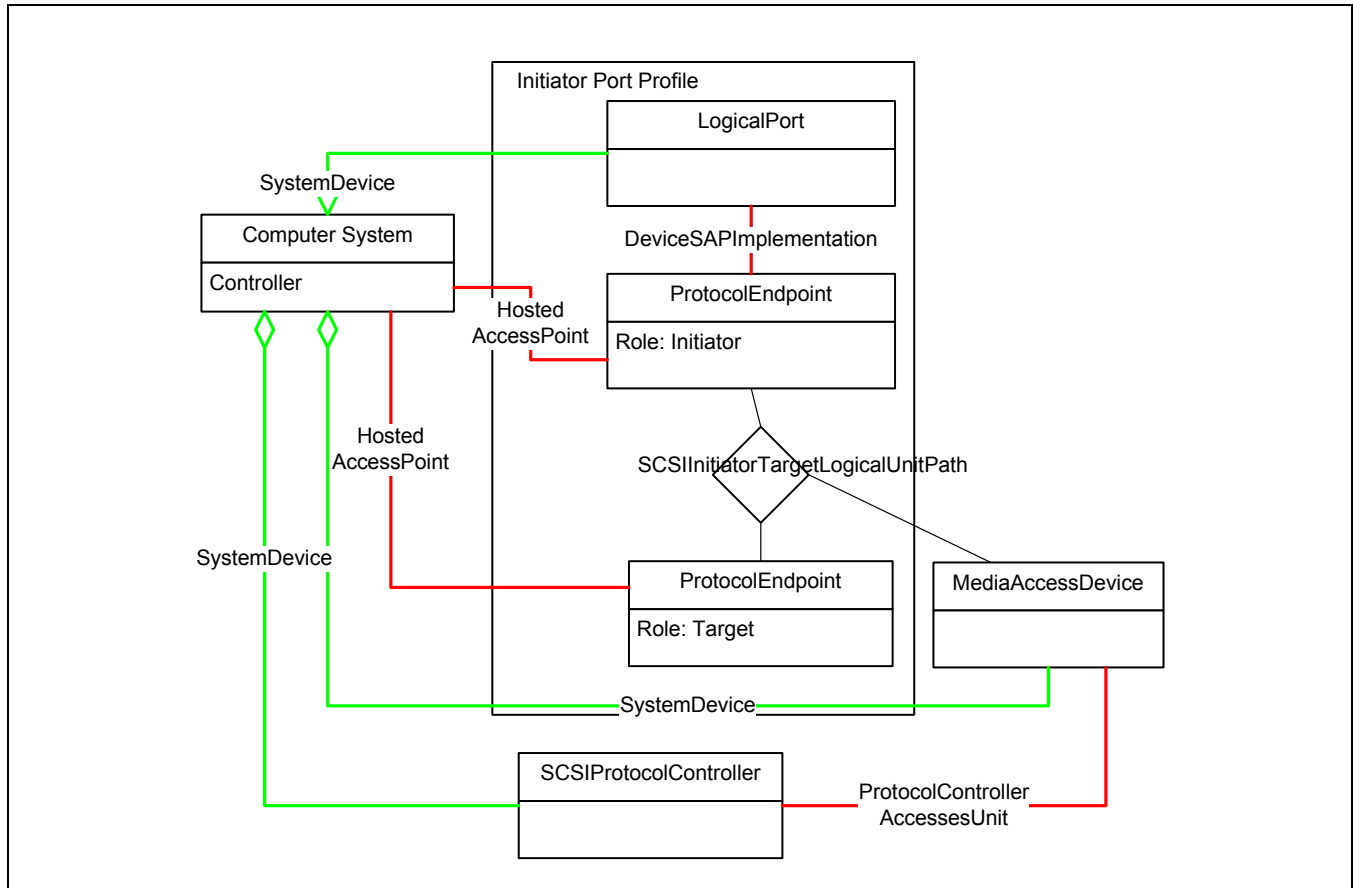


**Figure 33 - Imported Virtual Volumes**

**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 drives) 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 34, "Device "Pass Through" Example" shows this model.



**Figure 34 - 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 = null

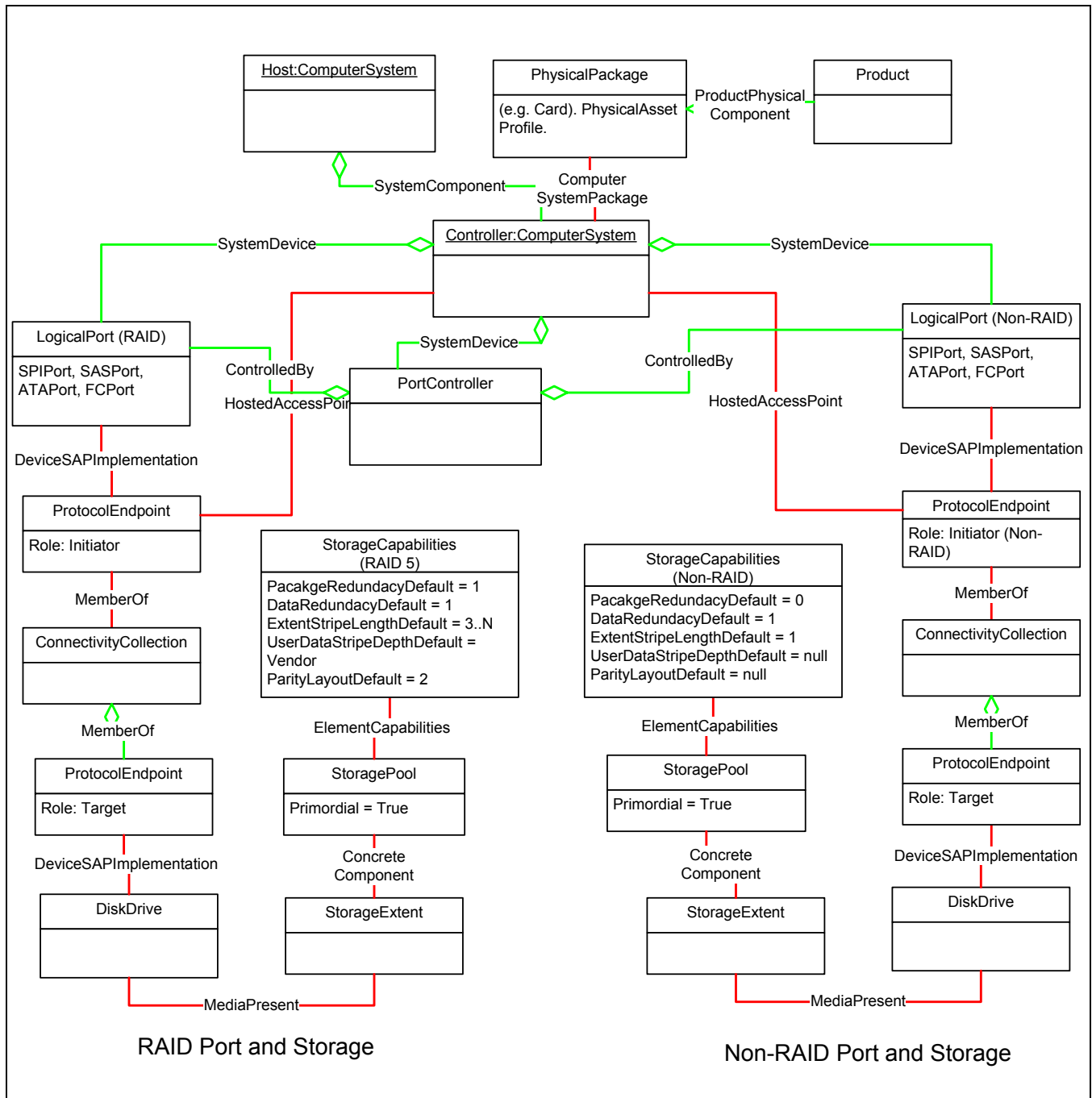


Figure 35 - Example of Mutli-Function Controllers

**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

### 8.5.1 Test an audible Alarm

```
//
// DESCRIPTION
//
// Test an audible alarm for the RAID controller.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
//
// 1. A reference to the ComputerSystem that represents the controller
//     in the Host Hardware Profile, is known as $Controller->
//     Get a list of all the ports for the RAID controller

$Alarms->[] = AssociatorNames($Controller->, // ObjectName
                             CIM_AssociatedAlarm, // AssocClass
                             CIM_AlarmDevice, // ResultClass
                             GroupComponent, // Role
```



```

PartComponent) // ResultRole

if ($Alarms->[] == null || $Alarms->[].length == 0) {
    <ERROR! No Alarms on the controller!>
}

// invoke the method to blink the alarm
for (#i in $Alarms->[] ) {
    if Alarms[i].AudibleAlarm = "true" {
        #MethodReturn = InvokeMethod( $Alarms->[0],
            SetAlarmState,
            "Alternating")
        if(#MethodReturn != 0) {
            <ERROR! SetAlarmState () method Failed >
        }
    }
}
}

```

## 8.6 CIM Elements

Table 76 describes the CIM elements for Host Hardware RAID Controller.

**Table 76 - 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.
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.
8.6.6 CIM_LogicalIdentity	Mandatory	Used to associate the ComputerSystem representing the controller with PortController.
8.6.7 CIM_MediaAccessDevice	Mandatory	Represents a tape or optical drive.
8.6.8 CIM_PortController	Mandatory	Serves as a component of the server ComputerSystem and is associated to the controller ComputerSystem.
8.6.9 CIM_Product	Mandatory	Asset information about the RAID controller.
8.6.10 CIM_ProductPhysicalComponent	Mandatory	Associates Product and PhysicalPackage.
8.6.11 CIM_ProtocolControllerForUnit (Extent or MediaAccessDevice)	Mandatory	Associates SCSIProtocolController to StorageExtent or MediaAccessDevice.

**Table 76 - CIM Elements for Host Hardware RAID Controller**

Element Name	Requirement	Description
8.6.12 CIM_ProtocolControllerForUnit (Volume)	Mandatory	Associated ProtocolController to StorageVolume.
8.6.13 CIM_Realizes (Associates PhysicalPackage to PortController)	Mandatory	Associates PortController and PhysicalPackage from the Physical Asset profile.
8.6.14 CIM_SAPAvailableForElement	Mandatory	Associates SCSIProtocolController to the DAPort ProtocolEndpoint.
8.6.15 CIM_SCSIProtocolController	Mandatory	Represents the target/device aspects of storage exported by the RAID controller.
8.6.16 CIM_SoftwareIdentity (Driver)	Mandatory	Override SoftwareIdentity from Software Inventory profile to assure Classifications property includes Driver.
8.6.17 CIM_SoftwareIdentity (FCode/BIOS)	Optional	Override SoftwareIdentity from Software Inventory profile to assure Classifications property includes FCODE/BIOS.
8.6.18 CIM_SoftwareIdentity (Firmware)	Optional	Override SoftwareIdentity from Software Inventory profile to assure Classifications property includes Firmware.
8.6.19 CIM_StorageExtent	Optional	Models an imported volume from an external array.
8.6.20 CIM_SystemComponent	Mandatory	Associates ComputerSystems representing the hosting system and the RAID controller.
8.6.21 CIM_SystemDevice (Associates System to AlarmDevice)	Optional	Associates System to AlarmDevice.
8.6.22 CIM_SystemDevice (Associates controller system to PortController)	Mandatory	Associates controller system to PortController.
8.6.23 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 77 describes class CIM\_AlarmDevice.

**Table 77 - SMI Referenced Properties/Methods for CIM\_AlarmDevice**

Properties	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 78 describes class CIM\_AssociatedAlarm.

**Table 78 - SMI Referenced Properties/Methods for CIM\_AssociatedAlarm**

Properties	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

Table 79 describes class CIM\_ComputerSystem (Host Hardware RAID Controller).

**Table 79 - SMI Referenced Properties/Methods for CIM\_ComputerSystem (Host Hardware RAID Controller)**

Properties	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	Optional	Contact a details for owner.
PrimaryOwnerName	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.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 80 describes class CIM\_ComputerSystemPackage (Associates PhysicalPackage to ComputerSystem).

**Table 80 - SMI Referenced Properties/Methods for CIM\_ComputerSystemPackage (Associates PhysicalPackage to ComputerSystem)**

Properties	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 81 describes class CIM\_ControlledBy.

**Table 81 - SMI Referenced Properties/Methods for CIM\_ControlledBy**

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

### 8.6.6 CIM\_LogicalIdentity

Associates the ComputerSystem representing the controller and the RAID controller.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 82 describes class CIM\_LogicalIdentity.

**Table 82 - SMI Referenced Properties/Methods for CIM\_LogicalIdentity**

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

### 8.6.7 CIM\_MediaAccessDevice

Represents a tape or optical drive.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 83 describes class CIM\_MediaAccessDevice.

**Table 83 - SMI Referenced Properties/Methods for CIM\_MediaAccessDevice**

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

**Table 83 - SMI Referenced Properties/Methods for CIM\_MediaAccessDevice**

Properties	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	

**8.6.8 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 84 describes class CIM\_PortController.

**Table 84 - SMI Referenced Properties/Methods for CIM\_PortController**

Properties	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.

**8.6.9 CIM\_Product**

Asset information about the RAID controller.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 85 describes class CIM\_Product.

**Table 85 - SMI Referenced Properties/Methods for CIM\_Product**

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

**Table 85 - SMI Referenced Properties/Methods for CIM\_Product**

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

**8.6.10 CIM\_ProductPhysicalComponent**

Associates Product and PhysicalPackage.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 86 describes class CIM\_ProductPhysicalComponent.

**Table 86 - SMI Referenced Properties/Methods for CIM\_ProductPhysicalComponent**

Properties	Requirement	Description & Notes
GroupComponent	Mandatory	
PartComponent	Mandatory	

**8.6.11 CIM\_ProtocolControllerForUnit (Extent or MediaAccessDevice)**

Associates SCSIProtocolController to StorageExtent or MediaAccessDevice.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 87 describes class CIM\_ProtocolControllerForUnit (Extent or MediaAccessDevice).

**Table 87 - SMI Referenced Properties/Methods for CIM\_ProtocolControllerForUnit (Extent or MediaAccessDevice)**

Properties	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.

**8.6.12 CIM\_ProtocolControllerForUnit (Volume)**

Associated ProtocolController to StorageVolume.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 88 describes class CIM\_ProtocolControllerForUnit (Volume).

**Table 88 - SMI Referenced Properties/Methods for CIM\_ProtocolControllerForUnit (Volume)**

Properties	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.

**8.6.13 CIM\_Realizes (Associates PhysicalPackage to PortController)**

Associates PortController and PhysicalPackage from the Physical Asset profile.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 89 describes class CIM\_Realizes (Associates PhysicalPackage to PortController).

**Table 89 - SMI Referenced Properties/Methods for CIM\_Realizes (Associates PhysicalPackage to PortController)**

Properties	Requirement	Description & Notes
Dependent	Mandatory	
Antecedent	Mandatory	

**8.6.14 CIM\_SAPAvailableForElement**

Associates SCSIProtocolController to the DAPort ProtocolEndpoint.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory



Table 90 describes class CIM\_SAPAvailableForElement.

**Table 90 - SMI Referenced Properties/Methods for CIM\_SAPAvailableForElement**

Properties	Requirement	Description & Notes
ManagedElement	Mandatory	
AvailableSAP	Mandatory	

### 8.6.15 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 91 describes class CIM\_SCSIProtocolController.

**Table 91 - SMI Referenced Properties/Methods for CIM\_SCSIProtocolController**

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

### 8.6.16 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 92 describes class CIM\_SoftwareIdentity (Driver).

**Table 92 - SMI Referenced Properties/Methods for CIM\_SoftwareIdentity (Driver)**

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

### 8.6.17 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 93 describes class CIM\_SoftwareIdentity (FCode/BIOS).

**Table 93 - SMI Referenced Properties/Methods for CIM\_SoftwareIdentity (FCode/BIOS)**

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

#### 8.6.18 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 94 describes class CIM\_SoftwareIdentity (Firmware).

**Table 94 - SMI Referenced Properties/Methods for CIM\_SoftwareIdentity (Firmware)**

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

#### 8.6.19 CIM\_StorageExtent

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

Table 95 describes class CIM\_StorageExtent.

**Table 95 - SMI Referenced Properties/Methods for CIM\_StorageExtent**

Properties	Requirement	Description & Notes
SystemCreationClassName	Mandatory	
SystemName	Mandatory	
CreationClassName	Mandatory	
DeviceID	Mandatory	
BlockSize	Mandatory	
NumberOfBlocks	Mandatory	The number of blocks as reported by the hardware.

**Table 95 - SMI Referenced Properties/Methods for CIM\_StorageExtent**

Properties	Requirement	Description & Notes
ConsumableBlocks	Mandatory	The number of usable blocks.
Primordial	Mandatory	Shall be true.
OperationalStatus	Mandatory	

**8.6.20 CIM\_SystemComponent**

Associates ComputerSystems representing the hosting system and the RAID controller.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 96 describes class CIM\_SystemComponent.

**Table 96 - SMI Referenced Properties/Methods for CIM\_SystemComponent**

Properties	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.

**8.6.21 CIM\_SystemDevice (Associates System to AlarmDevice)**

Associates System to AlarmDevice.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 97 describes class CIM\_SystemDevice (Associates System to AlarmDevice).

**Table 97 - SMI Referenced Properties/Methods for CIM\_SystemDevice (Associates System to AlarmDevice)**

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

**8.6.22 CIM\_SystemDevice (Associates controller system to PortController)**

Associates controller system to PortController.

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

Table 98 describes class CIM\_SystemDevice (Associates controller system to PortController).

**Table 98 - SMI Referenced Properties/Methods for CIM\_SystemDevice (Associates controller system to PortController)**

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

### 8.6.23 CIM\_SystemDevice (System to SCSIProtocolController)

Links SCSIProtocolController to the controller system.

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

Table 99 describes class CIM\_SystemDevice (System to SCSIProtocolController).

**Table 99 - SMI Referenced Properties/Methods for CIM\_SystemDevice (System to SCSIProtocolController)**

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

## **EXPERIMENTAL**

---



---

---



---

**EXPERIMENTAL**
**Clause 9: iSCSI Initiator Profile**
**9.1 Description**

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 spans SCSI and network concepts and introduces new terms. The following table is a summary of some key iSCSI terms, their equivalent CIM classes, and definitions (from the IETF iSCSI RFC).

**Table 100 - iSCSI Terminology**

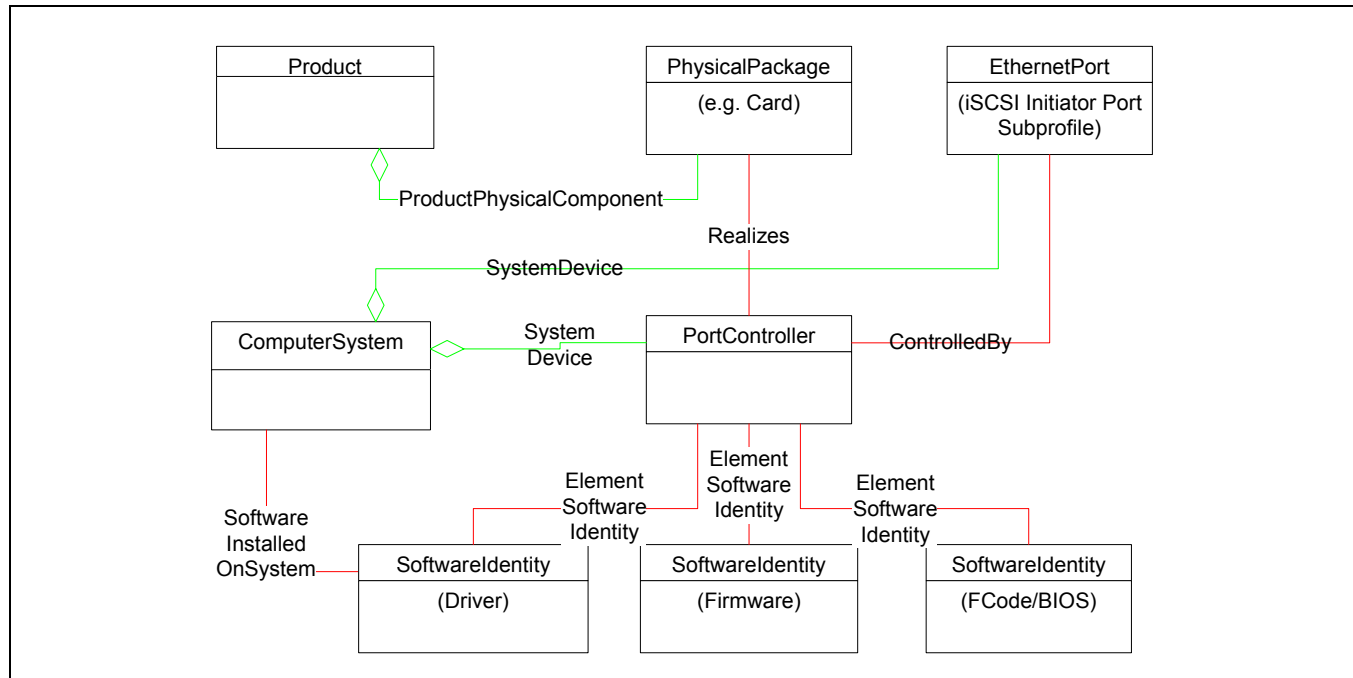
iSCSI Term	CIM Class Name	Notes
Network Entity	ComputerSystem	The Network Entity represents a device or gateway that is accessible from the IP network. A Network Entity shall have one or more Network Portals, each of which can be used to gain access to the IP network by some iSCSI Nodes contained in 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.
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 Subprofile [see *Storage Management Technical Specification, Part 2 Common Profiles, 1.4.0 Rev 6*, Clause 16: iSCSI Initiator Port Profile] that includes classes (EthernetPort, iSCSIProtocolEndpoint) that model SCSI ports and network portals.

Figure 36 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 36.

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

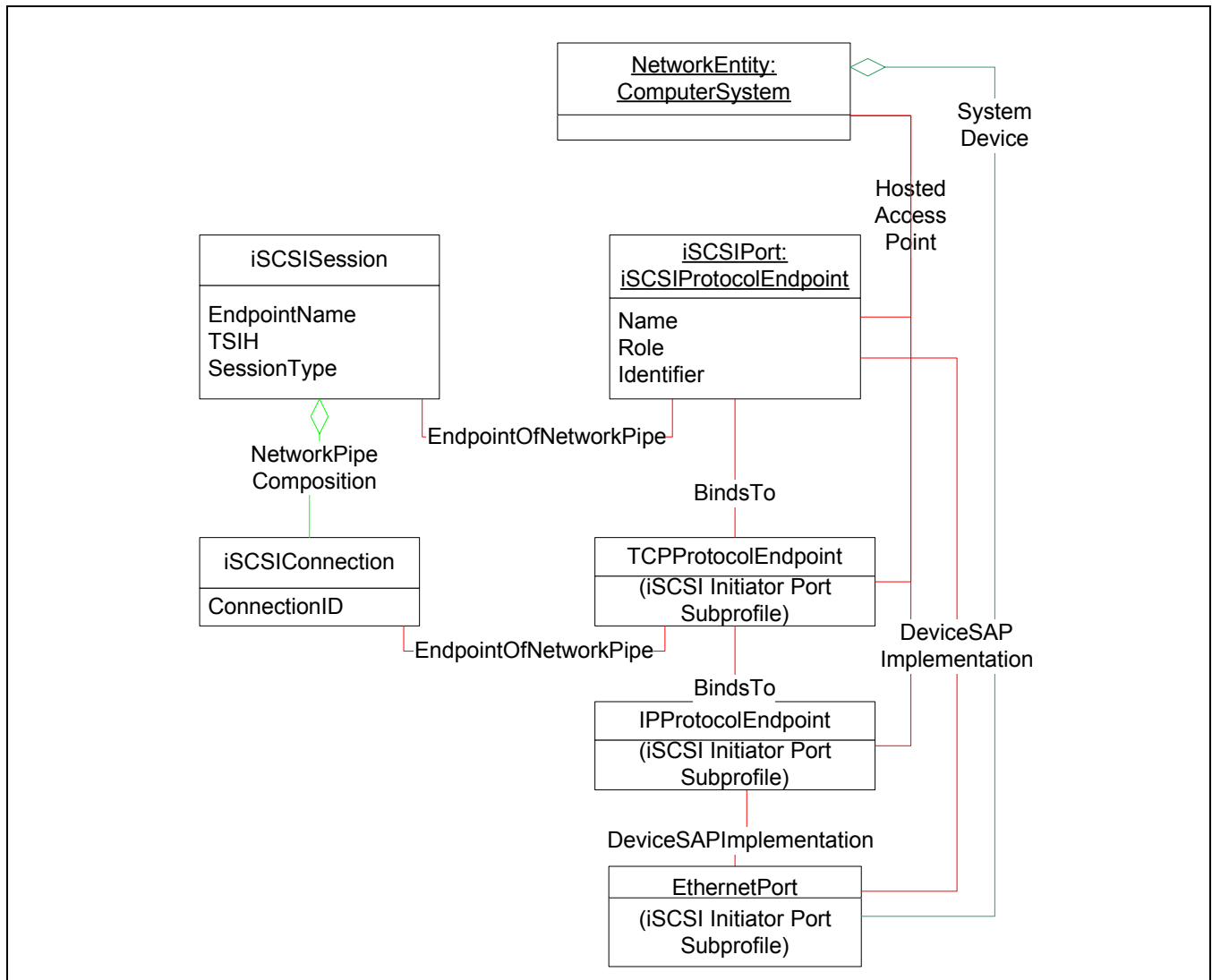


**Figure 36 - iSCSI Product and Package Model**

### 9.1.1 Sessions and Connections

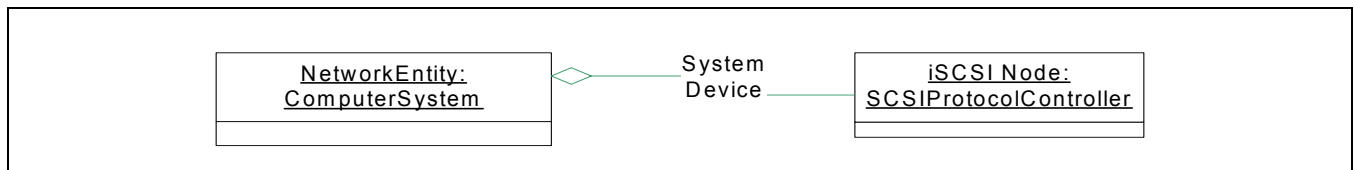
A session is an active communication stream between an iSCSI initiator port and an iSCSI target port. However, any given session may contain part or all of the TCP/IP addresses within a Portal Group. Conceptually, a Portal Group is a pool of addresses which may be used to create/receive a session.

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 37 - iSCSI Sessions and Connections Model**

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



**Figure 38 - iSCSI Initiator Node**

**9.1.2 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 2 Common Profiles, 1.4.0 Rev 6 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 2 Common Profiles, 1.4.0 Rev 6* 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 1 Common Architecture, 1.4.0 Rev 6*, 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 101 defines the possible states that shall be supported for EthernetPort.OperationalStatus. The main OperationalStatus shall be the first element in the array

**Table 101 - 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 Supported Subprofiles and Packages

Table 102 describes the supported profiles for iSCSI Initiator.

**Table 102 - Supported Profiles for iSCSI Initiator**

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

## 9.4 Methods of the Profile

None

## 9.5 Client Considerations and Recipes

### 9.5.1 Add an additional NIC port

See 9.7.7 in *Storage Management Technical Specification, Part 2 Common Profiles, 1.4.0 Rev 6*.

### 9.5.2 Find the health of an initiator

See 9.7.9 in *Storage Management Technical Specification, Part 2 Common Profiles, 1.4.0 Rev 6*.

### 9.5.3 Enable/disable header and data digest

See 9.5.3 in *Storage Management Technical Specification, Part 2 Common Profiles, 1.4.0 Rev 6*.



## 9.6 Registered Name and Version

iSCSI Initiator version 1.1.0

## 9.7 CIM Elements

Table 103 describes the CIM elements for iSCSI Initiator.

**Table 103 - CIM Elements for iSCSI Initiator**

Element Name	Requirement	Description
9.7.1 CIM_BindsTo (TCPProtocolEndpoint to IPProtocolEndpoint)	Mandatory	
9.7.2 CIM_BindsTo (iSCSIProtocolEndpoint to TCPProtocolEndpoint)	Mandatory	
9.7.3 CIM_ComputerSystem	Mandatory	
9.7.4 CIM_ControlledBy	Optional	
9.7.5 CIM_DeviceSAPImplementation (EthernetPort to IPProtocolEndpoint)	Optional	
9.7.6 CIM_DeviceSAPImplementation (EthernetPort to iSCSIProtocolEndpoint)	Optional	
9.7.7 CIM_ElementSoftwareIdentity	Mandatory	
9.7.8 CIM_EndpointOfNetworkPipe (Between iSCSI Session and iSCSIProtocolEndpoint)	Mandatory	
9.7.9 CIM_EndpointOfNetworkPipe (Between iSCSI connection and TCPProtocolEndpoint)	Mandatory	
9.7.10 CIM_InstalledSoftwareIdentity	Optional	
9.7.11 CIM_NetworkPipeComposition	Mandatory	
9.7.12 CIM_PhysicalPackage	Mandatory	
9.7.13 CIM_PortController	Optional	
9.7.14 CIM_Product	Mandatory	
9.7.15 CIM_ProductPhysicalComponent	Mandatory	
9.7.16 CIM_ProtocolControllerForPort	Mandatory	
9.7.17 CIM_Realizes	Mandatory	
9.7.18 CIM_SAPAvailableForElement	Mandatory	
9.7.19 CIM_SCSIProtocolController	Mandatory	
9.7.20 CIM_SoftwareIdentity	Optional	
9.7.21 CIM_SystemDevice (to EthernetPort)	Mandatory	
9.7.22 CIM_SystemDevice (to PortController)	Mandatory	

**Table 103 - CIM Elements for iSCSI Initiator**

Element Name	Requirement	Description
9.7.23 CIM_SystemDevice (to ProtocolController)	Mandatory	
9.7.24 CIM_iSCSIConnection	Optional	
9.7.25 CIM_iSCSI_Session	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.

**9.7.1 CIM\_BindsTo (TCPProtocolEndpoint to IPProtocolEndpoint)**

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 104 describes class CIM\_BindsTo (TCPProtocolEndpoint to IPProtocolEndpoint).

**Table 104 - SMI Referenced Properties/Methods for CIM\_BindsTo (TCPProtocolEndpoint to IPProtocolEndpoint)**

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

**9.7.2 CIM\_BindsTo (iSCSIProtocolEndpoint to TCPProtocolEndpoint)**

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 105 describes class CIM\_BindsTo (iSCSIProtocolEndpoint to TCPProtocolEndpoint).

**Table 105 - SMI Referenced Properties/Methods for CIM\_BindsTo (iSCSIProtocolEndpoint to TCP-ProtocolEndpoint)**

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

### 9.7.3 CIM\_ComputerSystem

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 106 describes class CIM\_ComputerSystem.

**Table 106 - 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.
ElementName		Mandatory	
NameFormat		Mandatory	
OtherIdentifyingInfo	C	Mandatory	
OperationalStatus		Mandatory	
Dedicated		Mandatory	Shall be "Not Dedicated".
OtherDedicatedDescriptions		Optional	

### 9.7.4 CIM\_ControlledBy

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 107 describes class CIM\_ControlledBy.

**Table 107 - SMI Referenced Properties/Methods for CIM\_ControlledBy**

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

### 9.7.5 CIM\_DeviceSAPImplementation (EthernetPort to IPProtocolEndpoint)

Created By: Static

Modified By: Static

Deleted By: Static  
Requirement: Optional

Table 108 describes class CIM\_DeviceSAPImplementation (EthernetPort to IPProtocolEndpoint).

**Table 108 - SMI Referenced Properties/Methods for CIM\_DeviceSAPImplementation (EthernetPort to IPProtocolEndpoint)**

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

#### 9.7.6 CIM\_DeviceSAPImplementation (EthernetPort to iSCSIProtocolEndpoint)

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

Table 109 describes class CIM\_DeviceSAPImplementation (EthernetPort to iSCSIProtocolEndpoint).

**Table 109 - SMI Referenced Properties/Methods for CIM\_DeviceSAPImplementation (EthernetPort to iSCSIProtocolEndpoint)**

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

#### 9.7.7 CIM\_ElementSoftwareIdentity

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

Table 110 describes class CIM\_ElementSoftwareIdentity.

**Table 110 - SMI Referenced Properties/Methods for CIM\_ElementSoftwareIdentity**

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

### 9.7.8 CIM\_EndpointOfNetworkPipe (Between iSCSI Session and iSCSIProtocolEndpoint)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 111 describes class CIM\_EndpointOfNetworkPipe (Between iSCSI Session and iSCSIProtocolEndpoint).

**Table 111 - SMI Referenced Properties/Methods for CIM\_EndpointOfNetworkPipe (Between iSCSI Session and iSCSIProtocolEndpoint)**

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

### 9.7.9 CIM\_EndpointOfNetworkPipe (Between iSCSI connection and TCPProtocolEndpoint)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 112 describes class CIM\_EndpointOfNetworkPipe (Between iSCSI connection and TCPProtocolEndpoint).

**Table 112 - SMI Referenced Properties/Methods for CIM\_EndpointOfNetworkPipe (Between iSCSI connection and TCPProtocolEndpoint)**

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

### 9.7.10 CIM\_InstalledSoftwareIdentity

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 113 describes class CIM\_InstalledSoftwareIdentity.

**Table 113 - SMI Referenced Properties/Methods for CIM\_InstalledSoftwareIdentity**

Properties	Flags	Requirement	Description & Notes
InstalledSoftware		Mandatory	
System		Mandatory	

### 9.7.11 CIM\_NetworkPipeComposition

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 114 describes class CIM\_NetworkPipeComposition.

**Table 114 - SMI Referenced Properties/Methods for CIM\_NetworkPipeComposition**

Properties	Flags	Requirement	Description & Notes
PartComponent		Mandatory	
GroupComponent		Mandatory	

### 9.7.12 CIM\_PhysicalPackage

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 115 describes class CIM\_PhysicalPackage.

**Table 115 - 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.

### 9.7.13 CIM\_PortController

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 116 describes class CIM\_PortController.

**Table 116 - SMI Referenced Properties/Methods for CIM\_PortController**

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

#### 9.7.14 CIM\_Product

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 117 describes class CIM\_Product.

**Table 117 - 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.

#### 9.7.15 CIM\_ProductPhysicalComponent

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 118 describes class CIM\_ProductPhysicalComponent.

**Table 118 - SMI Referenced Properties/Methods for CIM\_ProductPhysicalComponent**

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	
PartComponent		Mandatory	

#### 9.7.16 CIM\_ProtocolControllerForPort

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 119 describes class CIM\_ProtocolControllerForPort.

**Table 119 - SMI Referenced Properties/Methods for CIM\_ProtocolControllerForPort**

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

#### 9.7.17 CIM\_Realizes

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 120 describes class CIM\_Realizes.

**Table 120 - SMI Referenced Properties/Methods for CIM\_Realizes**

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

#### 9.7.18 CIM\_SAPAvailableForElement

Created By: Static

Modified By: Static

Deleted By: Static



Requirement: Mandatory

Table 121 describes class CIM\_SAPAvailableForElement.

**Table 121 - SMI Referenced Properties/Methods for CIM\_SAPAvailableForElement**

Properties	Flags	Requirement	Description & Notes
AvailableSAP		Mandatory	
ManagedElement		Mandatory	

### 9.7.19 CIM\_SCSIProtocolController

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 122 describes class CIM\_SCSIProtocolController.

**Table 122 - SMI Referenced Properties/Methods for CIM\_SCSIProtocolController**

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

### 9.7.20 CIM\_SoftwareIdentity

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 123 describes class CIM\_SoftwareIdentity.

**Table 123 - 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).

#### 9.7.21 CIM\_SystemDevice (to EthernetPort)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 124 describes class CIM\_SystemDevice (to EthernetPort).

**Table 124 - SMI Referenced Properties/Methods for CIM\_SystemDevice (to EthernetPort)**

Properties	Flags	Requirement	Description & Notes
PartComponent		Mandatory	
GroupComponent		Mandatory	

#### 9.7.22 CIM\_SystemDevice (to PortController)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 125 describes class CIM\_SystemDevice (to PortController).

**Table 125 - SMI Referenced Properties/Methods for CIM\_SystemDevice (to PortController)**

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	
PartComponent		Mandatory	

### 9.7.23 CIM\_SystemDevice (to ProtocolController)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 126 describes class CIM\_SystemDevice (to ProtocolController).

**Table 126 - SMI Referenced Properties/Methods for CIM\_SystemDevice (to ProtocolController)**

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	
PartComponent		Mandatory	

### 9.7.24 CIM\_iSCSIConnection

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 127 describes class CIM\_iSCSIConnection.

**Table 127 - 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	
OtherHeaderDigestMethod		Optional	
DataDigestMethod		Mandatory	
OtherDataDigestMethod		Optional	
ReceivingMarkers		Mandatory	
SendingMarkers		Mandatory	

**Table 127 - SMI Referenced Properties/Methods for CIM\_iSCSIConnection**

Properties	Flags	Requirement	Description & Notes
ActiveiSCSIVersion		Mandatory	
AuthenticationMethodUsed		Mandatory	Maps to IMA_GetInUseInitiatorAuthMethods.
MutualAuthentication		Mandatory	

**9.7.25 CIM\_iSCSISession**

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 128 describes class CIM\_iSCSISession.

**Table 128 - 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.

**Table 128 - SMI Referenced Properties/Methods for CIM\_iSCSISession**

Properties	Flags	Requirement	Description & Notes
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**

**Clause 10: SCSI Multipath Management Subprofile**

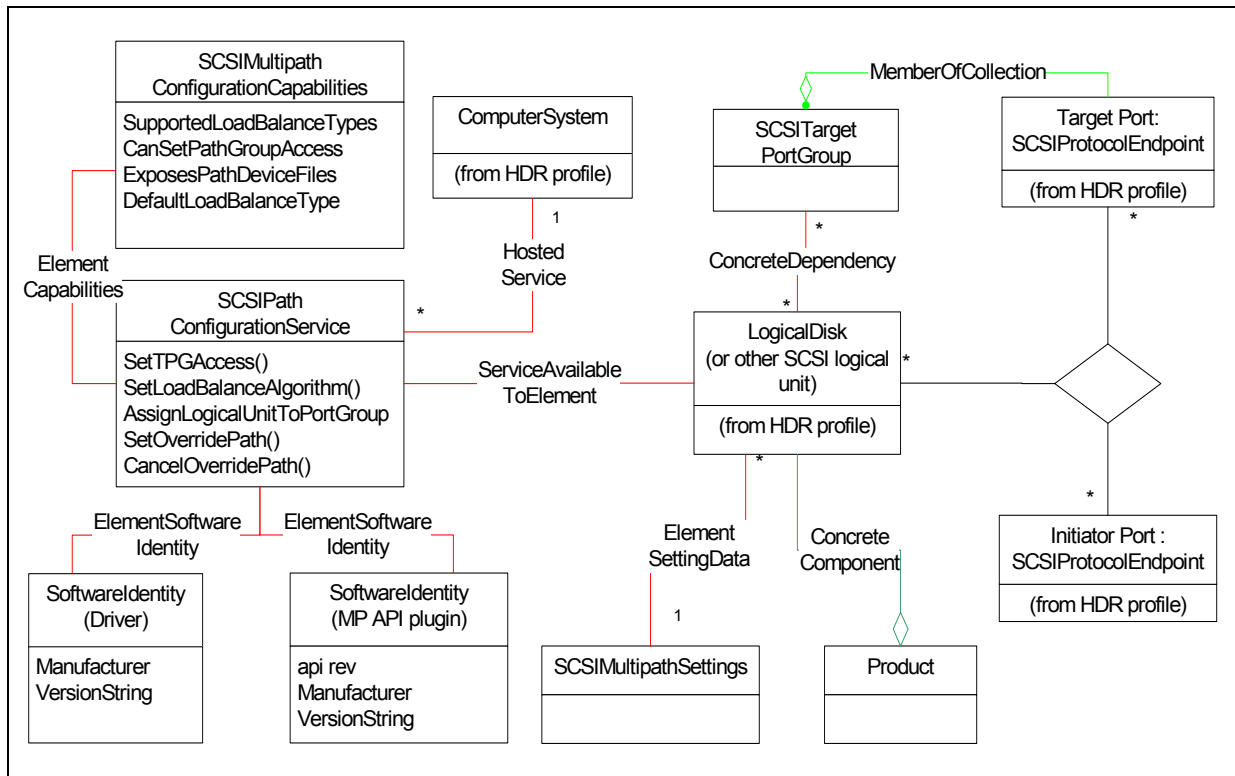
**10.1 Description**

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 7.1.1.1.

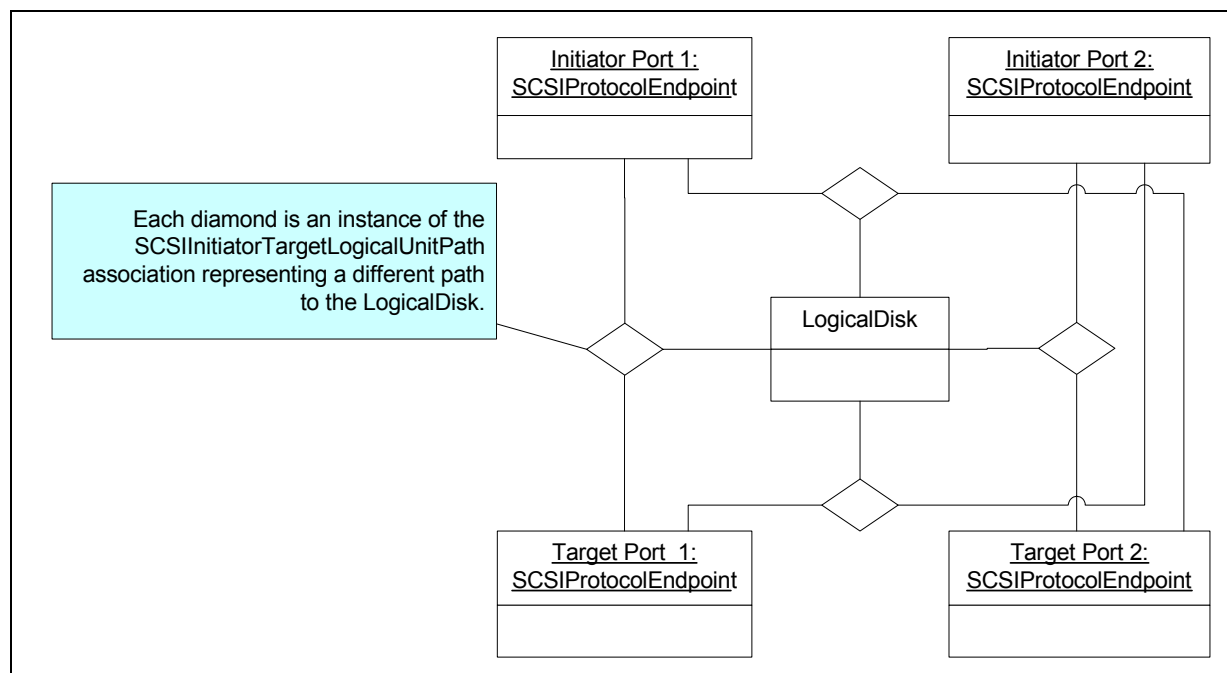
The rest of the examples in this section use LogicalDisks since multipath disk arrays are more common, but the same approach can be extended to other storage types. For example, a TapeDrive can model multipath access to a tape drive.

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 39.



**Figure 39 - Multipath Management Class Diagram**

.Figure 40 shows the relationship of target and initiator ports (SCSIProtocolEndpoint instances) and a disk



**Figure 40 - 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.

### 10.1.1 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. Multipath 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 Subprofile 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 subprofile 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 subprofiles.

## 10.3 Cascading Considerations

None.

## 10.4 Supported Subprofiles and Packages

None

## 10.5 Methods of the Profile

All methods are part of `SCSIPathConfigurationService` and are optional.

### 10.5.1 `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.

### 10.5.2 `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`.

### 10.5.3 `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 SCSI Multipath Configuration Capabilities 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.

#### 10.5.4 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 SCSI Initiator Target Logical Unit Path.

#### 10.5.5 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 SCSI Initiator Target Logical Unit Path.

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

## 10.6 Client Considerations and Recipes

### 10.6.1 Discover All Paths to a Disk Volume

```
// DESCRIPTION
//
// This recipe discovers the topology of HW resources attached to the
// current host system. Host controllers (HBAs) and attached volumes
// (disks) supporting SCSI protocol are reported.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
//
// 1. A reference to the top-level ComputerSystem in the HDR Profile
//    is known as $Host->
//
// Step 1. Get name(s) of the SCSIProtocolEndpoints representing
// SCSI initiators on the host system.
//
$SPEs->[] = AssociatorNames($Host->, // ObjectName
    "CIM_HostedAccessPoint", // AssocClass
    "CIM_SCSIProtocolEndpoint", // ResultClass
    "Antecedent", // Role
    "Dependent") // ResultRole

$Initiators->[] = <get the subset of $SPEs with Role = "Initiator">

if ($Initiators->[] == null || $Initiators->[].length == 0) {
```

```

    <EXIT: No SCSI Initiators on the host system!>
}

// Determine the topology of inititors, targets, and volumes.
//
for (#i in $Initiators->[]) {
    // Step 2. Find the paths attached to each initiator
    // SCSIProtocolEndpoint. Each path includes a REF to
    // a target SCSIProtocolEndpoint and to a logical unit.
    $Paths[] = References(
        $Initiators->[#i], // ObjectName
        "CIM_SCSIInitiatorTargetLogicalUnitPath", // ResultClass
        "Initiator", // Role
        false, // IncludeQualifiers
        false, // IncludeClassOrigin
        {"LogicalUnitNumber"}) // PropertyList
    // All members of Paths[] have the same Initiator REF.
    // Sort the paths so that all members with identical
    // Target REFs are consecutive.
    $SortedPaths->[] = <Paths[] sorted by Target property>
    // Step 3. Find all the logical units attached to an
    // initiator/target pair and verify that each has
    // a unique logical unit number.
    #l = 0; // the index of LU numbers to test
    $CurrentTarget = <initialize to null>
    for (#p in $SortedPaths->[]) {
        // Each time a new target REF is discovered, save it
        // in $CurrentTarget and empty the list if LU numbers.
        if ($CurrentTarget != $SortedPaths[#p]->Target) {
            $CurrentTarget = $SortedPaths[#p]->Target
            #LUNumbers[] = {}; // empty the list
        }
        if contains($SortedPath->LogicalUnitNumber, #LUNumbers[]) {
            <ERROR: logical unit number already in use>
        } else {
            LUNumbers[#l++] = $SortedPath->LogicalUnitNumber
        }
        // Other interesting bits of info available
        // $SortedPaths->State
        // $SortedPaths->LogicalUnit is a REF to the
        // LogicalDevice subclass (LogicalDisk, TapeDrive)
        // where Name is a logical unit correlatable ID
        // $CurrentTarget is a ref to the target
        // SCSIProtocolController where ConnectionType
        // is the transport and Name is the transport-
        // specific correlatable ID (e.g. PortWWN)
        // $Initiators->[#i] is a ref to the initiator
    }
}

```

```

        // SCSIProtocolController
    }
}

```

### 10.6.2 Force Failover or change Load Balancing on a volume

```

//
// DESCRIPTION:
// Set the desired path for a multipath disk volume
//
// Preconditions:
// $Host - Reference to the hosting system
// $Path - ref to SCSIInitiatorTargetLogicalUnitPath instance - desired path
//
/ Notes:
// If the volume is asymmetric, failover applies. If symmetric, then
// we use a driver override to set the path.
//
// $Vol = LU REF from $Path
$Vol = $Path->LogicalUnit
// Get SCSIPathConfigurationService instances associated
// to $Vol via ServiceAvailableToElement- ERROR if not exactly 1
$Services->[] = AssociatorNames($Vol, // this is a ref
    "CIM_ServiceAvailableToElement",
    "CIM_SCSIPathConfigurationService",
    null,null)
if ($Services == null || $Services->[].size != 1) {
    <ERROR: must not be more than 1
        CIM_SCSIPathConfigurationService
            associated with an LogicalDevice/volume>
}

// Get SCSIMultipathConfigurationCapabilities instances
// associated to $Service - Error if not exactly one
$Capabilities->[] = AssociatorNames($Services->[0],// ObjectName
    "CIM_ElementCapabilities",// AssocClass
    "CIM_SCSIMultipathConfigurationCapabilities",// ResultClass
    "ManagedElement", // Role
    "Capabilities") // ResultRole
if ($Capabilities == null || $Capabilities[].length != 1) {
    <ERROR: must be 1 CIM_SCSIMultipathConfigurationCapabilities instance
        associated with each SCSIPathConfigurationService>
}

// Look at CIM_SCSIMultipathSettings.Asymmetric to determine
// whether the Volume is Asymmetric MP. If no SCSIMultipathSettings
// is associated to the volume, or if Assymmetric property is
// not-present/null, then assume Symmetric

```

```

$SettingDatas-[] = AssociatorNames($Vol,
    "CIM_ElementSettingData",
    "CIM_SCSIMultipathSettings",
    "null,null)
If ($SettingDatas == null || $SettingDatas[].length != 1 ||
    $SettingDatas->[0].Asymmetric == null ||
    $SettingDatas->[0].Asymmetric == false) {
    // A Symmetric MP volume has multiple, active paths.
    // Use SetOverridePath to make just one path active
    if ($Capabilities->[0].CanOverridePaths == false) {
        <EXIT: Instrumentation does not support OverridePaths method>
    }
    // set up and invoke the method
    %InArguments["Path"]=$Path
    #MethodReturn = InvokeMethod(
        $Services->[0],
        "SetOverridePath",
        %InArguments,
        %OutArguments)
    if(#MethodReturn != 0) {
        <ERROR! SetOverridePath method Failed >
    }
} else {
    // The Volume has Assymmetric MP access
    if (Capabilities->[0].CanSetTPGAccess == false) {
        <EXIT: Instrumentation does not support SetTPGAccess method>
    }
    // Find the TargetPortGroups containing $Vol
    $TPGs->[] = AssociatorNames($Vol,
        "CIM_ConcreteDependency",
        "CIM_SCSITargetPortGroup",
        "Dependent", "Antecedent")
    // Some of these TPGs may not include the Target Port in $Path,
    // locate one the does.
    #foundTPG = false
    for i in $TPGs->[] {
        $TargetPorts->[] = AssociatorNames($TPGs->[#i],
            "CIM_MemberOfCollection",
            "CIM_LogicalPort",
            "Collection","Member")
        if contains($Path->Target, $TargetPorts) {
            $TheTPG = $TPGs->[#i]
            #foundTPG = false
            break
        }
    }
    %LogicalUnit["LogicalUnit"] = $Vol
}

```

```

%InArguments["TargetPortGroups"] = {$TheTPG}
%InArguments["AccessStates"] = {"6"} // Active
#MethodReturn = InvokeMethod(
    $Services->[0],
    "SetTPGAccess",
    %InArguments,
    %OutArguments)
if(#MethodReturn != 0) {
    <ERROR! SetSetTPGAccess method Failed >
}
// To be completely accurate, we should include SetOverridePath
// method call here; in theory a TPG can support multiple ports.
// But in practice, Asymmetric arrays have one port per TPG.
}

```

### 10.6.3 Change a LogicalDisk's LoadBalance Algorithm

```

//
// DESCRIPTION:
// Set the load balance algorithm for a multipath disk volume
//
// Preconditions:
// $Host - Reference to the hosting system
// $Vol - Reference to the volume
//
// Notes:
// The currentload balance type could be a driver-wide default (from
// SCSIMultipathConfigurationCapabilities), or a per-LU value from
// SCSIMultipathSettings associated with $Vol.
// Once we get the current value, we search the list of supported values for
// a different supported value, then use it to call SetLoadBalanceAlgorithm
//
// get SCSIPathConfigurationService instances associated to $Vol
// via ServiceAvailableToElement- ERROR if not exactly 1
$Services->[] = AssociatorNames($Vol, // this is a ref
    "CIM_ServiceAvailableToElement",
    "CIM_SCSIPathConfigurationService",
    null,null)
if ($Services == null || $Services->[].size != 1) {
    <ERROR: must not be more than 1
    CIM_SCSIPathConfigurationService
    associated with an LogicalDevice/volume>
}

// 3. Set $Capabilities to the instance SCSIMultipathConfigurationCapabilities
// associated to $Service - Error if not exactly one
$Capabilities->[] = AssociatorNames($Services->[0],// ObjectName
    "CIM_ElementCapabilities",// AssocClass
    "CIM_SCSIMultipathConfigurationCapabilities",// ResultClass

```

```

    "ManagedElement", // Role
    "Capabilities") // ResultRole
if ($Capabilities == null || $Capabilities[].length != 1) {
    <ERROR: must be 1 CIM_SCSIMultipathConfigurationCapabilities instance
        associated with each SCSIPathConfigurationService>
}
// the next two tests are not required, but will help diagnostics
if $Capabilities->[0].OnlySupportsSpecifiedProducts == true {
    <EXIT: the multipath instrumentation only supports 1 devices-specific
        load balance algorithm which cannot be changed>
}
if sizeof($Capabilities->[0].SupportedLoadBalanceTypes) == 1 {
    <EXIT: the multipath instrumentation only supports 1
        load balance algorithm which cannot be changed>
}

// Get the CIM_SCSIMultipathSettings instance associated with $Vol
$SettingDatas->[] = AssociatorNames($Vol,
    "CIM_ElementSettingData",
    "CIM_SCSIMultipathSettings",
    "null,null")
if ($SettingDatas != null || $SettingDatas[].length > 1) {
    <ERROR: must be 0 or 1 CIM_SCSIMultipathSettings instance
        associated with each SCSI logical unit>
}

// Determine the current load balance type
// The default from Capabilities applies unless overridden
#MyLoadBalanceType = $Capabilities->[0].DefaultLoadBalanceType
// If SettingData is associated to $Vol, it may override load balance
if (($SettingDatas != null || $SettingDatas[].length == 1)
    // SettingData load balance 7 means use Capabilities default
    && $SettingDatas->[0].CurrentLoadBalanceType != "7") {
    // Override with value from settings
    #MyLoadBalanceType = $SettingDatas->[0].CurrentLoadBalanceType
}

#newTypeFound = false
for i in $Capabilities->[0].SupportedLoadBalanceTypes {
    if $Capabilities->[0].SupportedLoadBalanceTypes[#i] != "0" // Unknown
        && $Capabilities->[0].SupportedLoadBalanceTypes[#i] != #MyLoadBalanceType {
        // We found a supported load balance type other than the
        // current one. We can live with "No Load Balance" (2),
        // but just for kicks, try to find another one
        #newTypeFound = true
        %InArguments["LoadBalanceAlgorithm" =
            $Capabilities->[0].SupportedLoadBalanceTypes[#i]

```

```

if $Capabilities->[0].SupportedLoadBalanceTypes[#i] == "1" { // Other
    %InArguments["OtherLoadBalanceAlgorithmDescription"] =
        $Capabilities->[0].OtherSupportedLoadBalanceTypes[#i]
}
if $Capabilities->[0].SupportedLoadBalanceTypes[#i] != "2" { // no LB
    break
}
}
}
if #newTypeFound == false {
    <EXIT: no supported load balance types found other than the current type>
}

// invoke the SetLoadBalanceAlgorithm method
%InArguments["LogicalDevice"] = $Vol
#MethodReturn = InvokeMethod(
    $Services->[0],
    "SetLoadBalanceAlgorithm",
    %InArguments,
    %OutArguments)
if(#MethodReturn != 0) {
    <ERROR! SetLoadBalanceAlgorithm method Failed >
}

```

## 10.7 Registered Name and Version

SCSI Multipath Management version 1.3.0

## 10.8 CIM Elements

Table 129 describes the CIM elements for SCSI Multipath Management.

**Table 129 - CIM Elements for SCSI Multipath Management**

Element Name	Requirement	Description
10.8.1 CIM_ConcreteComponent	Mandatory	Associates Product and LogicalDevice subclass instances representing SCSI logical units.
10.8.2 CIM_ConcreteDependency	Mandatory	Associates SCSITargetPortGroup to LogicalDevice subclass instances representing SCSI logical units.
10.8.3 CIM_ElementCapabilities	Mandatory	Associates SCSIMultipathConfigurationCapabilities and SCSIPathConfigurationService.
10.8.4 CIM_ElementSettingData	Mandatory	Associates SCSIMultipathSettings and LogicalDevice subclass instances representing SCSI logical units.



**Table 129 - CIM Elements for SCSI Multipath Management**

Element Name	Requirement	Description
10.8.5 CIM_ElementSoftwareIdentity (Driver)	Mandatory	Associates SCSIPathConfigurationService and the Driver SoftwareIdentity instance.
10.8.6 CIM_ElementSoftwareIdentity (MP API Plugin)	Mandatory	Associates SCSIPathConfigurationService and the MP API Plugin SoftwareIdentity instance.
10.8.7 CIM_HostedService	Mandatory	Associates SCSIPathConfigurationService and the ComputerSystem from Host Discovered Resources.
10.8.8 CIM_MemberOfCollection	Mandatory	Associates SCSITargetPortGroup and SCSIProtocolEndpoint.
10.8.9 CIM_Product	Mandatory	Models a Product as defined in MP API.
10.8.10 CIM_SCSIMultipathConfigurationCapabilities	Mandatory	A class derived from Capabilities that models the capabilities of a multipath driver.
10.8.11 CIM_SCSIMultipathSettings	Mandatory	Settings related to management of multiple paths to SCSI devices.
10.8.12 CIM_SCSIPathConfigurationService	Mandatory	A class providing methods related to management of multiple paths to SCSI devices.
10.8.13 CIM_SCSITargetPortGroup	Mandatory	Models SCSI Target Port Groups.
10.8.14 CIM_ServiceAvailableToElement	Mandatory	Associates SCSIPathConfigurationService with instances representing SCSI logical units.
10.8.15 CIM_SoftwareIdentity (Driver)	Mandatory	Driver.
10.8.16 CIM_SoftwareIdentity (MP API Plugin)	Mandatory	MP API Plugin.

**10.8.1 CIM\_ConcreteComponent**

Associates Product and LogicalDevice subclass instances representing SCSI logical units.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 130 describes class CIM\_ConcreteComponent.

**Table 130 - 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.

**10.8.2 CIM\_ConcreteDependency**

Associates SCSITargetPortGroup to LogicalDevice subclass instances representing SCSI logical units.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 131 describes class CIM\_ConcreteDependency.

**Table 131 - 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.

**10.8.3 CIM\_ElementCapabilities**

Associates SCSIMultipathConfigurationCapabilities and SCSIPathConfigurationService.

Created By: External

Modified By: External

Deleted By: External

Requirement: Mandatory

Table 132 describes class CIM\_ElementCapabilities.

**Table 132 - SMI Referenced Properties/Methods for CIM\_ElementCapabilities**

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

**10.8.4 CIM\_ElementSettingData**

Associates SCSIMultipathSettings and LogicalDevice subclass instances representing SCSI logical units.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 133 describes class CIM\_ElementSettingData.

**Table 133 - 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.

#### 10.8.5 CIM\_ElementSoftwareIdentity (Driver)

Associates SCSIPathConfigurationService and the Driver SoftwareIdentity instance.

Created By: External

Modified By: External

Deleted By: External

Requirement: Mandatory

Table 134 describes class CIM\_ElementSoftwareIdentity (Driver).

**Table 134 - 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.

#### 10.8.6 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 135 describes class CIM\_ElementSoftwareIdentity (MP API Plugin).

**Table 135 - 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.

#### 10.8.7 CIM\_HostedService

Associates SCSIPathConfigurationService and the ComputerSystem from Host Discovered Resources.

Created By: External

Modified By: External

Deleted By: External  
Requirement: Mandatory

Table 136 describes class CIM\_HostedService.

**Table 136 - SMI Referenced Properties/Methods for CIM\_HostedService**

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	Reference to ComputerSystem in Host Discovered Resources.
Dependent		Mandatory	Reference to SCSIPathConfigurationService.

### 10.8.8 CIM\_MemberOfCollection

Associates SCISITargetPortGroup and SCSIProtocolEndpoint representing a target port.

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

Table 137 describes class CIM\_MemberOfCollection.

**Table 137 - SMI Referenced Properties/Methods for CIM\_MemberOfCollection**

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

### 10.8.9 CIM\_Product

Models a Product as defined in MP API.

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

Table 138 describes class CIM\_Product.

**Table 138 - SMI Referenced Properties/Methods for CIM\_Product**

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

**Table 138 - SMI Referenced Properties/Methods for CIM\_Product**

Properties	Flags	Requirement	Description & Notes
Vendor		Mandatory	
Version		Mandatory	

**10.8.10 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 139 describes class CIM\_SCSIMultipathConfigurationCapabilities.

**Table 139 - 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	

**10.8.11 CIM\_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.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 140 describes class CIM\_SCSIMultipathSettings.

**Table 140 - 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	

#### 10.8.12 CIM\_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 141 describes class CIM\_SCSIPathConfigurationService.

**Table 141 - 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.

### 10.8.13 CIM\_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 142 describes class CIM\_SCSITargetPortGroup.

**Table 142 - 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	Indicates 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.

### 10.8.14 CIM\_ServiceAvailableToElement

Associates SCSIPathConfigurationService with instances representing SCSI logical units.

Created By: External

Modified By: External

Deleted By: External

Requirement: Mandatory

Table 143 describes class CIM\_ServiceAvailableToElement.

**Table 143 - 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.

**10.8.15 CIM\_SoftwareIdentity (Driver)**

SoftwareIdentity representing the Driver software.

Created By: External

Modified By: External

Deleted By: External

Requirement: Mandatory

Table 144 describes class CIM\_SoftwareIdentity (Driver).

**Table 144 - 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).

**10.8.16 CIM\_SoftwareIdentity (MP API Plugin)**

SoftwareIdentity representing the MP API plugin software.

Created By: External

Modified By: External

Deleted By: External

Requirement: Mandatory

Table 145 describes class CIM\_SoftwareIdentity (MP API Plugin).

**Table 145 - 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**



---

---

## EXPERIMENTAL

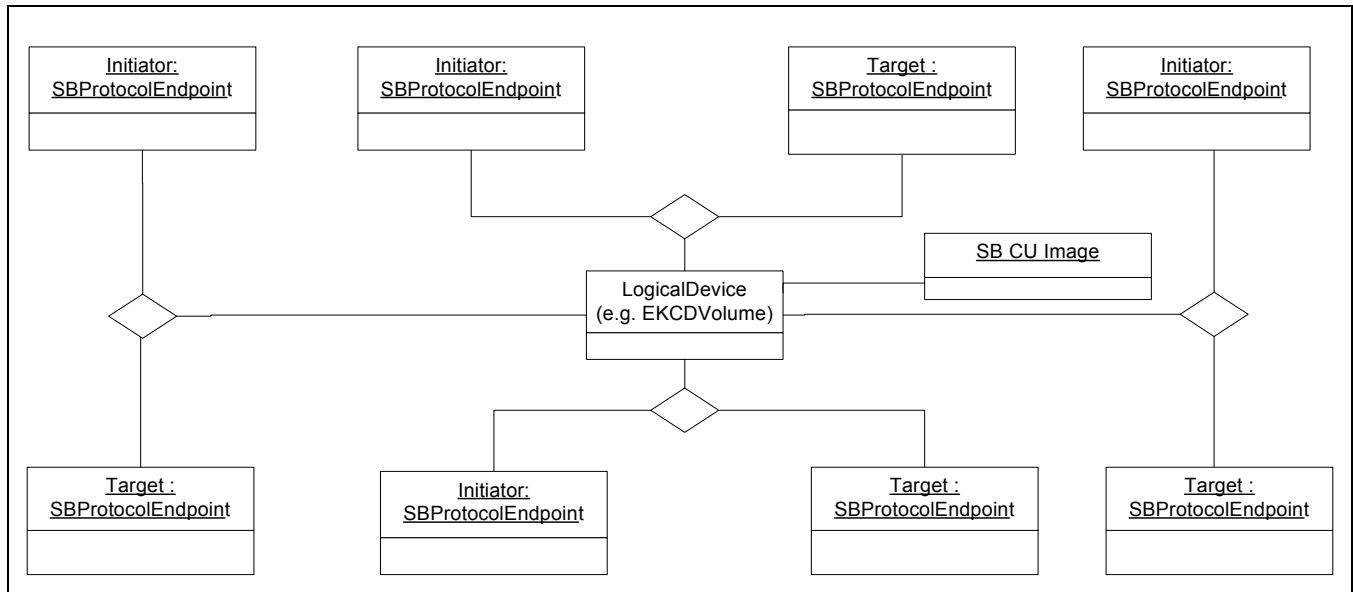
### Clause 11: SB Multipath Management Profile

#### 11.1 Description

Multipath access is inherent for SB (Single Byte) devices (see FC-SB-3).

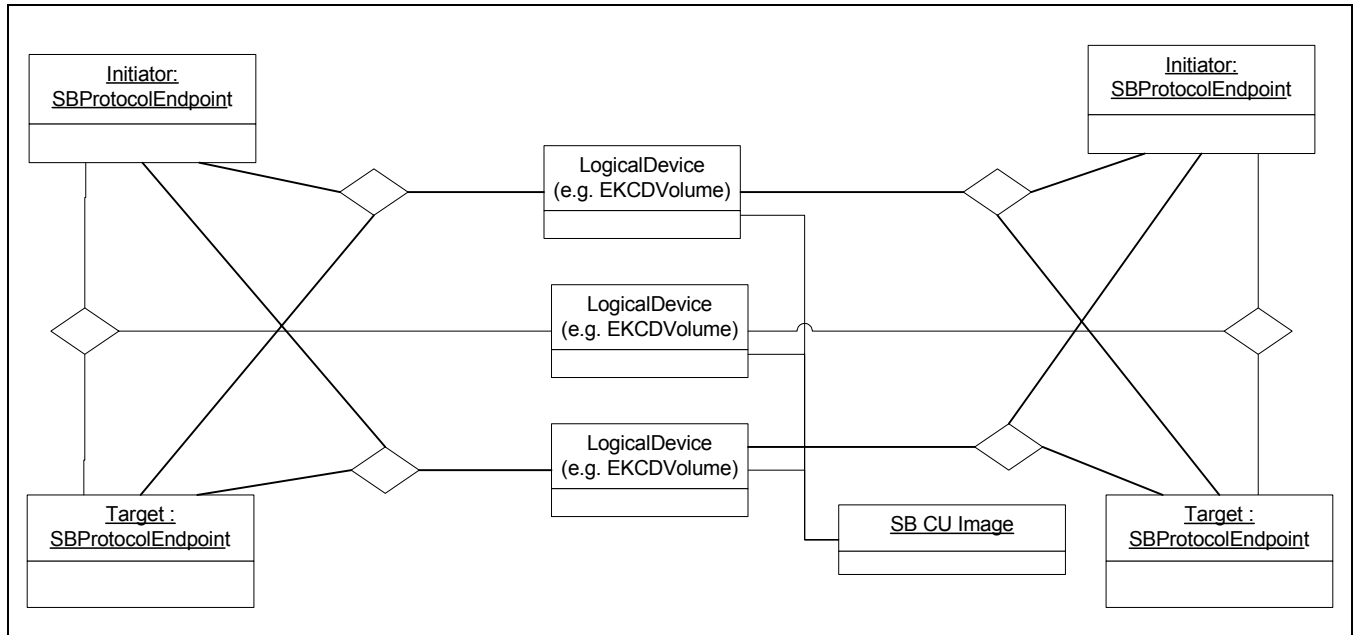
Unlike SCSI/FCP, SB only supports the appearance of a logical device once on a single channel image, i.e., one initiator can access a device only through one target port. Thus the maximum number of paths to a logical device from a single OS is equal to the number of SB channel images defined to that device from that OS's execution environment. For SB channels on zSeries systems, multipathing is a function shared between the operating system and the channel subsystem. The channel subsystem performs the final path selection from the set of paths permitted by the operating system. However, there currently are no external management interfaces provided by either the OS or the channel subsystem for managing path selection. There are mechanisms to vary individual paths to a device online or offline, as well as to configure an entire channel image online or offline; if a path to a device is taken offline or an entire channel image is configured offline, that "route" won't be used to access the device.

Figure 41 shows the relationship of target and initiator ports.



**Figure 41 - Four SB Channel Instance Diagram**

Figure 42 shows the relationship of multiple initiator ports to multiple target ports with multiple devices in a single control unit image.



**Figure 42 - Two SB Channel, Three Volume Instance Diagram**

## 11.2 Health and Fault Management Consideration

Not defined in the standard.

## 11.3 Cascading Considerations

Not defined in the standard.

## 11.4 Supported Profiles, Subprofiles, and Packages

Not defined in the standard.

## 11.5 Methods

### 11.5.1 Extrinsic Methods of this Profile

Not defined in the standard.

### 11.5.2 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

## 11.6 Client Considerations and Recipes

Not defined in the standard.

## 11.7 Registered Name and Version

SB Multipath Management version 1.0.0

## 11.8 CIM Elements

Table 146 describes the CIM elements for SB Multipath Management.

**Table 146 - CIM Elements for SB Multipath Management**

Element Name	Requirement	Description
11.8.1 SNIA_SBInitiatorTargetLogicalUnitPath	Mandatory	Associates initiator and target ProtocolEndpoints to a logical unit (LogicalDevice).
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_InitiatorTargetLogicalUnitPath	Mandatory	Path creation.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_InitiatorTargetLogicalUnitPath	Mandatory	Path deletion.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_InitiatorTargetLogicalUnitPath AND SourceInstance.State <> PreviousInstance.State	Mandatory	Path State change.

### 11.8.1 SNIA\_SBInitiatorTargetLogicalUnitPath

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 147 describes class SNIA\_SBInitiatorTargetLogicalUnitPath.

**Table 147 - SMI Referenced Properties/Methods for SNIA\_SBInitiatorTargetLogicalUnitPath**

Properties	Flags	Requirement	Description & Notes
LogicalUnit		Mandatory	A reference to a LogicalDevice.
Initiator		Mandatory	A reference to the initiator SBProtocolEndpoint.
Target		Mandatory	A reference to the target SBProtocolEndpoint.
OSDeviceName		Optional	The name of the logical unit as seen by the operating system.
UsePreferredPath		Optional	Boolean indicating whether preferred path processing is required.
PreferredPath		Optional	Boolean indicating whether this is a preferred path.
PathGroupState		Optional	One of 'Unknown', 'Path grouping not supported', 'Reset', 'Grouped', or 'Ungrouped'.
PathGroupMode		Optional	One of 'Unknown', 'None', 'Single path', 'Multipath'. Single path and Multipath only valid if PathGroupState is 4 (Grouped).
PathGroupID		Optional	String containing the ID from the OS, only valid if PathGroupState is 4 (Grouped).

## **EXPERIMENTAL**

---