Storage Management Technical Specification, Part 6 Host Elements

Version 1.3.0, Rev 6

SNIA Technical Position

21 April, 2009

Revision History

Revison 1

Date

4 January, 2007

SCRs Incorporated and other changes

Generic FC Controller Profile

- Updated for Virtual HBA work (Host-SMIS-SCR00024)

Physical FC HBA Profile

- Added this Profile (as a proposed change to FC HBA profile) (Host-SMIS-SCR00024)

Virtual FC HBA Profile

- Added this new Profile (Host-SMIS-SCR00024)

Autonomous HBA Profile

- Added this new "shim" Profile (Host-SMIS-SCR00024)

Comments

Editorial notes displayed.

Revison 2

Date

14 April 2007

SCRs Incorporated and other changes

Storage HBA Profile

- Added the Storage HBA profile (No SCR) (5-0-0)

Comments

Only minor editorial work for this revision.

Revison 3

Date

19 June 2007

SCRs Incorporated and other changes

Host Hardward RAID Controller

- Incorporate Realizes association from PhysicalPackage in Physical Asset profile to both PortController and the ComputerSystem representing the controller (No SCR) (1-0-0)

Comments

Editorial notes displayed.

Responses to INCITS editor queries re SMI-S 1.1.0 incorporated as applicable.

Typographical Conventions revised in all books: Revised explanation of Experimental text (per SMIS-120-Errata-SCR00061 - Typographical Conventions), added explanations of Draft and Editorial text.

Revision 4

Date

20 July 2007

SCRs Incorporated and other changes

Storage HBA
- **Promoted this profile to Experimental**

Host Hardward RAID Controller
- Add the Erasure Profile to the supported profiles of HHRc (Host-SMIS-SCR-00025) (3-0-0)

Comments

Editorial notes displayed, but the DRAFT material is not.

Revision 5

Date

14 November 2007

SCRs Incorporated and other changes

Clause 8: Host Hardware RAID Controller Profile

- Cleaned up leftover references to HHR in HHRC (Host-SMIS-SCR-00026)

Comments

Editorial notes and DRAFT material are not displayed.

Revision 6

Date

14 January 2009

SCRs Incorporated and other changes

References to *Storage Management Technical Specification, Part 7 Information Lifecycle Management*, deleted.

Removed test CARDINALITY (SMIS-130-Errata-SCR00001)

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

Invalid version numbers in supported profiles tables replaced with valid numbers (SMIS-130-Errata-SCR00017)

Replaced invalid HHRC association, cleaned up misspellings and diagram terms (SMIS-130-Errata-SCR00019)

Updated FCHBA profile (SMIS-130-Errata-SCR00036)

Updated HDR profile (SMIS-130-Errata-SCR00038)

Comments

Editorial notes and DRAFT material are not displayed.

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/.

Intended Audience

This document is intended for use by individuals and companies engaged in developing, deploying, and promoting interoperable multi-vendor SANs through the 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/.

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 Storage Networking Industry Association (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.

EXPERIMENTAL

Experimental content appears here.

EXPERIMENTAL

Figure 1 - Experimental Maturity Level Tag

Implemented Maturity Level

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

IMPLEMENTED

Implemented content appears here.

IMPLEMENTED

Figure 2 - Implemented Maturity Level Tag

Stable Maturity Level

Once content at the Implemented maturity level has garnered additional implementation experience, it can be tagged at the Stable maturity level. Material at this maturity level has been implemented by three different vendors, including both a provider and a client. Should material that has reached this maturity level become obsolete, it may only be deprecated as part of a minor revision to the specification. Material at this maturity level that has been deprecated may only be removed from the specification as part of a major revision. A profile that has reached this maturity level is guaranteed to preserve backward compatibility from one minor specification revision to the next. As a result, Profiles at or above the Stable maturity level shall not rely on any content that is Experimental. Figure 3 is a sample of the typographical convention for Implemented content.

STABLE

Stable content appears here.

STABLE

Figure 3 - Stable Maturity Level Tag

Finalized Maturity Level

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

Deprecated Material

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

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

DEPRECATED

Content that has been deprecated appears here.

DEPRECATED

Figure 4 - Deprecated Tag

Usage

Contents

Revision History ... iii
List of Tables ... xiii
List of Figures ... xvii
Foreword ... xix
1. Scope ... 1
2. Normative References ... 3
2.1 Approved references ... 3
2.2 References under development ... 3
2.3 Other references ... 3
3. Terms and definitions ... 5
3.1 General ... 5
3.2 Definitions ... 5
4. Disk Partition Subprofile ... 7
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
5. FC HBA Profile ... 27
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
6. Storage HBA Profile ... 59
6.1 Synopsis ... 59
6.2 Description ... 59
6.3 Implementation ... 59
6.4 Methods of the Profile ... 65
6.5 Use Cases ... 65
6.6 CIM Elements ... 66
7. Host Discovered Resources Profile ... 69
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
8. Host Hardware RAID Controller Profile ... 85
8.1 Synopsis ... 85
8.2 Description ... 86
8.3 Implementation ... 86
8.4 Methods ... 98
8.5 Use Cases ... 98
8.6 CIM Elements ... 99
9. iSCSI Initiator Profile ... 111

9.1 Description 111
9.2 Health and Fault Management Considerations 114
9.3 Supported Subprofiles and Packages 114
9.4 Methods of the Profile 114
9.5 Client Considerations and Recipes 114
9.6 Registered Name and Version 115
9.7 CIM Elements 115

10. SCSI Multipath Management Subprofile 129

10.1 Description 129
10.2 Health and Fault Management Considerations 131
10.3 Cascading Considerations 131
10.4 Supported Subprofiles and Packages 131
10.5 Methods of the Profile 131
10.6 Client Considerations and Recipes 132
10.7 Registered Name and Version 138
10.8 CIM Elements 138

11. SB Multipath Management Profile 141

11.1 Description 141
11.2 Health and Fault Management Consideration 142
11.3 Cascading Considerations 142
11.4 Supported Profiles, Subprofiles, and Packages 142
11.5 Methods 142
11.6 Client Considerations and Recipes 143
11.7 Registered Name and Version 143
11.8 CIM Elements 143

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_ElementStatisticalData....47
Table 31. SMI Referenced Properties/Methods for CIM_FCPort....47
Table 32. SMI Referenced Properties/Methods for CIM_FCPortStatistics....47
Table 33. SMI Referenced Properties/Methods for CIM_HostedCollection....48
Table 34. SMI Referenced Properties/Methods for CIM_HostedService....49
Table 35. SMI Referenced Properties/Methods for CIM_InstalledSoftwareIdentity (Driver)....49
Table 36. SMI Referenced Properties/Methods for CIM_LogicalPortGroup....49
Table 37. SMI Referenced Properties/Methods for CIM_MemberOfCollection....50
Table 38. SMI Referenced Properties/Methods for CIM_OSStorageNameBinding....50
Table 39. SMI Referenced Properties/Methods for CIM_PhysicalPackage....51
Table 40. SMI Referenced Properties/Methods for CIM_PortController....52
Table 41. SMI Referenced Properties/Methods for CIM_Product....52
Table 42. SMI Referenced Properties/Methods for CIM_ProductPhysicalComponent....53
Table 43. SMI Referenced Properties/Methods for CIM_Realizes....53
Table 44. SMI Referenced Properties/Methods for CIM_ServiceAvailableToElement....53
Table 45. SMI Referenced Properties/Methods for CIM_SoftwareIdentity (Driver)....54
Table 46. SMI Referenced Properties/Methods for CIM_SoftwareIdentity (FCode/BIOS)....54
Table 47. SMI Referenced Properties/Methods for CIM_SoftwareIdentity (Firmware)....55

Table 48. SMI Referenced Properties/Methods for CIM_StorageNameBinding 55
Table 49. SMI Referenced Properties/Methods for CIM_StorageNameBindingCapabilities 56
Table 50. SMI Referenced Properties/Methods for CIM_StorageNameBindingService 57
Table 51. SMI Referenced Properties/Methods for CIM_SystemDevice (Associates System to PortController) 57
Table 52. Related Profiles for Storage HBA 59
Table 53. CIM_PortController 65
Table 54. CIM_SystemDevice 65
Table 55. CIM Elements for Storage HBA 66
Table 56. SMI Referenced Properties/Methods for CIM_ControlledBy 66
Table 57. SMI Referenced Properties/Methods for CIM_PortController 67
Table 58. SMI Referenced Properties/Methods for CIM_Product 67
Table 59. SMI Referenced Properties/Methods for CIM_ProductElementComponent 67
Table 60. SMI Referenced Properties/Methods for CIM_Realizes 68
Table 61. SMI Referenced Properties/Methods for CIM_SystemDevice 68
Table 62. Supported Profiles for Host Discovered Resources 74
Table 63. CIM Elements for Host Discovered Resources 76
Table 64. SMI Referenced Properties/Methods for CIM_ATAInitiatorTargetLogicalUnitPath 77
Table 65. SMI Referenced Properties/Methods for CIM_ATAProtocolEndpoint 78
Table 66. SMI Referenced Properties/Methods for CIM_ComputerSystem 78
Table 67. SMI Referenced Properties/Methods for CIM_HostedAccessPoint 79
Table 68. SMI Referenced Properties/Methods for CIM_LogicalDisk 79
Table 69. SMI Referenced Properties/Methods for CIM_SCSIArbitraryLogicalUnit 80
Table 70. SMI Referenced Properties/Methods for CIM_SCSIInitiatorTargetLogicalUnitPath 80
Table 71. SMI Referenced Properties/Methods for CIM_SCSIProtocolEndpoint 81
Table 72. SMI Referenced Properties/Methods for CIM_StorageExtent 81
Table 73. SMI Referenced Properties/Methods for CIM_SystemDevice 82
Table 74. SMI Referenced Properties/Methods for CIM_TapeDrive 82
Table 75. SMI Referenced Properties/Methods for SNIA_SBInitiatorTargetLogicalUnitPath 83
Table 76. SMI Referenced Properties/Methods for SNIA_SBProtocolEndpoint 84
Table 77. Related Profiles for Host Hardware RAID Controller 85
Table 78. CIM Elements for Host Hardware RAID Controller 99
Table 79. SMI Referenced Properties/Methods for CIM_AlarmDevice 101
Table 80. SMI Referenced Properties/Methods for CIM_AssociatedAlarm 101
Table 81. SMI Referenced Properties/Methods for CIM_ComputerSystem 101
Table 82. SMI Referenced Properties/Methods for CIM_ComputerSystemPackage (Associates PhysicalPackage to ComputerSystem) 102
Table 83. SMI Referenced Properties/Methods for CIM_ControlledBy 103
Table 84. SMI Referenced Properties/Methods for CIM_LogicalIdentity 103
Table 85. SMI Referenced Properties/Methods for CIM_PortController 103
Table 86. SMI Referenced Properties/Methods for CIM_Product 104
Table 87. SMI Referenced Properties/Methods for CIM_ProductPhysicalComponent 104
Table 88. SMI Referenced Properties/Methods for CIM_ProtocolController 105
Table 89. SMI Referenced Properties/Methods for CIM_ProtocolControllerForUnit 105
Table 90. SMI Referenced Properties/Methods for CIM_Realizes (Associates PhysicalPackage to PortController) 106
Table 91. SMI Referenced Properties/Methods for CIM_SAPAvailableForElement 106
Table 92. SMI Referenced Properties/Methods for CIM_SoftwareIdentity (Driver) 106
Table 93. SMI Referenced Properties/Methods for CIM_SoftwareIdentity (FCode/BIOS) 107
Table 94. SMI Referenced Properties/Methods for CIM_SoftwareIdentity (Firmware) 107
Table 95. SMI Referenced Properties/Methods for CIM_SystemComponent 107

Table 96. SMI Referenced Properties/Methods for CIM_SystemDevice (Associates Hosting System to PortController) 108
Table 97. SMI Referenced Properties/Methods for CIM_SystemDevice (Associates System to AlarmDevice) 108
Table 98. SMI Referenced Properties/Methods for CIM_SystemDevice (System to ProtocolController) 109
Table 99. iSCSI Terminology 111
Table 100. OperationalStatus Values 114
Table 101. Supported Profiles for iSCSI Initiator 114
Table 102. CIM Elements for iSCSI Initiator 115
Table 103. SMI Referenced Properties/Methods for CIM_BindsTo (TCPProtocolEndpoint to IPProtocolEndpoint)................ 116
Table 104. SMI Referenced Properties/Methods for CIM_BindsTo (iSCSIProtocolEndpoint to TCPProtocolEndpoint).......... 116
Table 105. SMI Referenced Properties/Methods for CIM_ComputerSystem 117
Table 106. SMI Referenced Properties/Methods for CIM_ControlledBy 117
Table 107. SMI Referenced Properties/Methods for CIM_DeviceSAPImplementation (EthernetPort to IPProtocolEndpoint) 118
Table 108. SMI Referenced Properties/Methods for CIM_DeviceSAPImplementation (EthernetPort to iSCSIProtocolEndpoint). 118
Table 109. SMI Referenced Properties/Methods for CIM_ElementSoftwareIdentity 118
Table 110. SMI Referenced Properties/Methods for CIM_EndpointOfNetworkPipe (Between iSCSI Session and iSCSIProtoco-lENdpoint) 119
Table 111. SMI Referenced Properties/Methods for CIM_EndpointOfNetworkPipe (Between iSCSI connection and TCPProto-colENdpoint) 119
Table 112. SMI Referenced Properties/Methods for CIM_InstalledSoftwareIdentity 120
Table 113. SMI Referenced Properties/Methods for CIM_NetworkPipeComposition 120
Table 114. SMI Referenced Properties/Methods for CIM_PhysicalPackage 120
Table 115. SMI Referenced Properties/Methods for CIM_PortController 121
Table 116. SMI Referenced Properties/Methods for CIM_Product 121
Table 117. SMI Referenced Properties/Methods for CIM_ProductPhysicalComponent 122
Table 118. SMI Referenced Properties/Methods for CIM_ProtocolControllerForPort 122
Table 119. SMI Referenced Properties/Methods for CIM_Realizes 122
Table 120. SMI Referenced Properties/Methods for CIM_SAPAvailableForElement 123
Table 121. SMI Referenced Properties/Methods for CIM_SCSIProtocolController 123
Table 122. SMI Referenced Properties/Methods for CIM_SoftwareIdentity 124
Table 123. SMI Referenced Properties/Methods for CIM_SystemDevice 124
Table 124. SMI Referenced Properties/Methods for CIM_SystemDevice 124
Table 125. SMI Referenced Properties/Methods for CIM_SystemDevice 125
Table 126. SMI Referenced Properties/Methods for CIM_iSCSIConnection 125
Table 127. SMI Referenced Properties/Methods for CIM_iSCSISession 126
Table 128. CIM Elements for SCSI Multipath Management 138
Table 129. SMI Referenced Properties/Methods for CIM_SCSIInitiatorTargetLogicalUnitPath 139
Table 130. CIM Elements for SB Multipath Management 143
Table 131. SMI Referenced Properties/Methods for SNIA_SBInitiatorTargetLogicalUnitPath 144

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 60
Figure 15. Profile Registration Profile 61
Figure 16. Software Inventory Profile in Storage HBA 63
Figure 17. HBA Card with Physical Classes 64
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 86
Figure 25. Alarms in Host Hardware RAID Controller 87
Figure 26. Implementation of Physical Asset Profile 88
Figure 27. Block Services Package in Host Hardware RAID Controller 89
Figure 28. DAPort Subprofile in Host Hardware Controller 91
Figure 29. Software Inventory Profile in Host Hardware RAID Controller 92
Figure 30. Generic Initiator Port and Disk Drive Lite Subprofile 93
Figure 31. Host Hardware RAID Connected to a Storage Virtualizer 95
Figure 32. Example of Mutli-Function Controllers 97
Figure 33. iSCSI Product and Package Model 112
Figure 34. iSCSI Sessions and Connections Model 113
Figure 35. iSCSI Initiator Node 113
Figure 36. Multipath Management Class Diagram 129
Figure 37. Four Path Instance Diagram 130
Figure 38. Four SB Channel Instance Diagram 141
Figure 39. Two SB Channel, Three Volume Instance Diagram 142

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, 1.3.0 Rev 6*
- *Storage Management Technical Specification, Part 1 Common Architecture, 1.3.0 Rev 6*
- *Storage Management Technical Specification, Part 2 Common Profiles, 1.3.0 Rev 6*
- *Storage Management Technical Specification, Part 3 Block Devices, 1.3.0 Rev 6*
- *Storage Management Technical Specification, Part 4 File Systems, 1.3.0 Rev 6*
- *Storage Management Technical Specification, Part 5 Fabric, 1.3.0 Rev 6*
- *Storage Management Technical Specification, Part 6 Host Elements, 1.3.0 Rev 6*
- *Storage Management Technical Specification, Part 7 Media Libraries, 1.3.0 Rev 6*

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, 500 Sansome Street, Suite #504, San Francisco, CA 94111, U.S.A.

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:

Organization Represented	*Name of Representative*
Brocade	John Crandall
Dell	Vance Corn
EMC	Mike Thompson
Hewlett Packard	Alex Lenart
	Steve Peters
Hitachi Data Systems	Steve Quinn
Individual member	Tom West
IBM	Krishna Harathi
	Mike Walker
	Martine Wedlake
Olocity	Scott Baker
Pillar	Gary Steffens
Symantec	Steve Hand
	Paul von Behren

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:

- "The Host Discovered Resources Profile

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

- "The Fibre Channel HBA Profile

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

- "The Host Hardware RAID Profile

  This profile defines the model and functions of a host where a RAID Controller resides. This is a "shim" profile used in the absence of Base Server Profile, to instantiate an instance of ComputerSystem that represents the host computer. This profile is required to add compatibility with the Fibre Channel HBA profile.

- "The 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.)

- "The Storage Device Enclosure Profile

  This profile defines the modeling of instances, properties and functions for the management of a storage device enclosure.

This part of the specification defines component profiles used by the other autonomous profiles to describe aspects of host-based storage elements and services. The component profiles (subprofiles) defined in this version of the specification include:

- "The iSCSI Initiator Subprofile

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

- "The Disk Partition Subprofile

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

- "The 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.

- "The 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.

This part of the specification defines abstract profile(s) used as a model to create other specialized host-based storage autonomous profiles. The abstract profiles (subprofiles) defined in this version of the specification include:

- "The Generic FC Controller Profile

  This profile abstractly defines the model and function necessary to manage a Fibre Channel HBA. This profile is used as the basis for the autonomous Fibre Channel HBA profile and the future virtualization profiles.

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, 1.3.0 Rev 6

Storage Management Technical Specification, Part 2 Common Profiles, 1.3.0 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

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, 1.3.0 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™**[1] protocol), and IP.

3.2.2 Host Bus Adapter (HBA)

An I/O adapter that connects a host I/O bus to a computer's memory system.

3.2.3 Host Computer System

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

3.2.4 JBOD

An acronym for "Just a Bunch Of Disks." A cabinet or enclosure of disks.

3.2.5 Logical 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 standard 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.

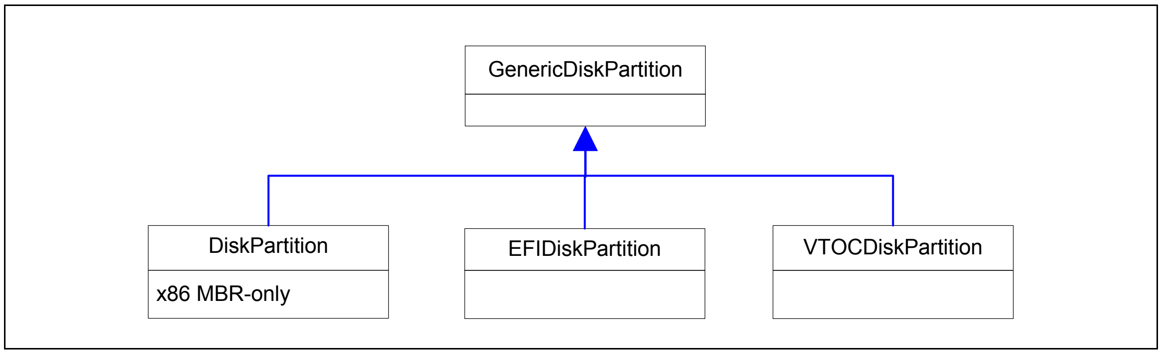

Figure 5 - Disk Partition Class Hierarchy

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.

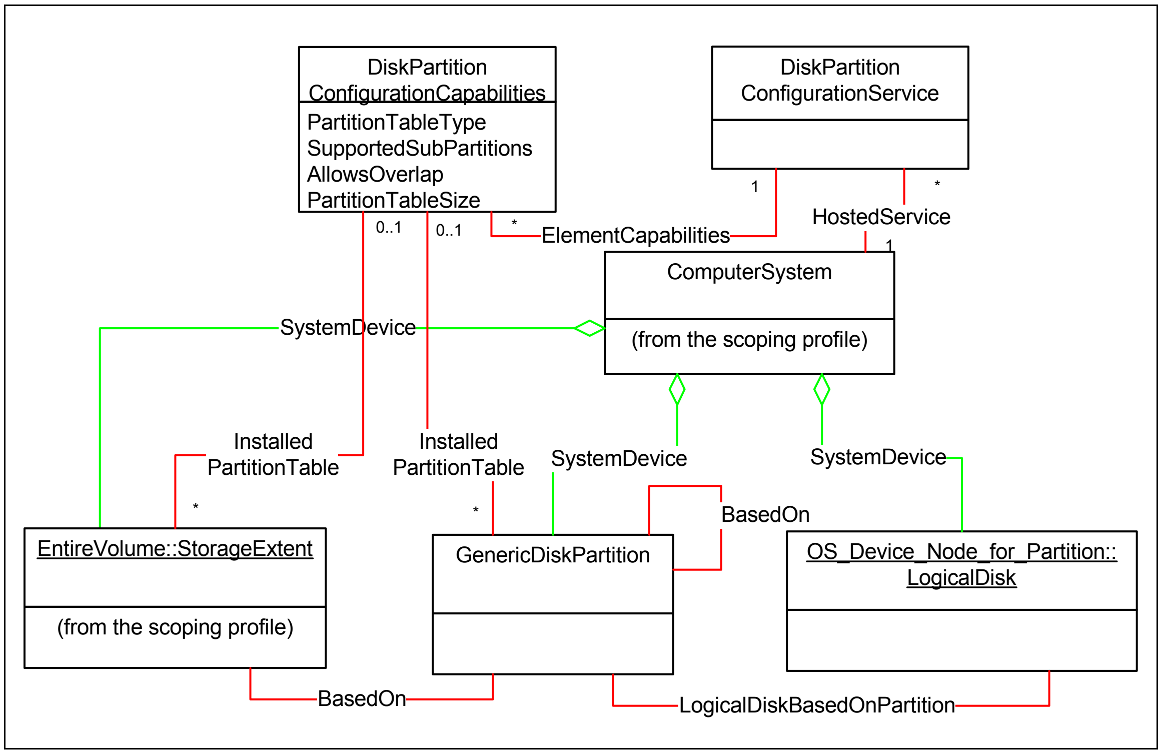

Figure 6 - Disk Partition Class Diagram

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.

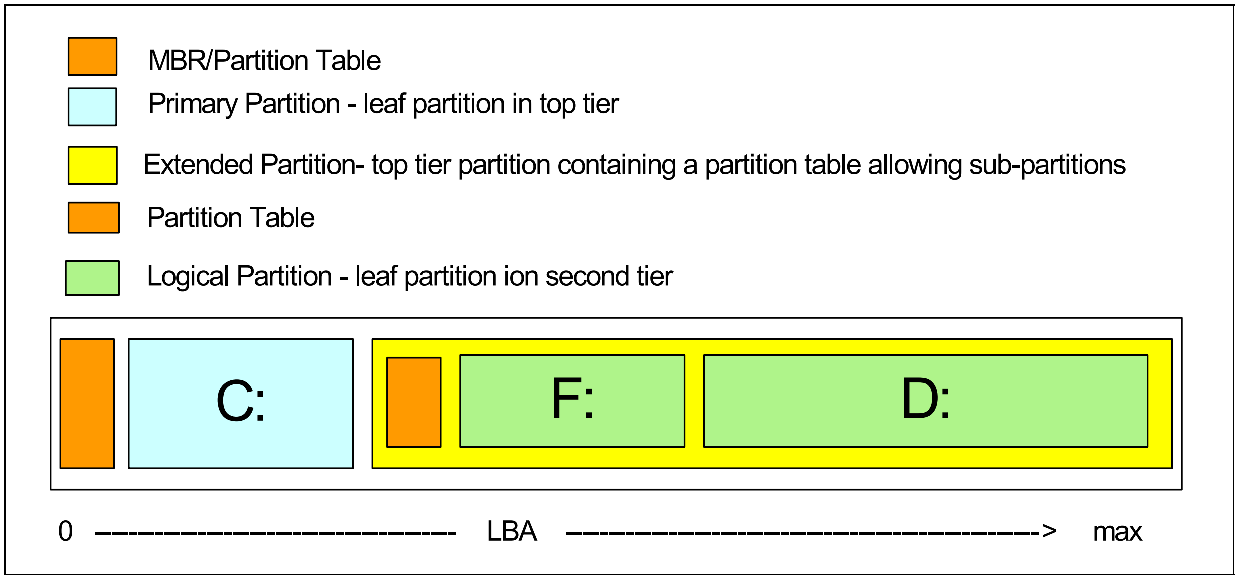
MBR/Partition Table
Primary Partition - leaf partition in top tier
Extended Partition- top tier partition containing a partition table allowing sub-partitions
Partition Table
Logical Partition - leaf partition ion second tier
C:
F:
D:
0 -- LBA ---> max

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:.

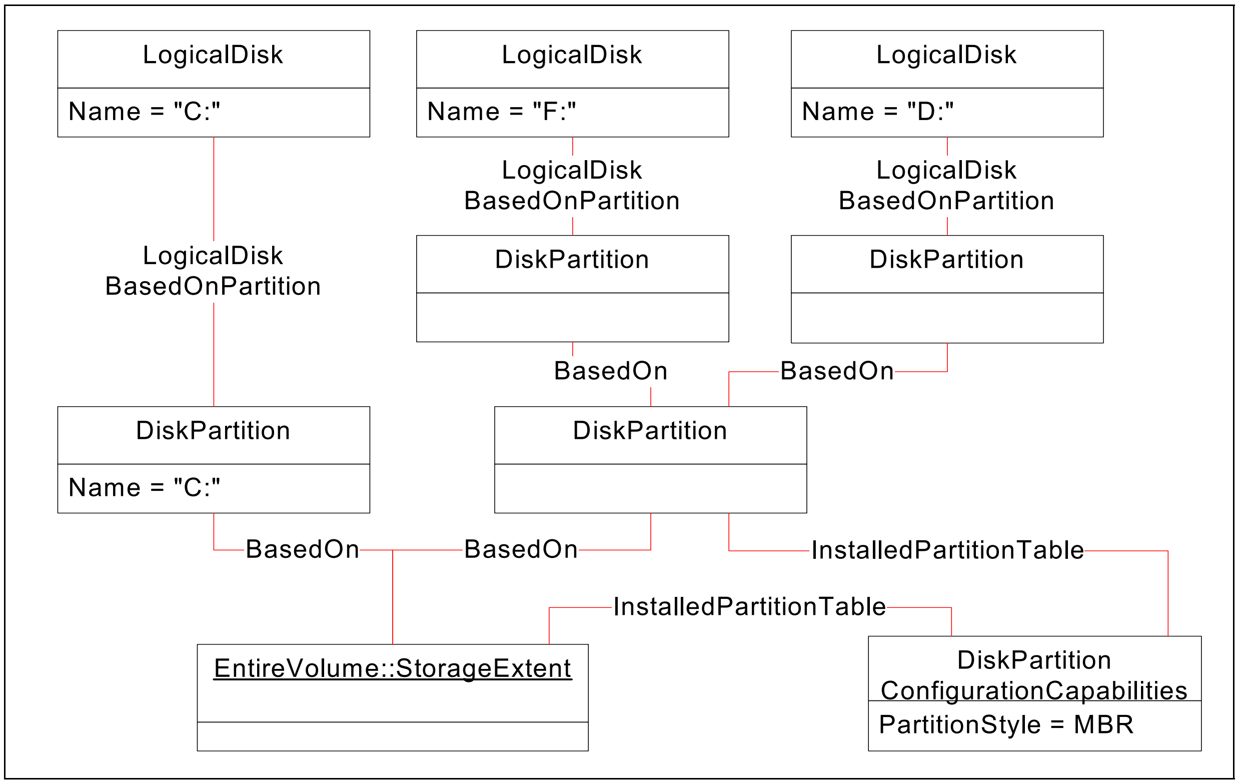

Figure 8 - MBR Partition Instance Diagram

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.

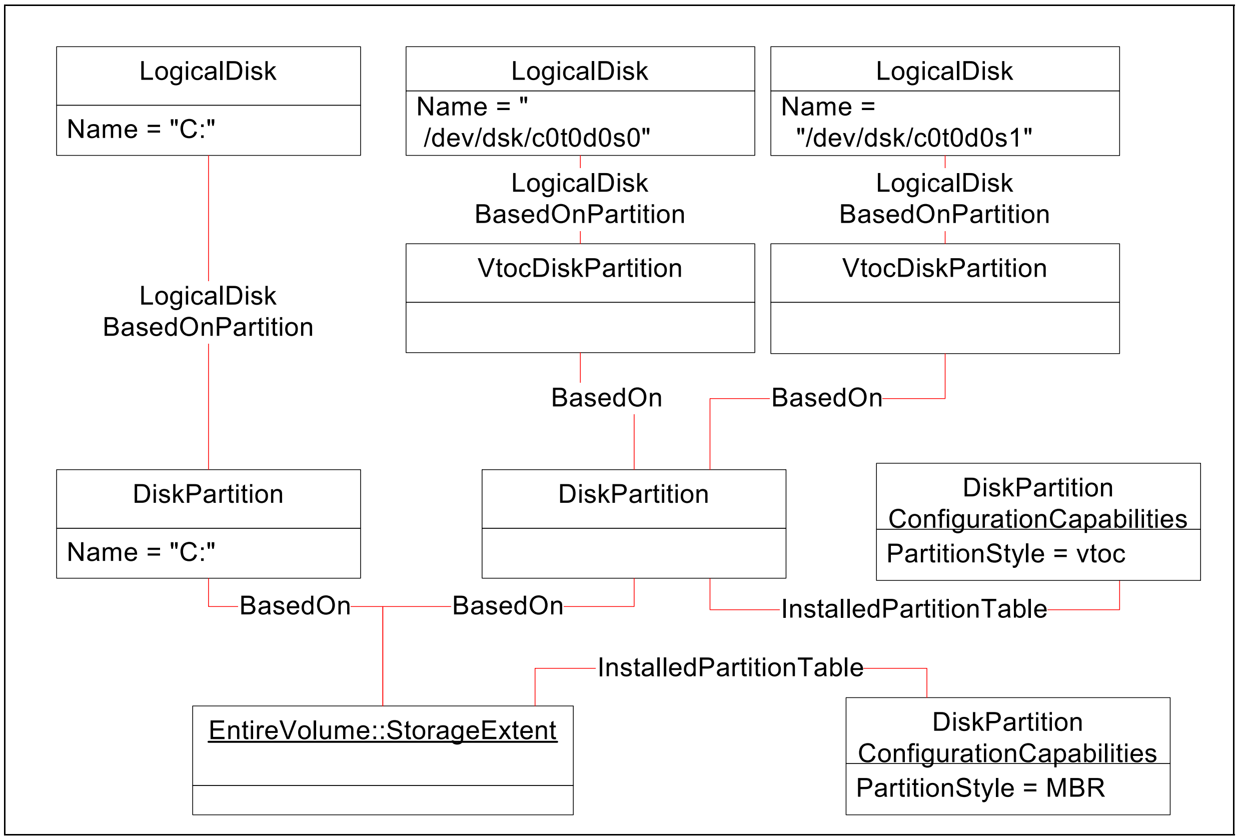
LogicalDisk
Name = "C:"
LogicalDisk
Name = " /dev/dsk/c0t0d0s0"
LogicalDisk
Name = "/dev/dsk/c0t0d0s1"
LogicalDisk BasedOnPartition
LogicalDisk BasedOnPartition
VtocDiskPartition
VtocDiskPartition
LogicalDisk BasedOnPartition
BasedOn
BasedOn
DiskPartition
Name = "C:"
DiskPartition
DiskPartition ConfigurationCapabilities
PartitionStyle = vtoc
BasedOn
BasedOn
InstalledPartitionTable
InstalledPartitionTable
EntireVolume::StorageExtent
DiskPartition ConfigurationCapabilities
PartitionStyle = MBR

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.

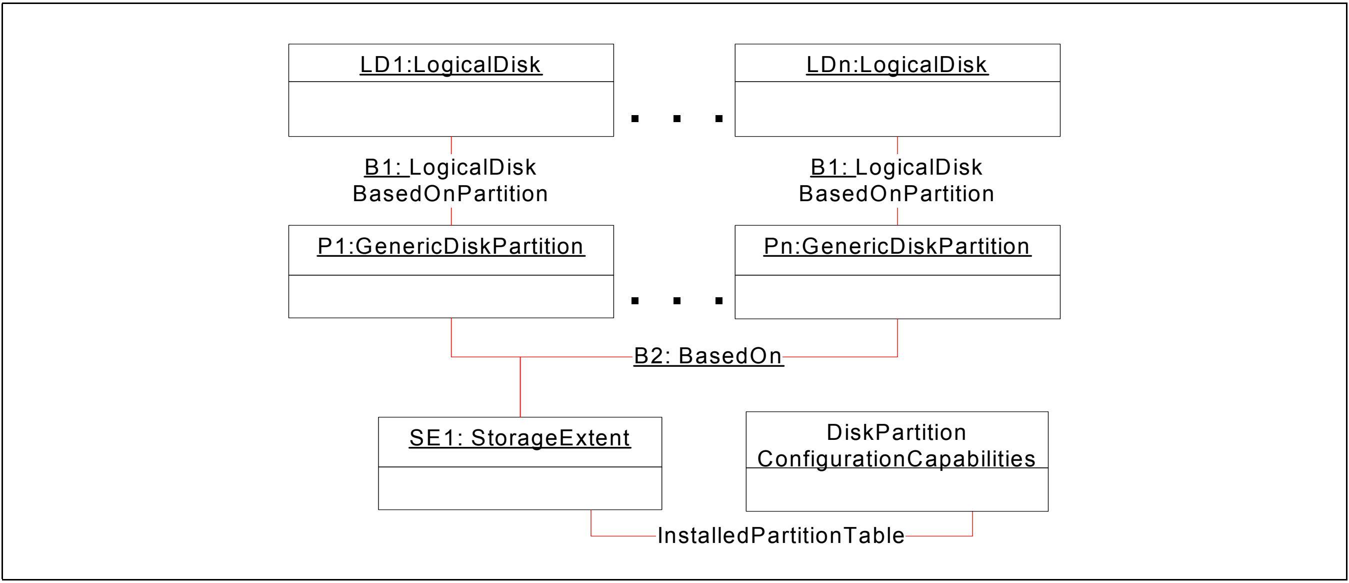

Figure 10 - Partition Instance Diagram for Size/Address Rules

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 a the DeviceFileName parameter is non-null, a LogicalDisk instance is created and associated via LogicalDiskBasedOnPartition to the partition. The underlying extent shall be associated to a capabilities class describing the installed partition style (partition table); this association is established using

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

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

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

       [IN, Description (
           "The platform-specific special file name to be assigned "
           "to the LogicalDisk instance BasedOn the new "
```

```
                "DiskPartition instance.")]
        string DeviceFileName,

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

```
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 assocaited 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(#MedthodReturn != 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(#MedthodReturn != 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
SystemCreationClassName		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
SystemCreationClassName		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: 1.3.0

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	1.3.0	Mandatory	
Software Installation Service	SNIA	1.2.0	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.

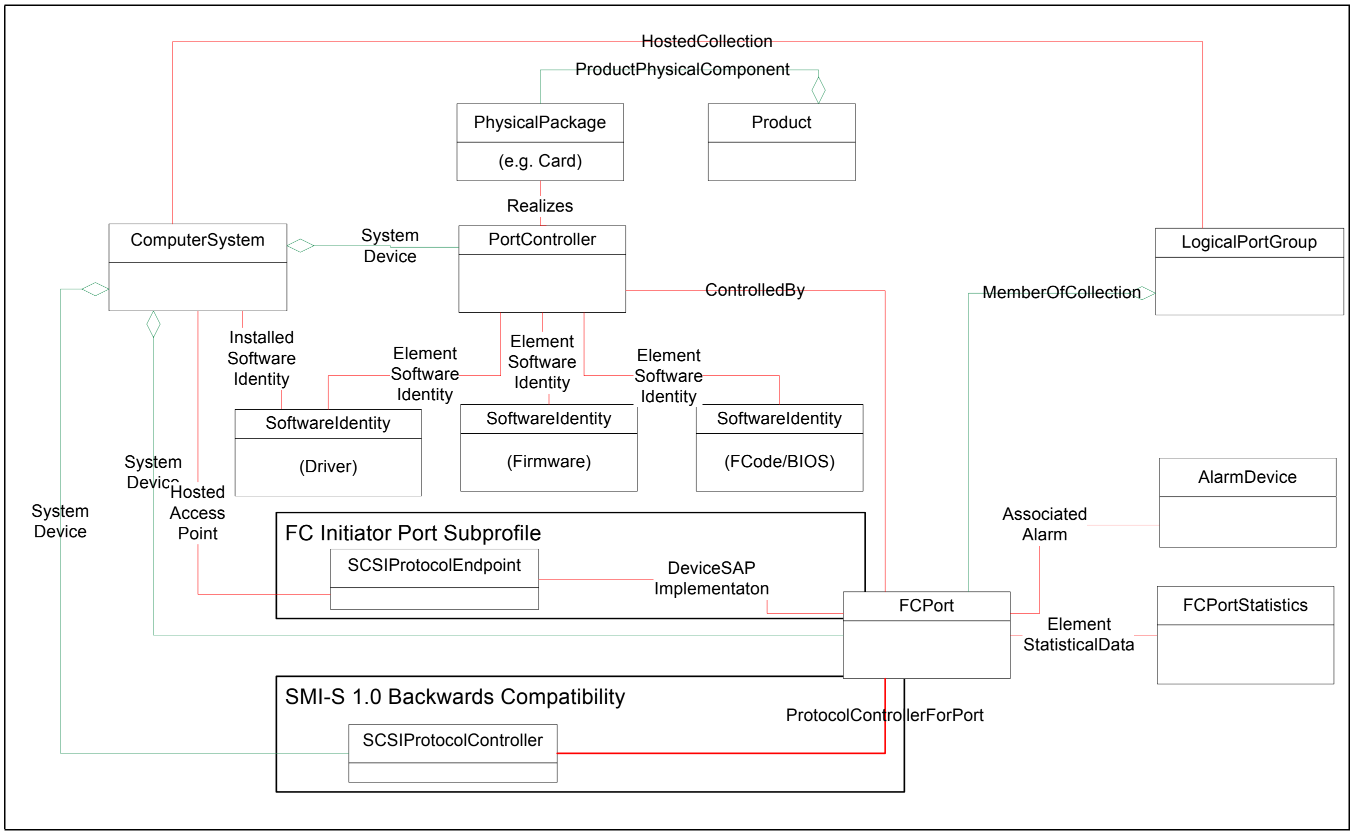

Figure 11 - FC HBA Instance Diagram

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, 1.3.0 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-SCSIProtocoEndpoint-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.

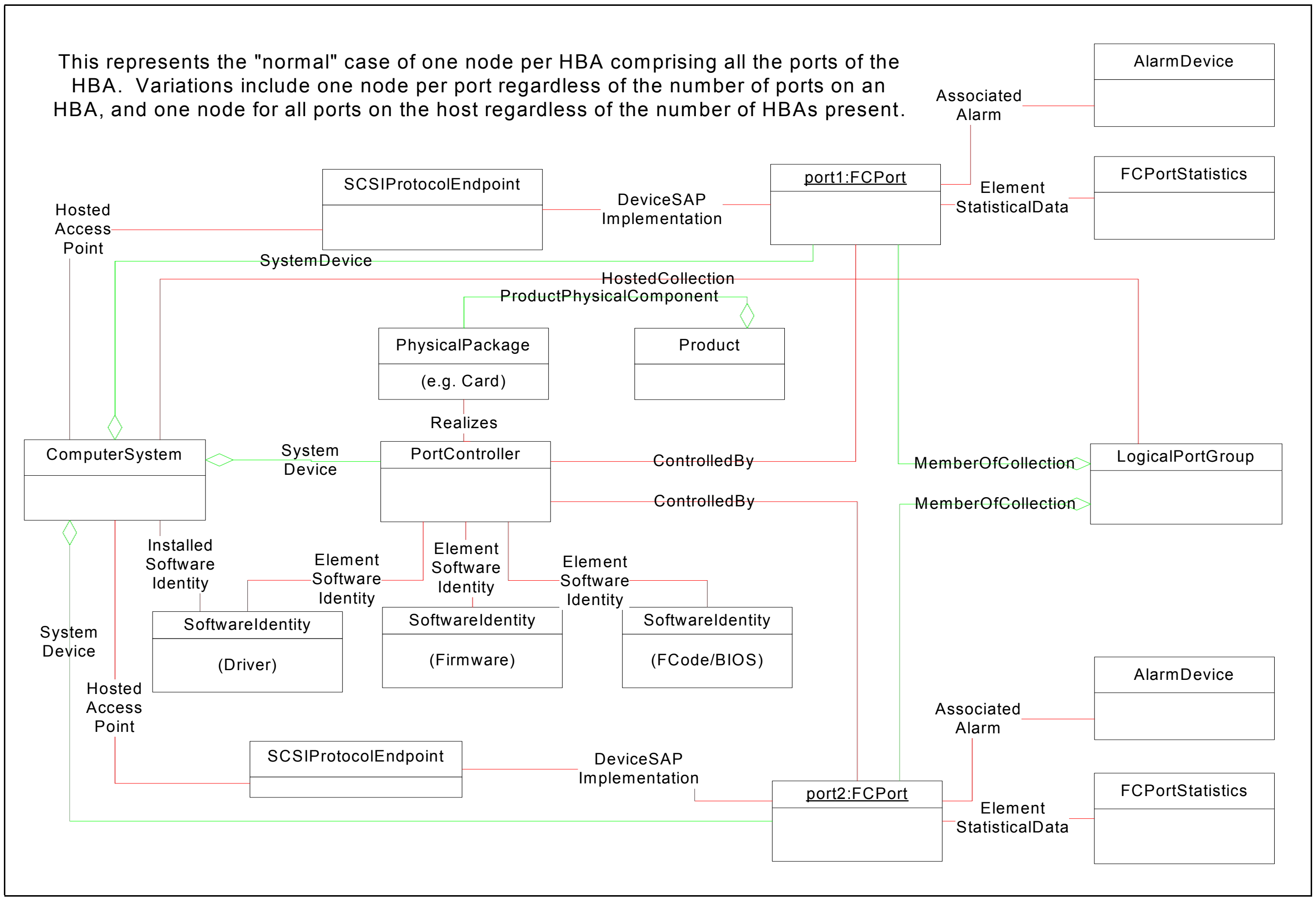

Figure 12 - HBA Card with Two Ports

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.

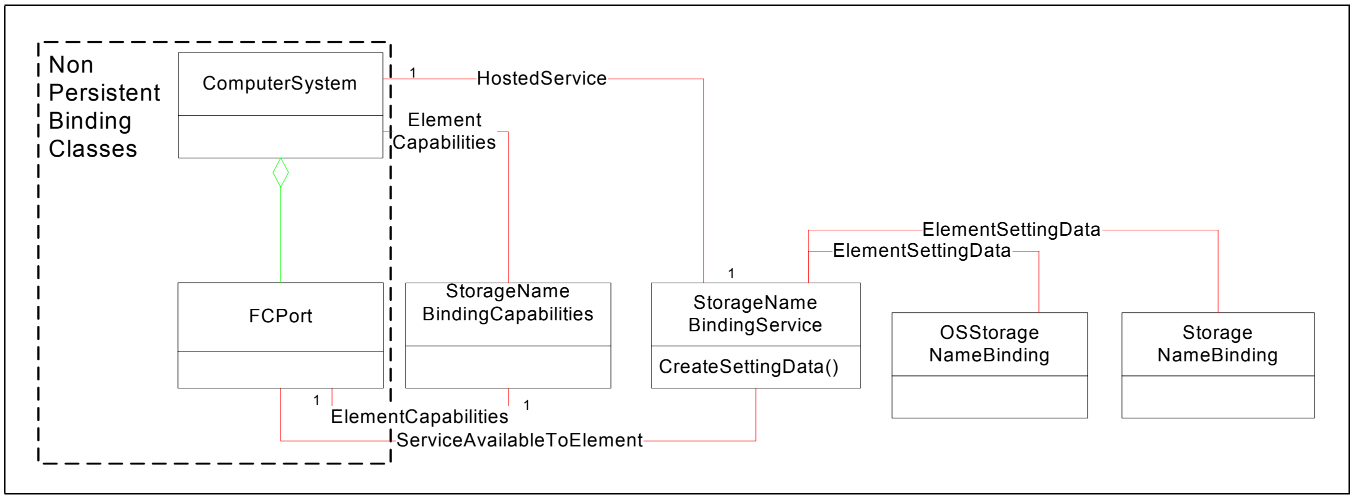

Figure 13 - Persistent Binding Model

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 yhan 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 demostrates 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 18 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_ElementStatisticalData	Mandatory	
5.7.12 CIM_FCPort	Mandatory	
5.7.13 CIM_FCPortStatistics	Mandatory	
5.7.14 CIM_HostedCollection	Optional	Associates the LogicalPortGroup (Fibre Channel Node) to the hosting System.
5.7.15 CIM_HostedService	Optional	
5.7.16 CIM_InstalledSoftwareIdentity (Driver)	Optional	
5.7.17 CIM_LogicalPortGroup	Optional	Collection of Fibre Channel ports that share a Node WWN
5.7.18 CIM_MemberOfCollection	Optional	Associates FCPort to the LogicalPortGroup

Table 19 - CIM Elements for FC HBA

Element Name	Requirement	Description
5.7.19 CIM_OSStorageNameBinding	Optional	
5.7.20 CIM_PhysicalPackage	Mandatory	
5.7.21 CIM_PortController	Mandatory	
5.7.22 CIM_Product	Mandatory	
5.7.23 CIM_ProductPhysicalComponent	Mandatory	
5.7.24 CIM_Realizes	Mandatory	
5.7.25 CIM_ServiceAvailableToElement	Optional	
5.7.26 CIM_SoftwareIdentity (Driver)	Mandatory	Driver
5.7.27 CIM_SoftwareIdentity (FCode/BIOS)	Optional	FCODE/BIOS
5.7.28 CIM_SoftwareIdentity (Firmware)	Optional	Firmware
5.7.29 CIM_StorageNameBinding	Optional	
5.7.30 CIM_StorageNameBindingCapabilities	Optional	
5.7.31 CIM_StorageNameBindingService	Optional	
5.7.32 CIM_SystemDevice (Associates System to PortController)	Mandatory	
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_PortController	Optional	PortController (HBA) Creation. See 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 SoftwareIdentityValidation Property : Limit to Driver
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 SoftwareIdentityValidation Property : Limit to FCode/BIOS
Dependent	Mandatory	Reference to 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 SoftwareIdentityValidation Property : Limit to Firmware
Dependent	Mandatory	Reference to the PortController

5.7.11 CIM_ElementStatisticalData

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

Table 30 describes class CIM_ElementStatisticalData.

Table 30 - SMI Referenced Properties/Methods for CIM_ElementStatisticalData

Properties	Requirement	Description & Notes
ManagedElement	Mandatory	Reference to FCPort (defined in FC Initiator Ports profile)
Stats	Mandatory	Reference to FCPortStatistics

5.7.12 CIM_FCPort

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

Table 31 describes class CIM_FCPort.

Table 31 - SMI Referenced Properties/Methods for CIM_FCPort

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

5.7.13 CIM_FCPortStatistics

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

Table 32 describes class CIM_FCPortStatistics.

Table 32 - SMI Referenced Properties/Methods for CIM_FCPortStatistics

Properties	Requirement	Description & Notes
ElementName	Mandatory	
InstanceID	Mandatory	
BytesTransmitted	Mandatory	From NetworkPortStatistics Superclass. Maps to HBA_PortStatistics.TxWords. Multiply word count by 4
BytesReceived	Mandatory	From NetworkPortStatistics Superclass. Maps to HBA_PortStatistics.RxWords. Multiply word count by 4
PacketsTransmitted	Mandatory	From NetworkPortStatistics Superclass. Maps to HBA_PortStatistics.TxFrames

Table 32 - SMI Referenced Properties/Methods for CIM_FCPortStatistics

Properties	Requirement	Description & Notes
PacketsReceived	Mandatory	From NetworkPortStatistics Superclass. Maps to HBA_PortStatistics.RxFrames
CRCErrors	Mandatory	Maps to HBA_PortStatistics.InvalidCRCCount
LinkFailures	Mandatory	Maps to HBA_PortStatistics.LinkFailureCount
PrimitiveSeqProtocolErrCount	Mandatory	
LossOfSignalCounter	Mandatory	Maps to HBA_PortStatistics.LossOfSignalCount
InvalidTransmissionWords	Mandatory	Maps to HBA_PortStatistics.InvalidTxWordCount
StatisticTime	Optional	Optional - time last measurement was taken
LIPCount	Mandatory	
NOSCount	Mandatory	
ErrorFrames	Mandatory	
DumpedFrames	Mandatory	
LossOfSyncCounter	Mandatory	Maps to HBA_PortStatistics.LossOfSynchCount

5.7.14 CIM_HostedCollection

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

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

Table 33 describes class CIM_HostedCollection.

Table 33 - SMI Referenced Properties/Methods for CIM_HostedCollection

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

5.7.15 CIM_HostedService

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

Table 34 describes class CIM_HostedService.

Table 34 - SMI Referenced Properties/Methods for CIM_HostedService

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

5.7.16 CIM_InstalledSoftwareIdentity (Driver)

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

Table 35 describes class CIM_InstalledSoftwareIdentity (Driver).

Table 35 - SMI Referenced Properties/Methods for CIM_InstalledSoftwareIdentity (Driver)

Properties	Requirement	Description & Notes
InstalledSoftware	Mandatory	Reference to Driver SoftwareIdentityValidation Property : Limit to Driver
System	Mandatory	Reference to ComputerSystem

5.7.17 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 36 describes class CIM_LogicalPortGroup.

Table 36 - 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.

5.7.18 CIM_MemberOfCollection

Associates FCPort to the LogicalPortGroup

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

Table 37 describes class CIM_MemberOfCollection.

Table 37 - 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)

5.7.19 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 38 describes class CIM_OSStorageNameBinding.

Table 38 - 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

Table 38 - SMI Referenced Properties/Methods for CIM_OSStorageNameBinding

Properties	Requirement	Description & Notes
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	Must be 2 - FC Port WWN
LocalPortName	Mandatory	initiator port WWN

5.7.20 CIM_PhysicalPackage

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

Table 39 describes class CIM_PhysicalPackage.

Table 39 - SMI Referenced Properties/Methods for CIM_PhysicalPackage

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

5.7.21 CIM_PortController

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

Table 40 describes class CIM_PortController.

Table 40 - 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)

5.7.22 CIM_Product

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

Table 41 describes class CIM_Product.

Table 41 - SMI Referenced Properties/Methods for CIM_Product

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

5.7.23 CIM_ProductPhysicalComponent

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

Requirement: Mandatory

Table 42 describes class CIM_ProductPhysicalComponent.

Table 42 - SMI Referenced Properties/Methods for CIM_ProductPhysicalComponent

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

5.7.24 CIM_Realizes

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

Table 43 describes class CIM_Realizes.

Table 43 - SMI Referenced Properties/Methods for CIM_Realizes

Properties	Requirement	Description & Notes
Dependent	Mandatory	Reference to PortController
Antecedent	Mandatory	Reference to PhyscialPackage

5.7.25 CIM_ServiceAvailableToElement

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

Table 44 describes class CIM_ServiceAvailableToElement.

Table 44 - 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)

5.7.26 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 45 describes class CIM_SoftwareIdentity (Driver).

Table 45 - 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)

5.7.27 CIM_SoftwareIdentity (FCode/BIOS)

FCODE/BIOS

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

Table 46 describes class CIM_SoftwareIdentity (FCode/BIOS).

Table 46 - 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)

5.7.28 CIM_SoftwareIdentity (Firmware)

Firmware

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

Table 47 describes class CIM_SoftwareIdentity (Firmware).

Table 47 - 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).

5.7.29 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 48 describes class CIM_StorageNameBinding.

Table 48 - 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_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

Table 48 - SMI Referenced Properties/Methods for CIM_StorageNameBinding

Properties	Requirement	Description & Notes
LocalPortNameType	Mandatory	Must be 2 - FC Port WWN
LocalPortName	Mandatory	initiator port WWN

5.7.30 CIM_StorageNameBindingCapabilities

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

Table 49 describes class CIM_StorageNameBindingCapabilities.

Table 49 - SMI Referenced Properties/Methods for CIM_StorageNameBindingCapabilities

Properties	Requirement	Description & Notes
ValidBindingTypes	Mandatory	API HBA_BIND_TYPE. Must include a subset of 2(FcApiBindToDID), 3(FcApiBindToWWPN), 4(FcApiBindToWWNN), or 5(BindToLUID)
ActivateBindingRequiresReset	Mandatory	True if creating a binding requires a system reboot
CanMapAddresses	Mandatory	True if the implementation allows overriding OS bus/target/ LUN numbers.
CanBindAllLuns	Mandatory	
AutoDiscovery	Mandatory	
CanSetOSDeviceName	Mandatory	

5.7.31 CIM_StorageNameBindingService

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

Table 50 describes class CIM_StorageNameBindingService.

Table 50 - SMI Referenced Properties/Methods for CIM_StorageNameBindingService

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

5.7.32 CIM_SystemDevice (Associates System to PortController)

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

Table 51 describes class CIM_SystemDevice (Associates System to PortController).

Table 51 - 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: 1.3.0

Organization: SNIA

CIM Schema Version: 2.15.0

Table 52 describes the related profiles for Storage HBA.

Table 52 - Related Profiles for Storage HBA

Profile Name	Organization	Version	Requirement	Description
Generic Initiator Ports	SNIA	1.3.0	Mandatory	
Software Inventory	DMTF	1.3.0	Mandatory	
Software Update	DMTF	1.0.0	Optional	
Physical Asset	DMTF	1.0.0a	Optional	
Indication	SNIA	1.3.0	Mandatory	

6.2 Description

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

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

6.3 Implementation

6.3.1 Health and Fault Management Consideration

None in this version of the stanadard.

6.3.2 Cascading Considerations

None in this version of the stanadard.-

6.3.3 Storage HBA Model Overview

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

6.3.3.1 PhysicalPackage Requirements

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

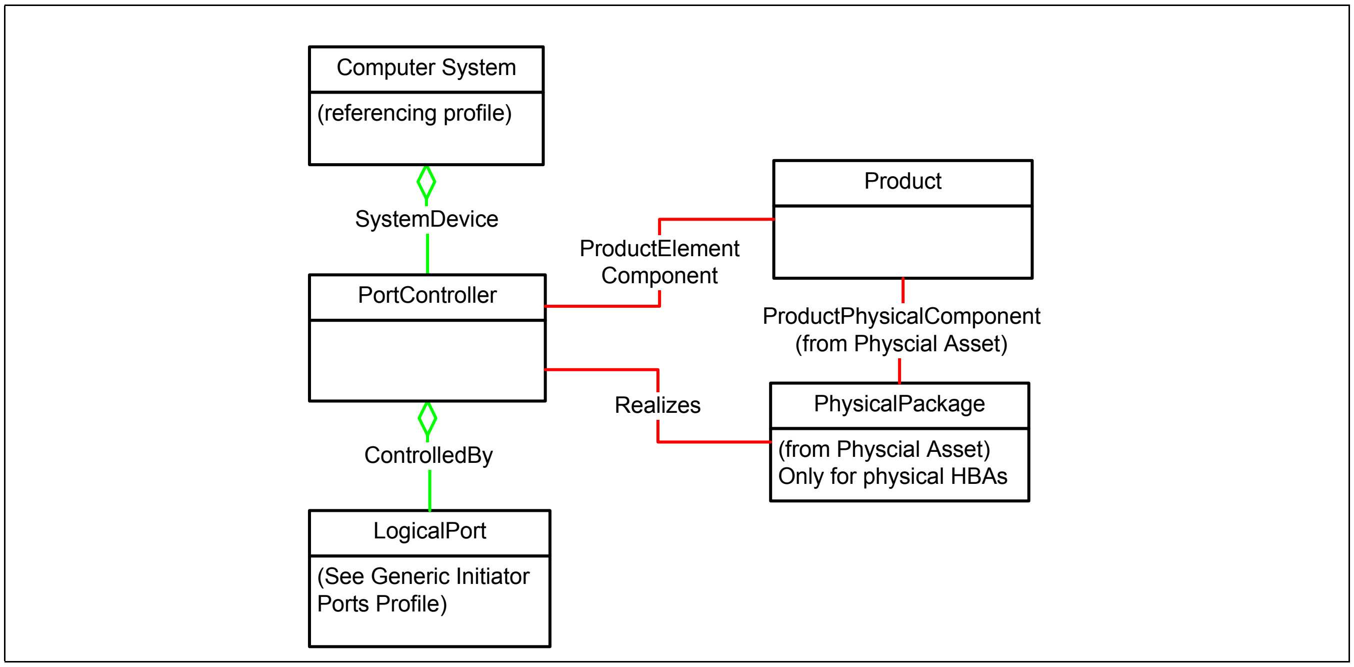

Figure 14 - Model Overview

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.

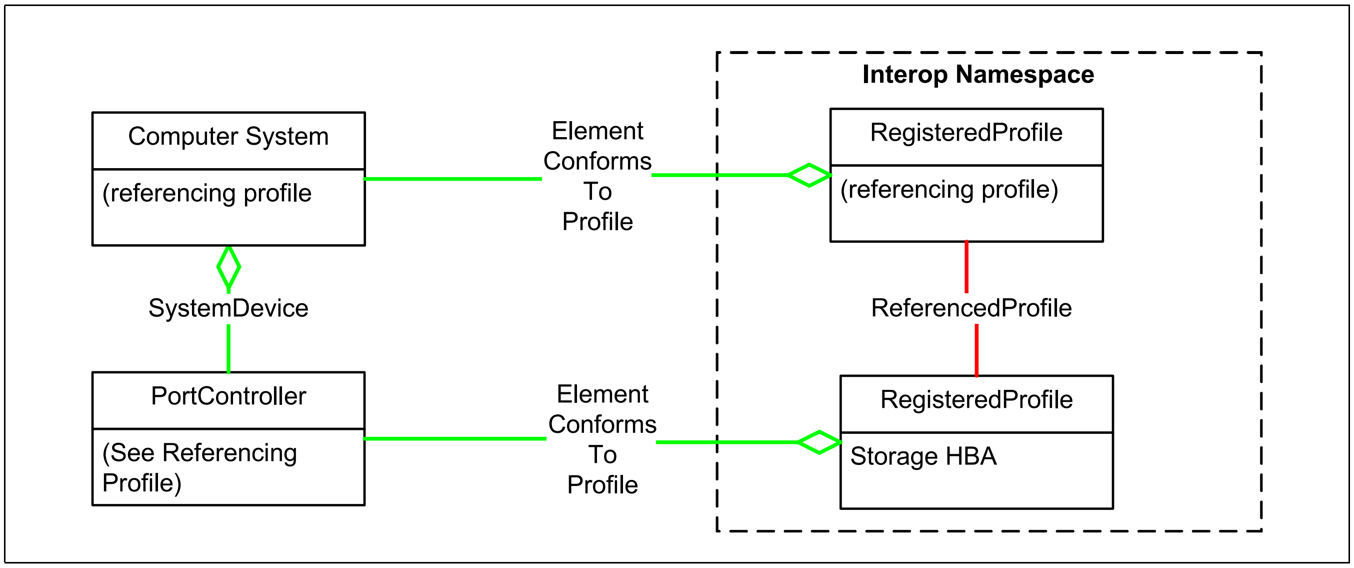

Figure 15 - Profile Registration Profile

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

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

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

6.3.7.2 Virtual HBA Considerations

The implementation shall use the Software Inventory profile 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.

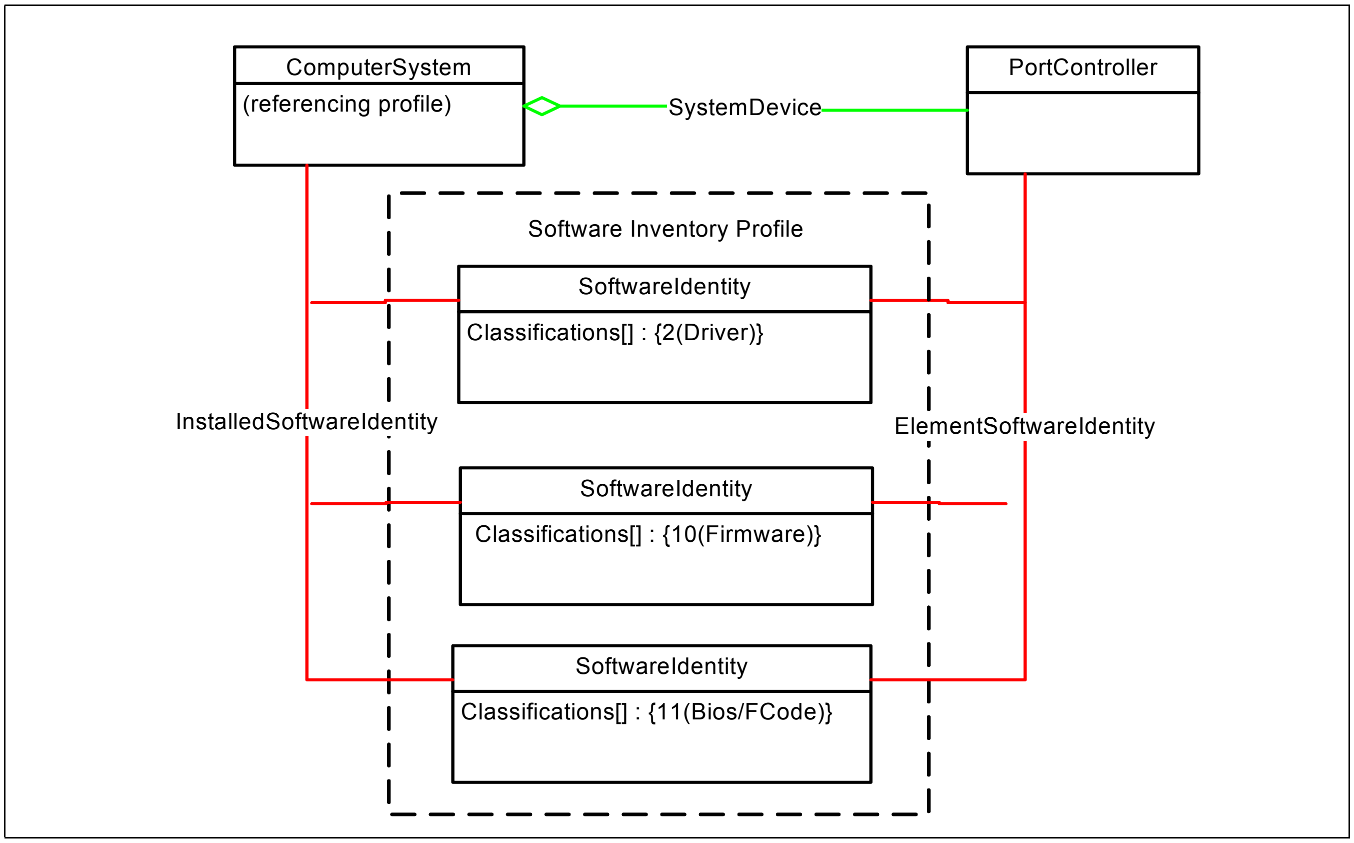
ComputerSystem
(referencing profile)
SystemDevice
PortController
Software Inventory Profile
SoftwareIdentity
Classifications[] : {2(Driver)}
InstalledSoftwareIdentity
ElementSoftwareIdentity
SoftwareIdentity
Classifications[] : {10(Firmware)}
SoftwareIdentity
Classifications[] : {11(Bios/FCode)}

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).

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.

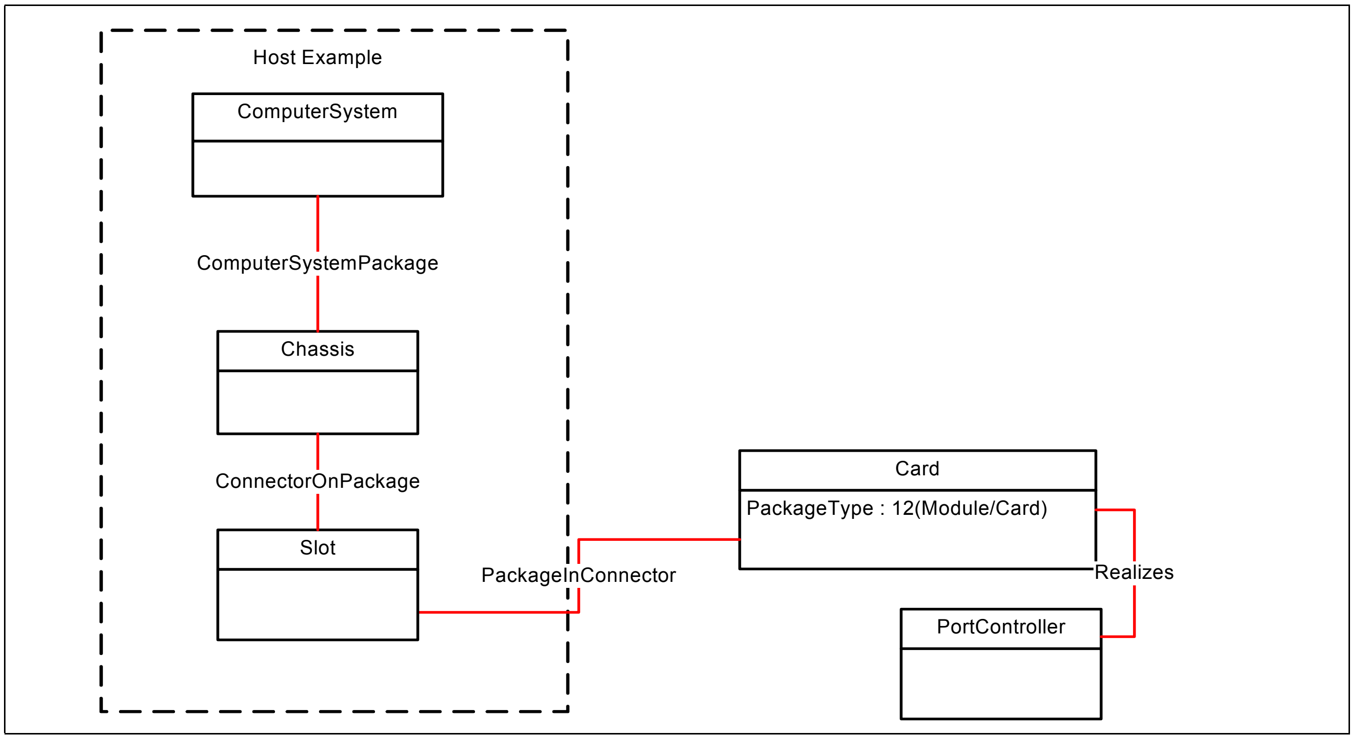

Figure 17 - HBA Card with Physical Classes

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 53 - 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 54 - 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 52 describes the CIM elements for Storage HBA.

Table 55 - 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

6.6.1 CIM_ControlledBy

Associates PortController and LogicalPort.

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

Table 56 describes class CIM_ControlledBy.

Table 56 - 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 refereed to as the bus number.
Dependent	Mandatory	
Antecedent	Mandatory	

6.6.2 CIM_PortController

Represents the logical aspects of the HBA.

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

Table 57 describes class CIM_PortController.

Table 57 - 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 58 describes class CIM_Product.

Table 58 - 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 59 describes class CIM_ProductElementComponent.

Table 59 - 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 60 describes class CIM_Realizes.

Table 60 - 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 61 describes class CIM_SystemDevice.

Table 61 - 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.

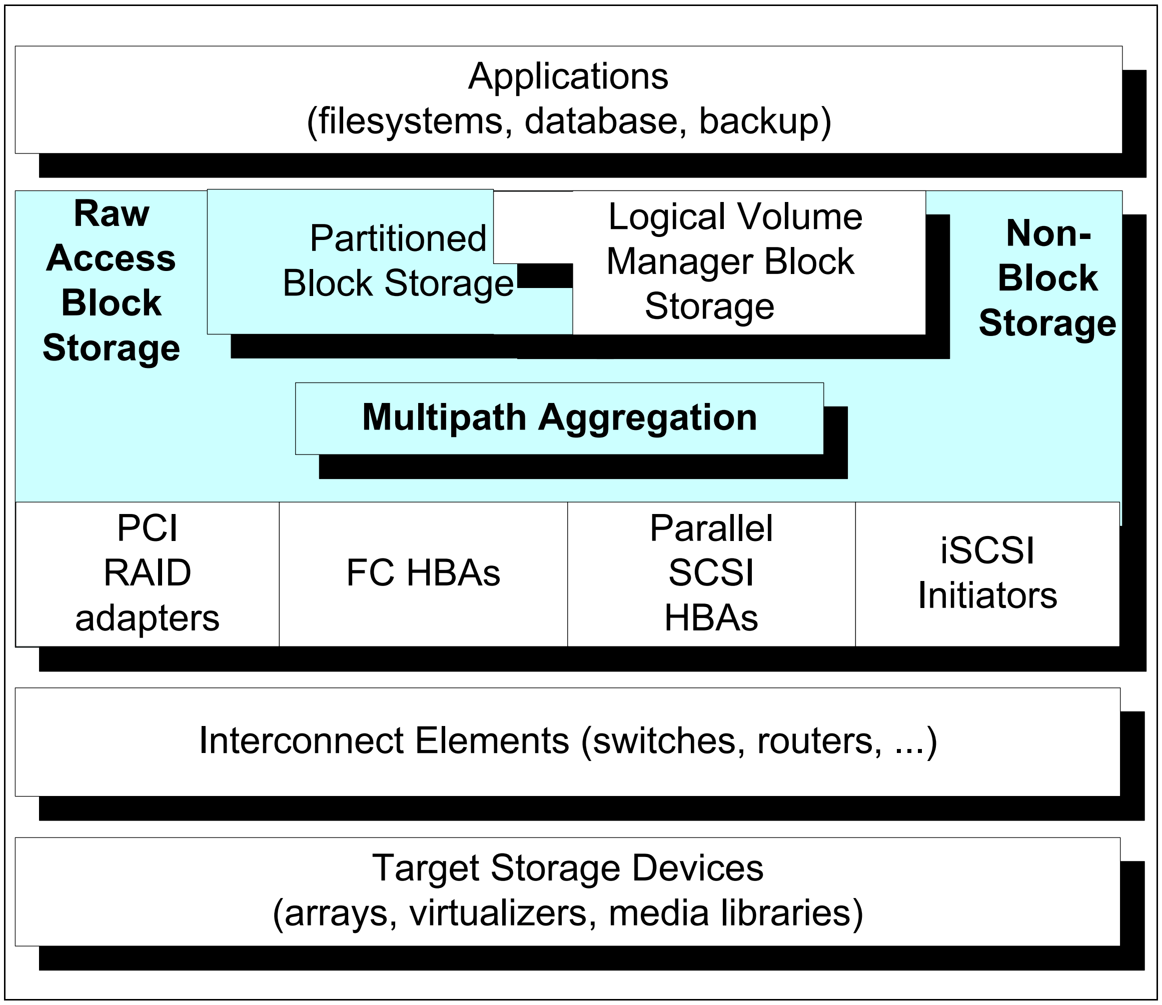
Applications
(filesystems, database, backup)
Raw Access Block Storage
Partitioned Block Storage
Logical Volume Manager Block Storage
Non-Block Storage
Multipath Aggregation
PCI RAID adapters
FC HBAs
Parallel SCSI HBAs
iSCSI Initiators
Interconnect Elements (switches, routers, ...)
Target Storage Devices
(arrays, virtualizers, media libraries)

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 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 Resouces - 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.

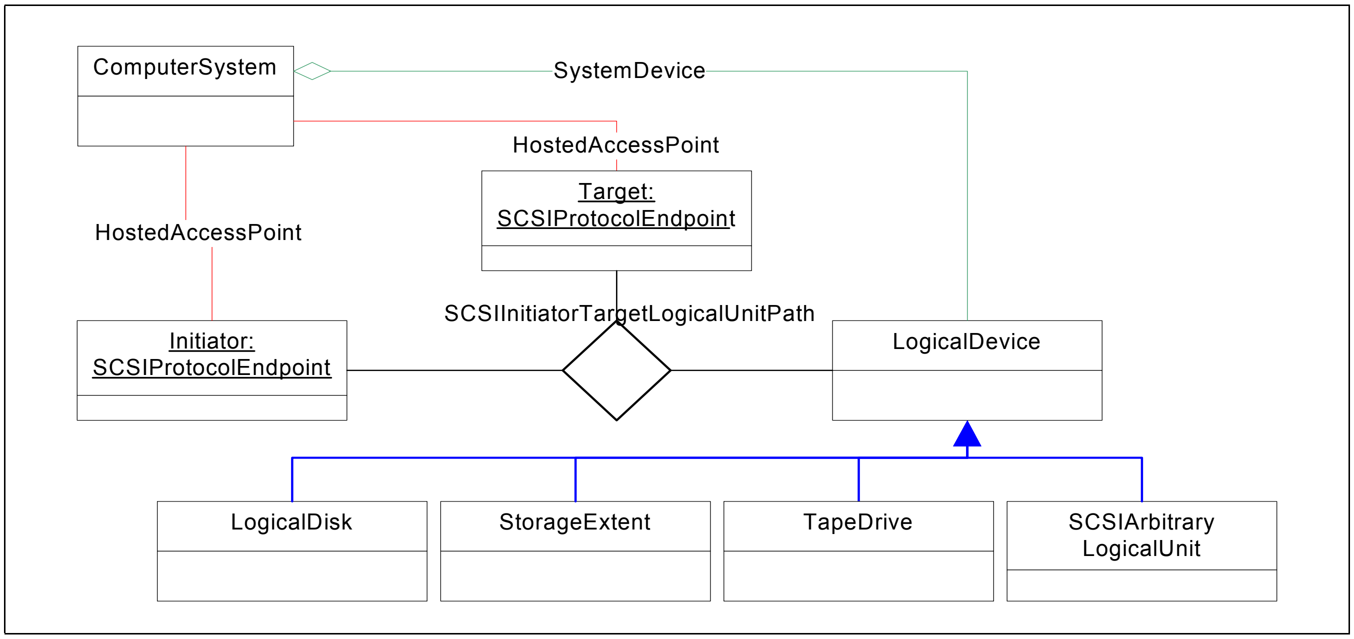
ComputerSystem
SystemDevice
HostedAccessPoint
Target: SCSIProtocolEndpoint
HostedAccessPoint
SCSIInitiatorTargetLogicalUnitPath
Initiator: SCSIProtocolEndpoint
LogicalDevice
LogicalDisk
StorageExtent
TapeDrive
SCSIArbitrary LogicalUnit

Figure 19 - Host Discovered Resources Class Diagram

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

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.

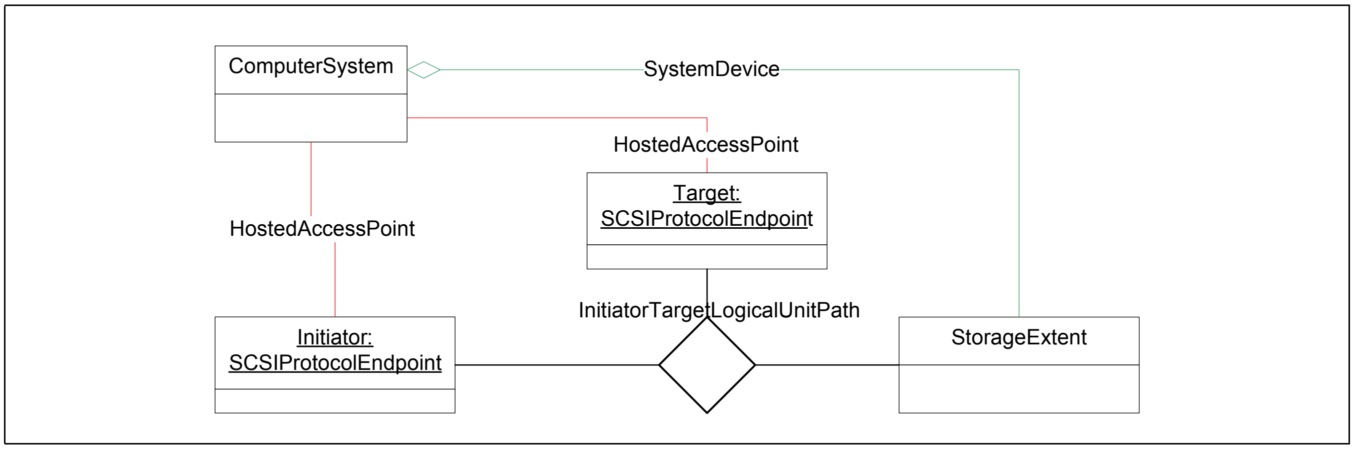

Figure 20 - Single SPI Disk Model

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.

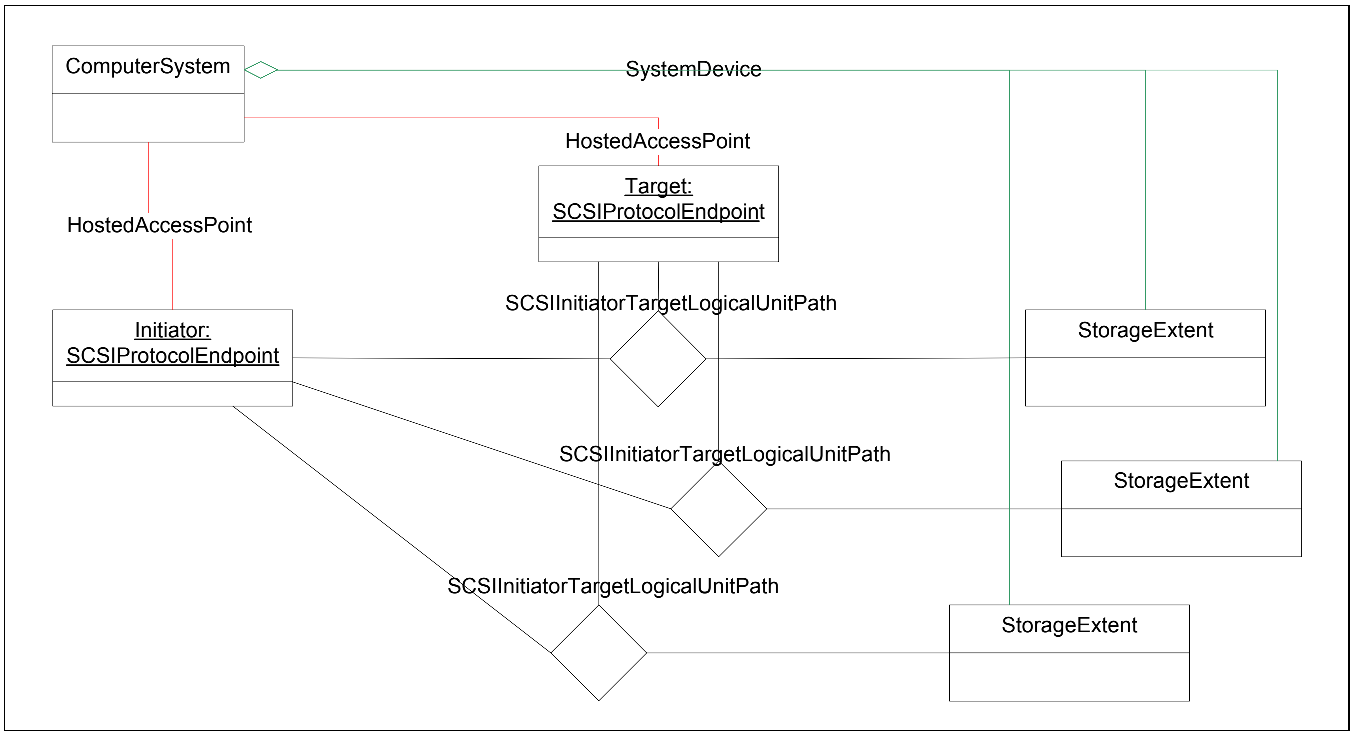

Figure 21 - Three FCP Logical Unit Instance Diagram

The Multipath Subprofile describes more complicated multipath configurations. See Figure 37.

Model for ATA Protocol Resources

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

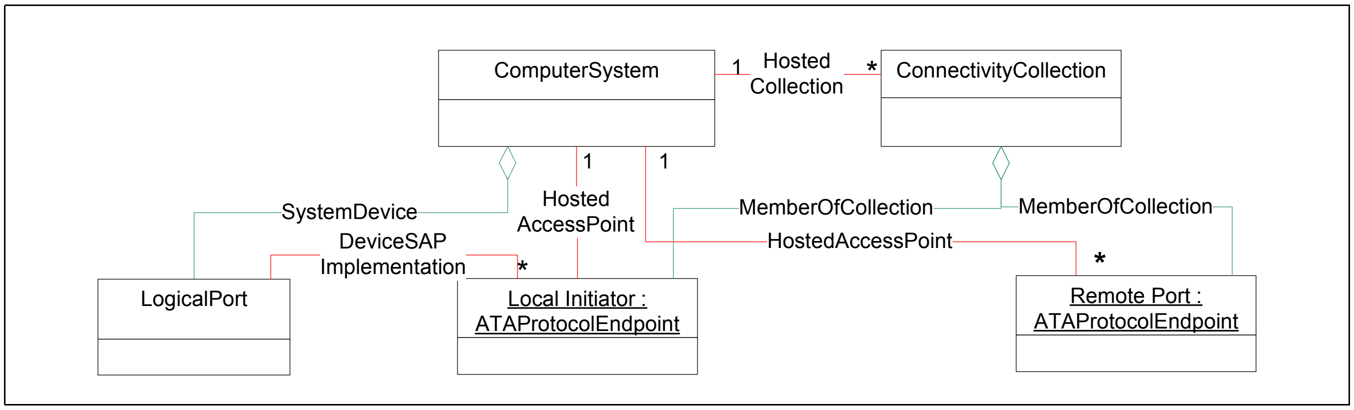

Figure 22 - ATA Discovered Resource Model

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 SBCUImage.

Figure 23 depicts SB Host Discovered Resources.

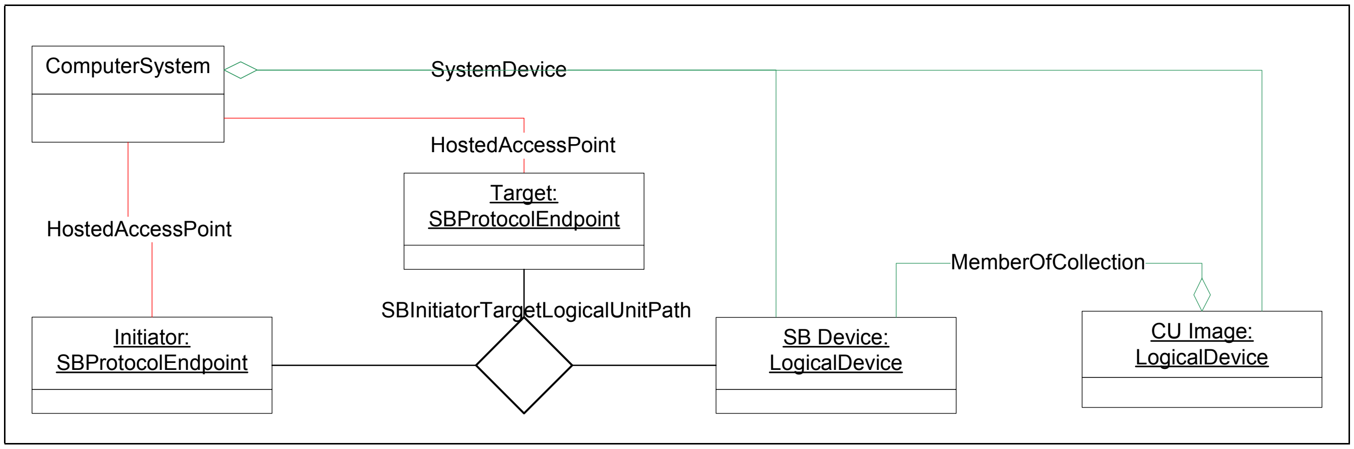

Figure 23 - SB Host Discovered Resources

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 62 describes the supported profiles for Host Discovered Resources.

Table 62 - Supported Profiles for Host Discovered Resources

Registered Profile Names	Mandatory	Version
SCSI Multipath Management	No	1.3.0
SB Multipath Management	No	1.0.0
Disk Partition	No	1.3.0

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 62 describes the CIM elements for Host Discovered Resources.

Table 63 - 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.
7.8.7 CIM_SCSIInitiatorTargetLogicalUnitPath	Optional	Associates initiator and target SCSIProtocolEndpoints to a logical unit.

Table 63 - CIM Elements for Host Discovered Resources

Element Name	Requirement	Description
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	

7.8.1 CIM_ATAInitiatorTargetLogicalUnitPath

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

Table 64 describes class CIM_ATAInitiatorTargetLogicalUnitPath.

Table 64 - 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 65 describes class CIM_ATAProtocolEndpoint.

Table 65 - SMI Referenced Properties/Methods for CIM_ATAProtocolEndpoint

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

7.8.3 CIM_ComputerSystem

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

Table 66 describes class CIM_ComputerSystem.

Table 66 - 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 67 describes class CIM_HostedAccessPoint.

Table 67 - 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 68 describes class CIM_LogicalDisk.

Table 68 - SMI Referenced Properties/Methods for CIM_LogicalDisk

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

7.8.6 CIM_SCSIArbitraryLogicalUnit

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

Table 69 describes class CIM_SCSIArbitraryLogicalUnit.

Table 69 - SMI Referenced Properties/Methods for CIM_SCSIArbitraryLogicalUnit

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

7.8.7 CIM_SCSIInitiatorTargetLogicalUnitPath

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

Table 70 describes class CIM_SCSIInitiatorTargetLogicalUnitPath.

Table 70 - 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 71 describes class CIM_SCSIProtocolEndpoint.

Table 71 - 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 72 describes class CIM_StorageExtent.

Table 72 - SMI Referenced Properties/Methods for CIM_StorageExtent

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	Opaque identifer
ElementName		Mandatory	User-friendly name
Name	C	Mandatory	OS device name
NameFormat	C	Mandatory	Shall be 12 (OS Device Name)
NameNamespace	C	Mandatory	Shall be 8 (OS Device NameSpace)

Table 72 - SMI Referenced Properties/Methods for CIM_StorageExtent

Properties	Flags	Requirement	Description & Notes
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 73 describes class CIM_SystemDevice.

Table 73 - 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 74 describes class CIM_TapeDrive.

Table 74 - SMI Referenced Properties/Methods for CIM_TapeDrive

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	Opaque identifer
ElementName		Mandatory	User-friendly name
Name	C	Mandatory	OS device name

Table 74 - SMI Referenced Properties/Methods for CIM_TapeDrive

Properties	Flags	Requirement	Description & Notes
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 75 describes class SNIA_SBInitiatorTargetLogicalUnitPath.

Table 75 - 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 whiether this is a preferred path
PathGroupState		Optional	One of 'Unknown, 'Path grouping not supported','Reset', 'Grouped', 'Ungrouped'
PathGroupMode		Optional	One of 'Unknown', 'None', 'Single path', 'Multipath' (SIngle path and multipath only v alid if PathGroupState is grouped.
PathGroupID		Optional	String containing the ID from the OS, only valid if PathGroupState is Grouped
LogicalUnit		Mandatory	A reference to a LogicalDevice
Initiator		Mandatory	A reference to the initiator CIM_SBProtocolEndpoint
Target		Mandatory	A reference to the target CIM_SBProtocolEndpoint

7.8.13 SNIA_SBProtocolEndpoint

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

Table 76 describes class SNIA_SBProtocolEndpoint.

Table 76 - SMI Referenced Properties/Methods for SNIA_SBProtocolEndpoint

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
Name	C	Mandatory	
ProtocolIFType		Mandatory	
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 77 describes the related profiles for Host Hardware RAID Controller.

Table 77 - Related Profiles for Host Hardware RAID Controller

Profile Name	Organization	Version	Requirement	Description
Physical Asset	DMTF	1.0.0a	Mandatory	
Block Services	SNIA	1.3.0	Mandatory	
Disk Drive Lite	SNIA	1.3.0	Optional	
Software Inventory	SNIA	1.3.0	Mandatory	
Software Update	DMTF	1.0.0	Optional	
Job Control	SNIA	1.3.0	Optional	
Extent Composition	SNIA	1.2.0	Optional	
Disk Sparing	SNIA	1.3.0	Optional	
Generic Initiator Ports	SNIA	1.3.0	Mandatory	
SPI Initiator Ports	SNIA	1.2.0	Optional	
FC Initiator Ports	SNIA	1.3.0	Optional	
ATA Initiator Ports	SNIA	1.2.0	Optional	
SAS/SATA Inititator Ports	SNIA	1.3.0	Optional	
SASInititator Ports	SNIA	1.3.0	Optional	
DA Target Ports	SNIA	1.2.0	Mandatory	
Indication	SNIA	1.3.0	Mandatory	
Erasure	SNIA	1.2.0	Optional	

The Host Hardware RAID Controller profile describes classes, properties, and other profiles necessary to manage a host-based RAID 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 SCSI or ATA disks contained within a server's internal drive cage or an external drive enclosure. In addition, a host-based RAID controller may manage physical aspects of RAID controller card, such as battery-backed cache, audible alarms, external SAS, SCSI or ATA ports or other miscellaneous sub-elements. The Host Hardware RAID Controller profile may be used to model manageability for software-based RAID included in drivers where storage volumes map to physical drives. However, manageability for volume manager-based RAID, running on a host operating system, is out of scope for the Host Hardware RAID Controller profile. In addition, the profile does not include enclosure management of storage devices connected to the controller. Storage device enclosure management is performed via the Storage Device 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 host-based RAID controller are described by this profile. Other classes that are common to other profiles, reference to by the Host Hardware RAID Controller profile, may be found in their respective section within this specification, the DMTF SMWG specifications, or the DMTF CIM schema specification.

While the Host Hardware RAID Controller profile describes the classes, associations and profiles necessary to represent a host-based RAID storage controller;

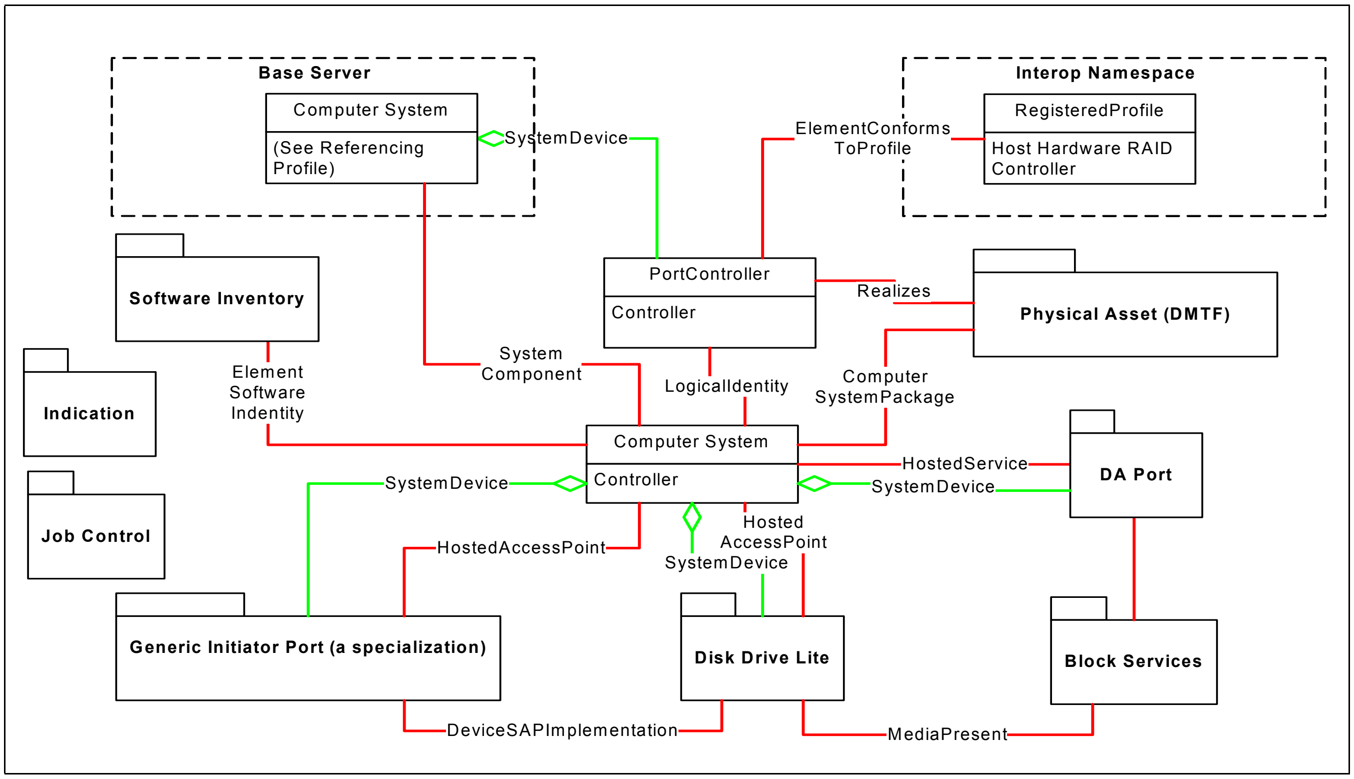

Figure 24 - Host Hardware RAID Controller Package Diagram

8.3 Implementation

8.3.1 CIM_PortController

The PortController class is the top-level and central class of the Host Hardware RAID Controller Profile. It represents an instance of a RAID controller and controllers the backend port to the storage managed by this controller. Any implementation of Host Hardware RAID Controller shall instantiate an instance of PortController that is associated to an instance of RegisteredProfile using the ElementConformsToProfile association. 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 ConcreteIdentity association. Finally, the PortController shall be associated to one or more instances of LogicalPort using the ControlledBy association.

8.3.2 SystemDevice

The SystemDevice association shall be used to associate the PortController in Host Hardware RAID Controller profile to the ComputerSystem that represents the Host in the referencing profile.

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 ConcreteIdentity association. Also, the implementation shall set the Dedicated array property to values of “Host Hardware RAID Controller” and “Block Server”.

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.

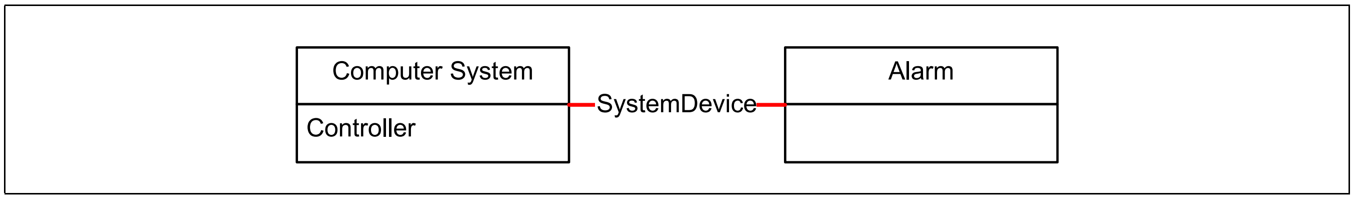

Figure 25 - Alarms in Host Hardware RAID Controller

8.3.5 Server Profile

For the Host Hardware RAID Controller Profile, the Server profile is required. Any implementation shall follow the requirements of the Server profile.

8.3.5.1 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 of profile registration from the Server Profile. The scoping class of the Host Hardware RAID Controller profile is the instance of CIM_ComputerSystem that represents the host where the controller is located. The instance of

RegisteredProfile in the Interop namespace shall have an association to the instance of CIM_ComputerSystem of the host in the implementation namespace.

8.3.5.2 Profile Discovery and Advertisement

For the Host Hardware RAID Controller profile, the profile shall advertise itself using the default advertising mechanism available from the CIMOM. If the CIMOM does not advertise profiles, then the profile is not required to advertise.

In some cases, SLP is used as the advertising protocol for the CIMOM. If SLP is used, the value of the RegisteredProfilesSupported attribute shall include “SNIA:Host Hardware RAID Controller”

8.3.6 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). 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.

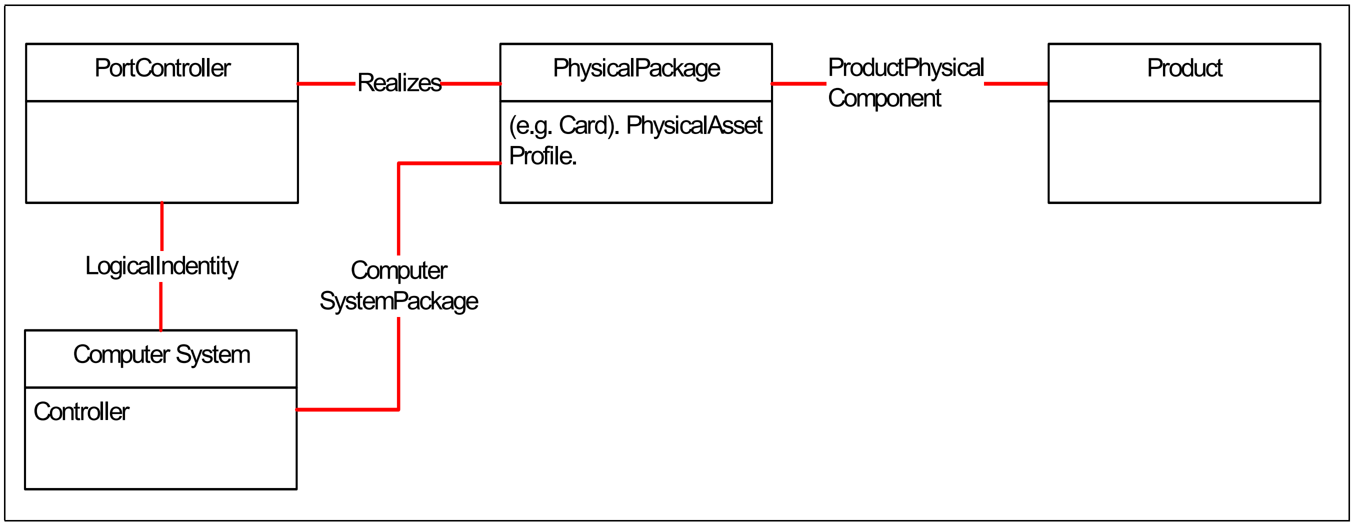

Figure 26 - Implementation of Physical Asset Profile

8.3.7 Implementation of Block Services Package

The Host Hardware RAID Controller profile shall follow all classes, properties and associations of the block services package with the following constraints:

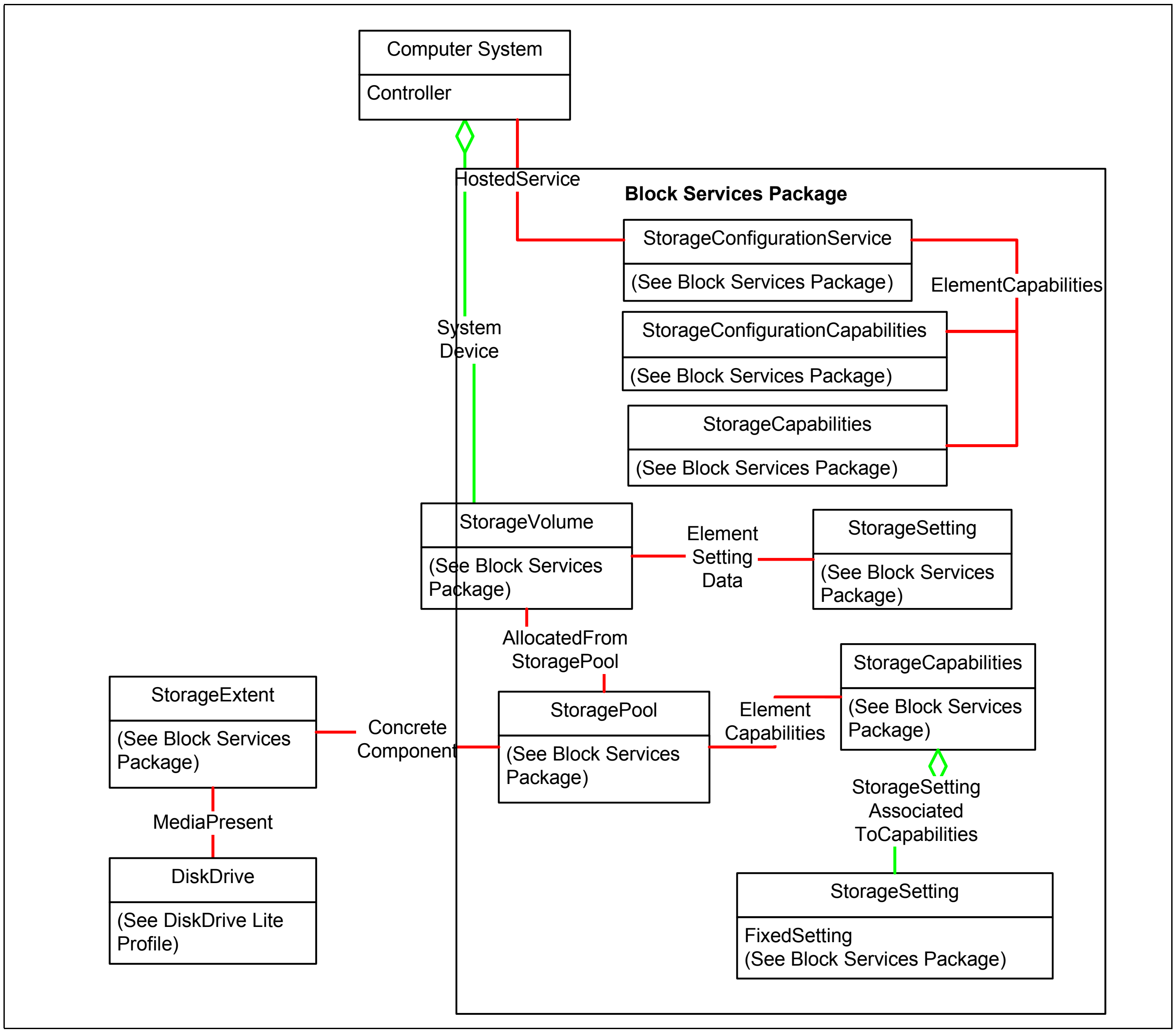
Computer System
Controller
HostedService
Block Services Package
StorageConfigurationService
(See Block Services Package)
ElementCapabilities
StorageConfigurationCapabilities
(See Block Services Package)
StorageCapabilities
(See Block Services Package)
System Device
StorageVolume
(See Block Services Package)
Element Setting Data
StorageSetting
(See Block Services Package)
AllocatedFrom StoragePool
StorageCapabilities
(See Block Services Package)
StorageExtent
(See Block Services Package)
Concrete Component
StoragePool
(See Block Services Package)
Element Capabilities
StorageSetting Associated ToCapabilities
MediaPresent
DiskDrive
(See DiskDrive Lite Profile)
StorageSetting
FixedSetting
(See Block Services Package)

Figure 27 - Block Services Package in Host Hardware RAID Controller

8.3.7.1 Storage Pools

8.3.7.1.1 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.

8.3.7.2 Storage Configuration Service and Storage Configuration Capabilities

Implementations shall instantiate a single instance of StorageConfigurationService and StorageConfigurationCapabilities as required by block services package. The StorageConfigurationService instance shall be associated to the ComputerSystem that represents the controller using the HostedService association.

8.3.7.3 Storage Configuration Capabilities

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 MOF definition for this profile, the following properties shall be limited to a subset of the values defined in the DMTF MOF files for the StorageConfigurationCapabilities class:

- SupportStorageElementTypes
  - Supported Values: SupportedStoragePoolFeatures = 2 | 3
  - Supported ValueMap: "StorageVolume", "StorageExtent"
- SupportedStoragePoolFeatures
  - Supported Values: SupportedStoragePoolFeatures = 2 | 3
  - Supported ValueMap: "InExtents", "Single InPools".
  - Note that the following values shall not be selected for this profile: 4- "Multiple InPools".
- SupportedStorageElementFeatures
  - Value: SupportedStoragePoolFeatures = 2|3|4|5|6|10
  - ValueMap: “StorageExtent Creation”, "StorageVolume Creation", "StorageExtent Modification", "StorageVolume Modification" or "Single InPool" or "InExtents".
  - Note that the following values shall not be selected for this profile: 2-"StorageExtents Creation", 3-"StorageExtents Modification", 8-"LogicalDisk Creation", 9-"LogicalDisk Modification", 7-"Multiple InPools".

8.3.7.4 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.

8.3.7.5 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.

8.3.8 Implementation of DAPort

For the Host Hardware RAID Controller profile, the Direct Attach Port profile models the port facing the host. The DAport class includes a property that defines the way the controller is connected to the host. The property shall be set to a valid port type as defined by the class. These include but are not limited to PCI, PCI-E, PCI express, and embedded. The ProtocolEndpoint shall be specialized to the protocol the host is using to communicate with the controller. This is normally SCSI but maybe others. The DAport also is associated to StorageVolumes (LogicalDevice) the controller makes visible to the host. The StorageVolumes shall be associated to the DAport ProtocolController with a ProtocolControllerForUnit association.

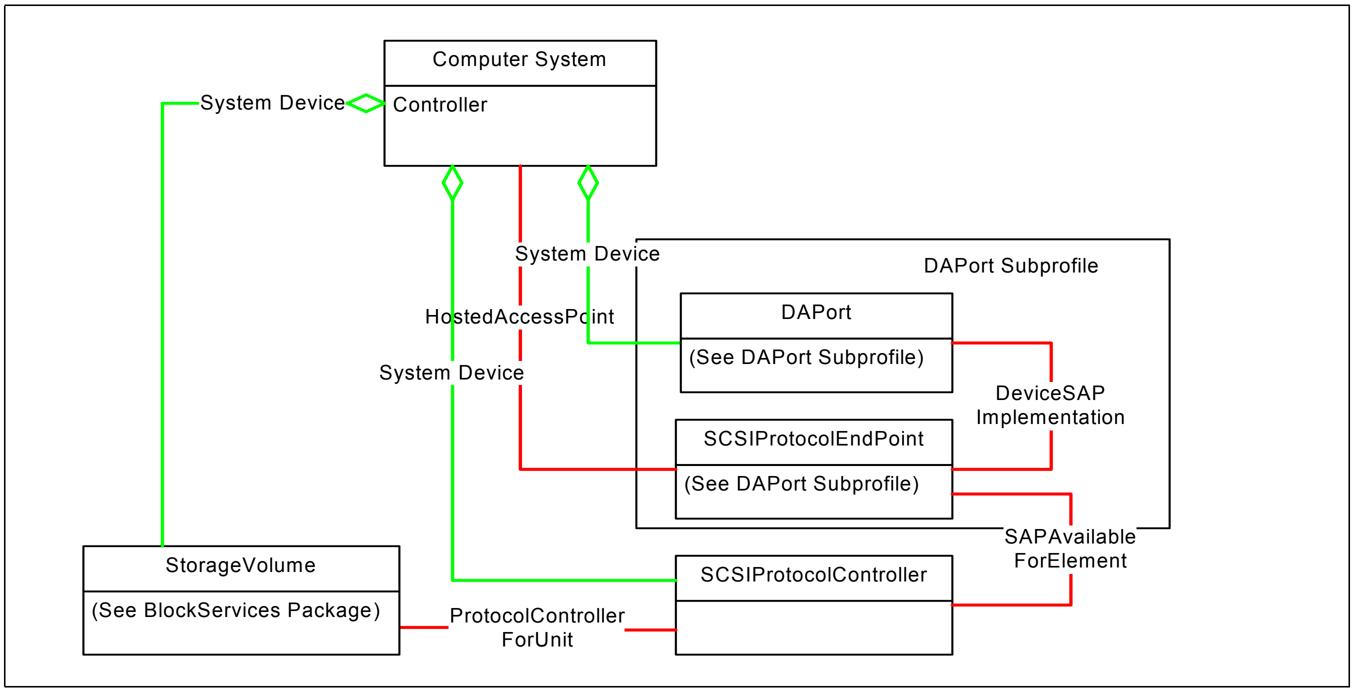

Figure 28 - DAPort Subprofile in Host Hardware Controller

8.3.9 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 differenciated by including the values Driver, Firmeware, or FCode/BIOS in the Classifications property.

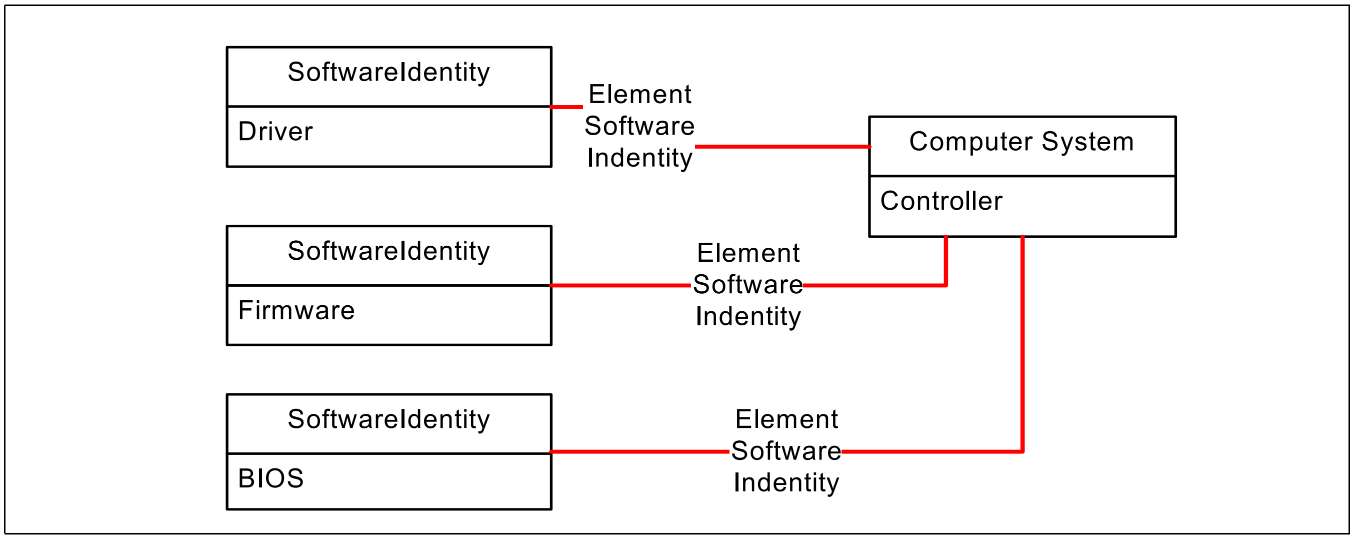

Figure 29 - Software Inventory Profile in Host Hardware RAID Controller

8.3.10 Implementation of Generic Initiator Ports Profile

The Host Hardware RAID Controller profile utilizes the Generic Initiator Ports Profile to model the back-end ports of the controller that are connected to the storage managed by the RAID controller.

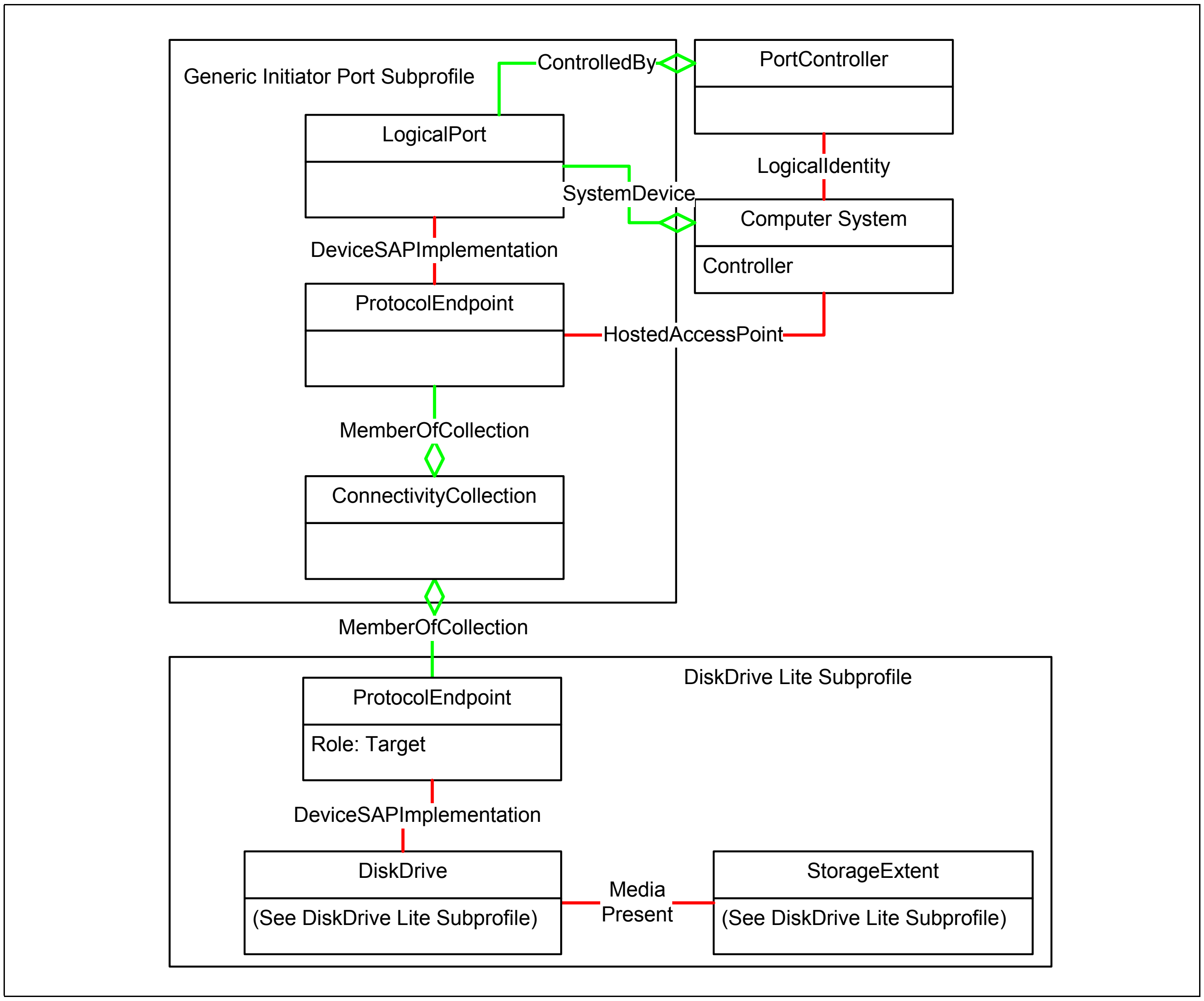

Figure 30 - Generic Initiator Port and Disk Drive Lite Subprofile

8.3.10.1 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, SATA, ATA, iSCSI or FC port depending on the controller implementation. Thus, the instance of this class shall be sub-classed to SPIPort, SASPort, iSCSIPort, 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.

8.3.10.2 CIM_ProtocolEndpoint

The ProtocolEndpoint class represents both ends of the logical data path between the controller 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 iSCSI protocol. Thus, the instance of the ProtocolEndpoint shall be subclassed to SCSIProtocolEndpoint or ATAProtocolEndpoint.

8.3.11 Implementation of Disk Drive Lite Profile and Connectivity to Storage Virtualizers

Many Host-based RAID controllers can be connected to set of internal disk drives, external JBOD or external virtualized storage. Therefore an implementation may use the Disk Drive Lite profile to model internal storage of the host or JBOD. The Disk Drive Lite profile may be used as a stand-alone profile or as a part of the Storage Device Enclosure component profile.

However, some host-based RAID controllers may be connected to another storage virtualizer, like a RAID Array. In this case, the implementation may use the Disk Drive Lite profile to model the storage imported from the storage virtualizer. The implementation may also model the storage from the virtualizer using a specialization of the Generic Initiator Port Subprofile and instantiating instances of StorageExtent class.

8.3.11.1 Primordial Storage Extents

In Host Hardware RAID Controller profile, a StorageExtent describes the storage space, capabilities and management of the media that exist to store and retrieve data. StorageExtents that have a "primordial" property set to "true" are used to represent the raw, unformatted logical extent on top of the physical media. A primordial StorageExtent shall be associated to only one primordial StoragePool using the CIM_ConcreteComponent association.

When the host-based RAID controller is connected to an internal set of disks or JBOD, the a primordial StorageExtents shall be associated to only one instance of CIM_DiskDrive that represents the physical media of a disk drive.

When the host-based RAID controller is connected to another storage virtualizer, the implementation shall instantiate an instance of StorageExtent to represent the storage devices imported to the host-based RAID controller. Each instance of StorageExtent shall have the "primordial" property set to "true" and shall be associated to the imported target Storage device use the LogicalIdentity association.

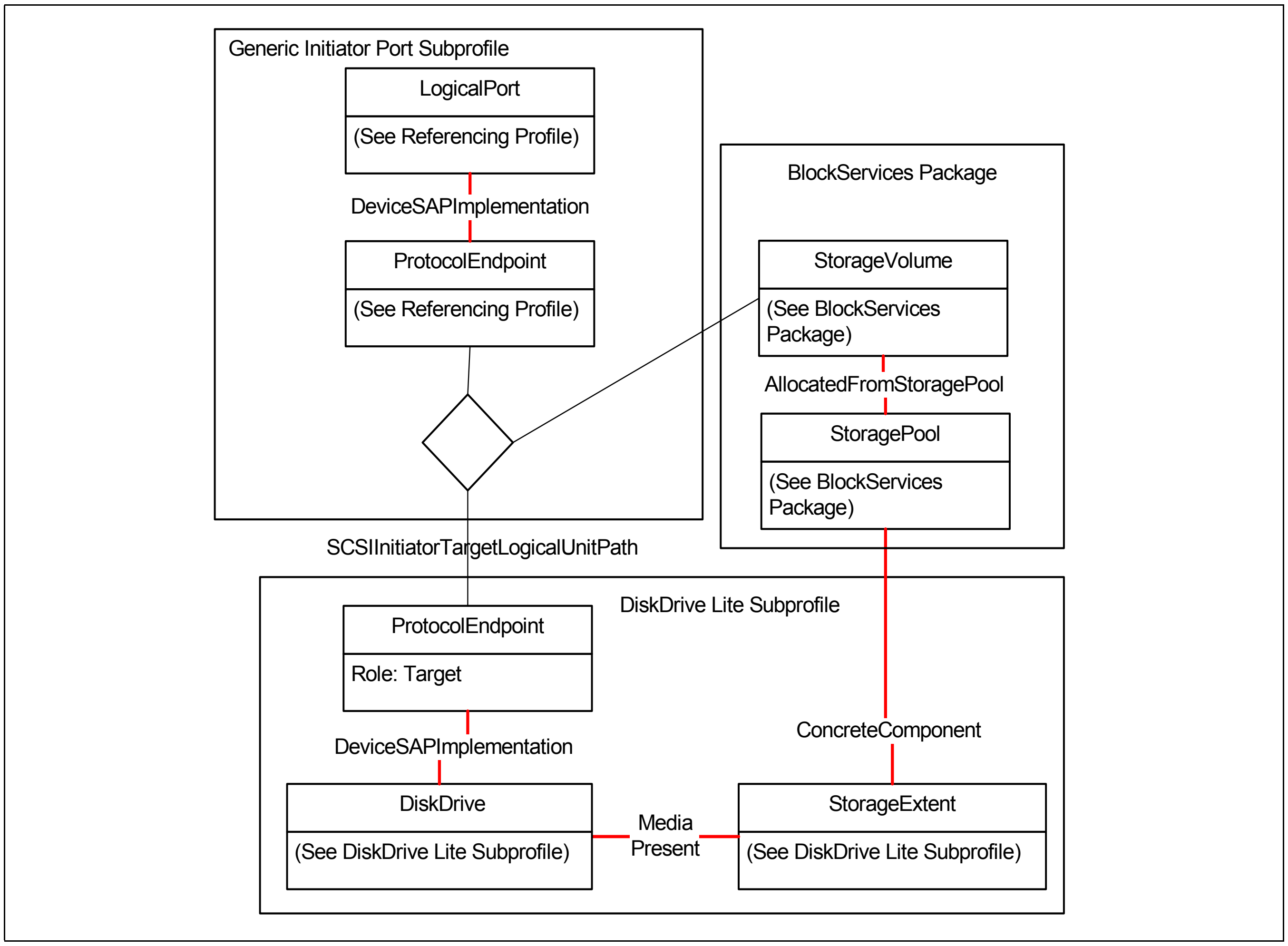
Generic Initiator Port Subprofile
LogicalPort
(See Referencing Profile)
DeviceSAPImplementation
ProtocolEndpoint
(See Referencing Profile)
BlockServices Package
StorageVolume
(See BlockServices Package)
AllocatedFromStoragePool
StoragePool
(See BlockServices Package)
SCSIInitiatorTargetLogicalUnitPath
DiskDrive Lite Subprofile
ProtocolEndpoint
Role: Target
DeviceSAPImplementation
ConcreteComponent
DiskDrive
(See DiskDrive Lite Subprofile)
Media Present
StorageExtent
(See DiskDrive Lite Subprofile)

Figure 31 - Host Hardware RAID Connected to a Storage Virtualizer

8.3.12 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 StorageVolumes 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. It is highly recommended that provider developers fully comprehend Extent Composition profile before implementing Extent Composition profile.

8.3.13 Disk Metadata Format

Many RAID controllers write metadata to the drives of the managed storage to hold configuration data about the layout of the data on the drives. If a host-based RAID controller uses meta-data on stored on the drives of the managed storage, the RAID controller should add a value to the DiskMetaDataFormat property on the CIM_StorageConfigurationService class with an enumeration value that best matches the description of the meta-data type.

Determination: A client or consumer can determine if a Host Hardware RAID Controller uses meta-data to manage the data format of the drives by reading the DiskMetaDataFormat property of the StorageConfigurationService instance associated to the ComputerSystem that represents the controller.

8.3.14 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.

8.3.15 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

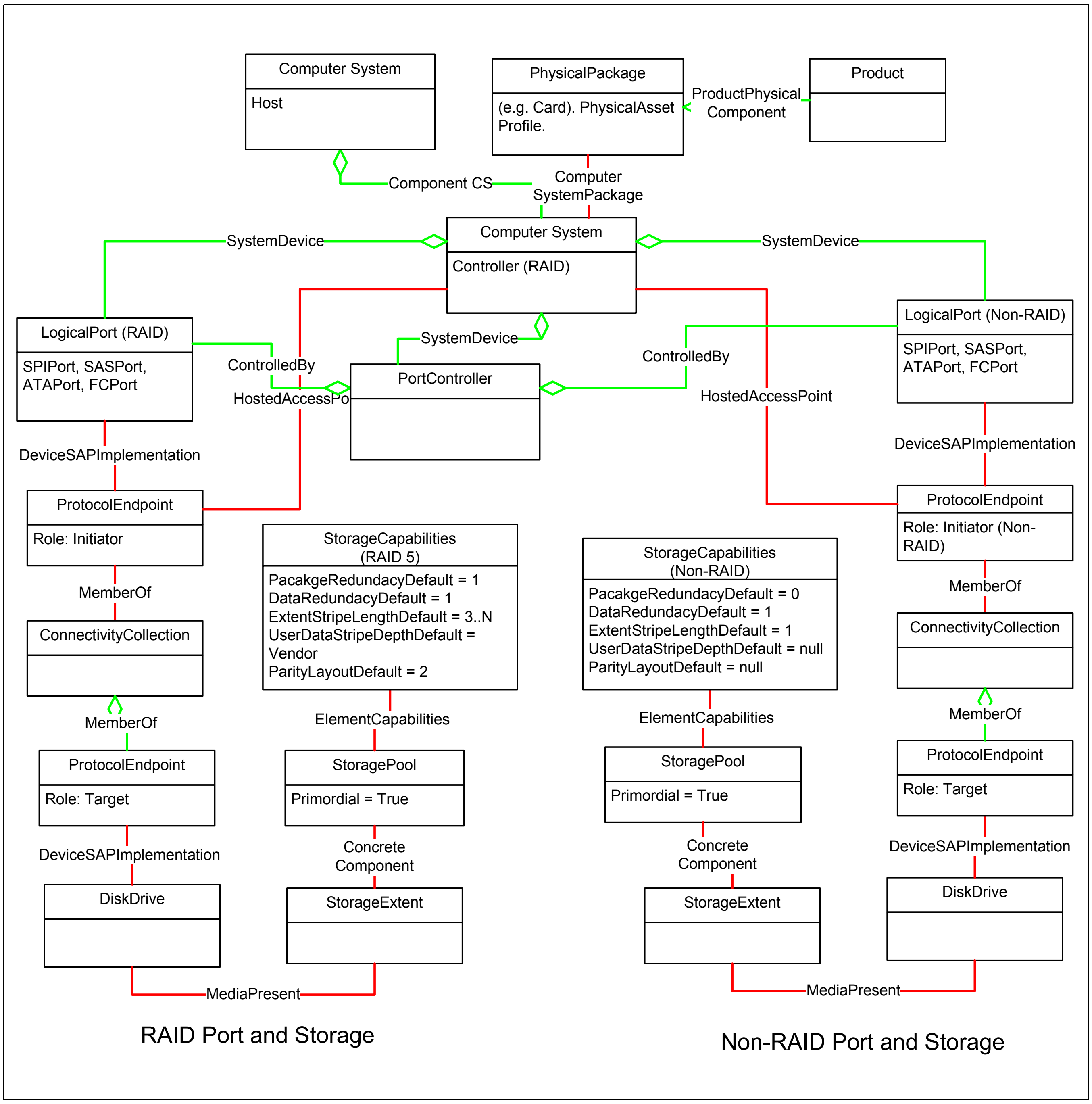

Figure 32 - Example of Mutli-Function Controllers

8.3.16 Health and Fault Management Consideration

8.3.17 Cascading Considerations

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 77 describes the CIM elements for Host Hardware RAID Controller.

Table 78 - 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	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_PortController	Mandatory	Serves as a component of the server ComputerSystem and is associated to the controller ComputerSystem.
8.6.8 CIM_Product	Mandatory	Asset information about the RAID controller.

Table 78 - CIM Elements for Host Hardware RAID Controller

Element Name	Requirement	Description
8.6.9 CIM_ProductPhysicalComponent	Mandatory	Associates Product and PhysicalPackage.
8.6.10 CIM_ProtocolController	Mandatory	Represents the target/device aspects of storage exported by the RAID controller.
8.6.11 CIM_ProtocolControllerForUnit	Mandatory	Associated ProtocolController to StorageVolume
8.6.12 CIM_Realizes (Associates PhysicalPackage to PortController)	Mandatory	Associates PortController and PhysicalPackage from the Physical Asset profile.
8.6.13 CIM_SAPAvailableForElement	Mandatory	Associates ProtocolController to the DAPort ProtocolEndpoint
8.6.14 CIM_SoftwareIdentity (Driver)	Mandatory	Override SoftwareIdentity from Software Inventory profile to assure Classifications property includes Driver.
8.6.15 CIM_SoftwareIdentity (FCode/BIOS)	Optional	Override SoftwareIdentity from Software Inventory profile to assure Classifications property includesFCODE/BIOS
8.6.16 CIM_SoftwareIdentity (Firmware)	Optional	Override SoftwareIdentity from Software Inventory profile to assure Classifications property includesFirmware
8.6.17 CIM_SystemComponent	Mandatory	Associates ComputerSystems representing the hosting system and the RAID controller.
8.6.18 CIM_SystemDevice (Associates Hosting System to PortController)	Mandatory	Associates Hosting System to PortController
8.6.19 CIM_SystemDevice (Associates System to AlarmDevice)	Optional	Associates System to AlarmDevice
8.6.20 CIM_SystemDevice (System to ProtocolController)	Mandatory	Links ProtocolController to the HHRC 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 79 describes class CIM_AlarmDevice.

Table 79 - SMI Referenced Properties/Methods for CIM_AlarmDevice

Properties	Requirement	Description & Notes
SystemCreationClassName	Mandatory	
SystemName	Mandatory	
CreationClassName	Mandatory	
DeviceID	Mandatory	
VisibleAlarm	Mandatory	shall be 'true'
Urgency	Mandatory	shall be 3 (Alternating)
SetAlarmState()	Mandatory	

8.6.2 CIM_AssociatedAlarm

Associates AlarmDevice and LogicalPort

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

Table 80 describes class CIM_AssociatedAlarm.

Table 80 - SMI Referenced Properties/Methods for CIM_AssociatedAlarm

Properties	Requirement	Description & Notes
Antecedent	Mandatory	
Dependent	Mandatory	

8.6.3 CIM_ComputerSystem

System that represents the Host Hardware RAID controller.

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

Table 81 describes class CIM_ComputerSystem.

Table 81 - SMI Referenced Properties/Methods for CIM_ComputerSystem

Properties	Requirement	Description & Notes
CreationClassName	Mandatory	
Name	Mandatory	Identifier for the Host Hardware RAID Controller

Table 81 - SMI Referenced Properties/Methods for CIM_ComputerSystem

Properties	Requirement	Description & Notes
NameFormat	Mandatory	Format for Name property. Shall be 'HID' for a hardware ID or 'Other'.
ElementName	Mandatory	User friendly name
Dedicated	Mandatory	Shall be 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 HHRC and not the hosting ComputerSystem

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 82 describes class CIM_ComputerSystemPackage (Associates PhysicalPackage to ComputerSystem).

Table 82 - 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).Validation Property : Limit to HHRC systems
Antecedent	Mandatory	

8.6.5 CIM_ControlledBy

Associates PortController to LogicalPorts

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 83 describes class CIM_ControlledBy.

Table 83 - 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 ComputerSystems representing the hosting system and the RAID controller.

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

Table 84 describes class CIM_LogicalIdentity.

Table 84 - SMI Referenced Properties/Methods for CIM_LogicalIdentity

Properties	Requirement	Description & Notes
SameElement	Mandatory	Reference to the ComputerSystem with Dedicated = 0 (Not dedicated)
SystemElement	Mandatory	Reference to the PortController

8.6.7 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 85 describes class CIM_PortController.

Table 85 - SMI Referenced Properties/Methods for CIM_PortController

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

Table 85 - SMI Referenced Properties/Methods for CIM_PortController

Properties	Requirement	Description & Notes
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.

8.6.8 CIM_Product

Asset information about the RAID controller.

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

Table 86 describes class CIM_Product.

Table 86 - 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.
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	

8.6.9 CIM_ProductPhysicalComponent

Associates Product and PhysicalPackage.

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

Table 87 describes class CIM_ProductPhysicalComponent.

Table 87 - SMI Referenced Properties/Methods for CIM_ProductPhysicalComponent

Properties	Requirement	Description & Notes
GroupComponent	Mandatory	
PartComponent	Mandatory	

8.6.10 CIM_ProtocolController

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

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

Table 88 describes class CIM_ProtocolController.

Table 88 - SMI Referenced Properties/Methods for CIM_ProtocolController

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

8.6.11 CIM_ProtocolControllerForUnit

Associated ProtocolController to StorageVolume

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

Table 89 describes class CIM_ProtocolControllerForUnit.

Table 89 - SMI Referenced Properties/Methods for CIM_ProtocolControllerForUnit

Properties	Requirement	Description & Notes
DeviceNumber	Optional	Required for SCSI devics, not used for ATA. 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

8.6.12 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 90 describes class CIM_Realizes (Associates PhysicalPackage to PortController).

Table 90 - SMI Referenced Properties/Methods for CIM_Realizes (Associates PhysicalPackage to PortController)

Properties	Requirement	Description & Notes
Dependent	Mandatory	
Antecedent	Mandatory	

8.6.13 CIM_SAPAvailableForElement

Associates ProtocolController to the DAPort ProtocolEndpoint

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

Table 91 describes class CIM_SAPAvailableForElement.

Table 91 - SMI Referenced Properties/Methods for CIM_SAPAvailableForElement

Properties	Requirement	Description & Notes
ManagedElement	Mandatory	
AvailableSAP	Mandatory	

8.6.14 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 be 2 (Driver)

8.6.15 CIM_SoftwareIdentity (FCode/BIOS)

Override SoftwareIdentity from Software Inventory profile to assure Classifications property includesFCODE/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 be 11 (FCODE/BIOS)

8.6.16 CIM_SoftwareIdentity (Firmware)

Override SoftwareIdentity from Software Inventory profile to assure Classifications property includesFirmware

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 be 10 (Firmware).

8.6.17 CIM_SystemComponent

Associates ComputerSystems representing the hosting system and the RAID controller.

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

Table 95 describes class CIM_SystemComponent.

Table 95 - SMI Referenced Properties/Methods for CIM_SystemComponent

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

8.6.18 CIM_SystemDevice (Associates Hosting System to PortController)

Associates Hosting System to PortController

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

Table 96 describes class CIM_SystemDevice (Associates Hosting System to PortController).

Table 96 - SMI Referenced Properties/Methods for CIM_SystemDevice (Associates Hosting System to PortController)

Properties	Requirement	Description & Notes
GroupComponent	Mandatory	Reference to host ComputerSystem with Dedicated = 0 (no dedicated)Validation Property : Limit to hosting systems
PartComponent	Mandatory	

8.6.19 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)Validation Property : Limit to HHRC systems
PartComponent	Mandatory	

8.6.20 CIM_SystemDevice (System to ProtocolController)

Links ProtocolController to the HHRC system.

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

Table 98 describes class CIM_SystemDevice (System to ProtocolController).

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

Properties	Requirement	Description & Notes
PartComponent	Mandatory	
GroupComponent	Mandatory	Reference to ComputerSystem with Dedicated = 30 (Host-based RAID controller)Validation Property : Limit to HHRC systems

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. Table 99 is a summary of some key iSCSI terms, their equivalent CIM classes, and definitions (from the IETF iSCSI RFC).

Table 99 - 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, 1.3.0 Rev 6*, Clause 16: iSCSI Initiator Port Profile] that includes classes (EthernetPort, iSCSIProtocolEndoint) that model SCSI ports and network portals.

Figure 33 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 the previous diagram.

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

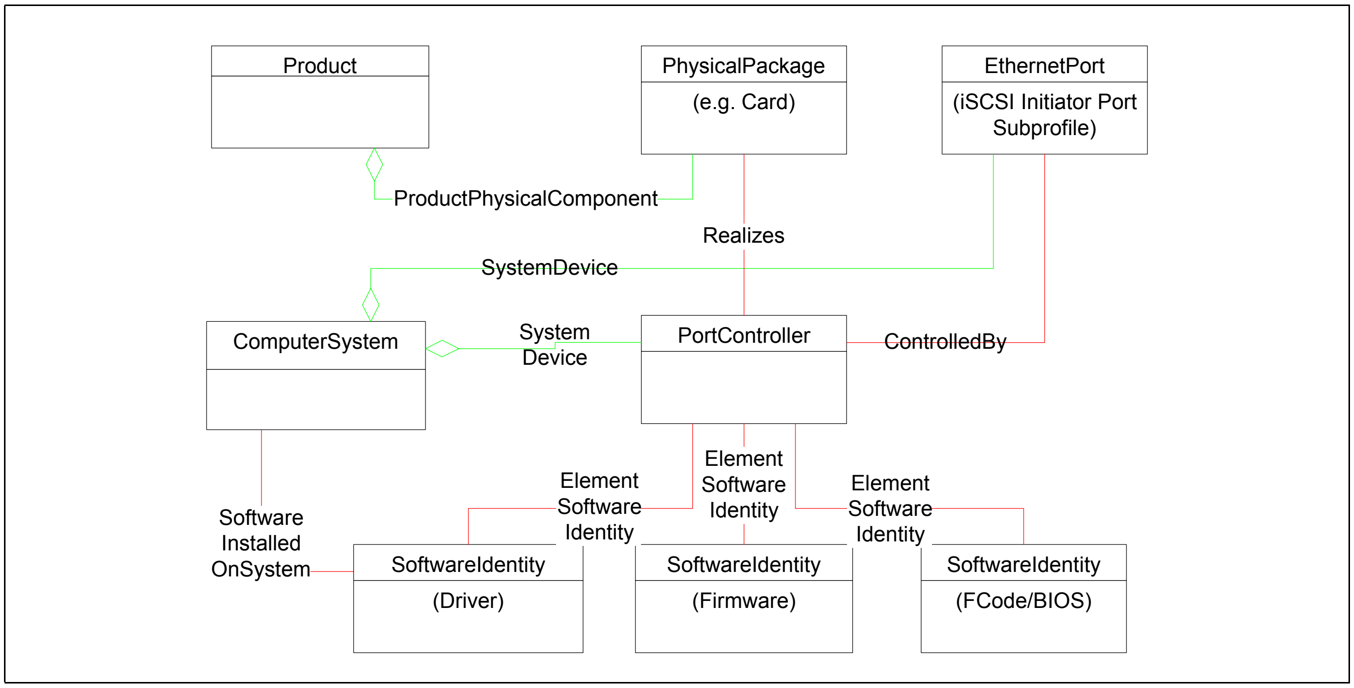

Figure 33 - iSCSI Product and Package Model

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.

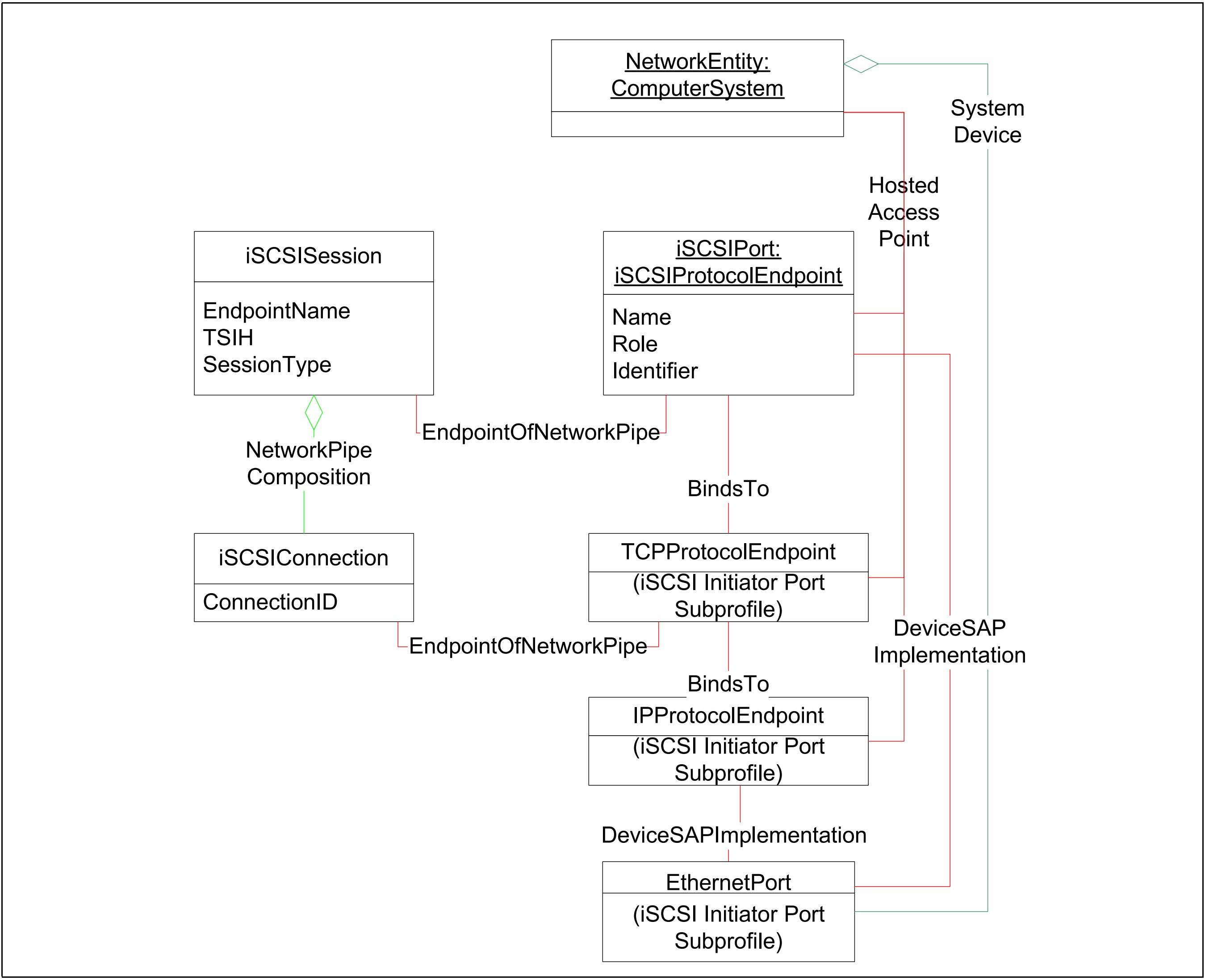

Figure 34 - iSCSI Sessions and Connections Model

There should be a single instance of SCSIProtocolController representing the initiator iSCSI node. This is associated via SystemDevice to the ComputerSystem. See Figure 33

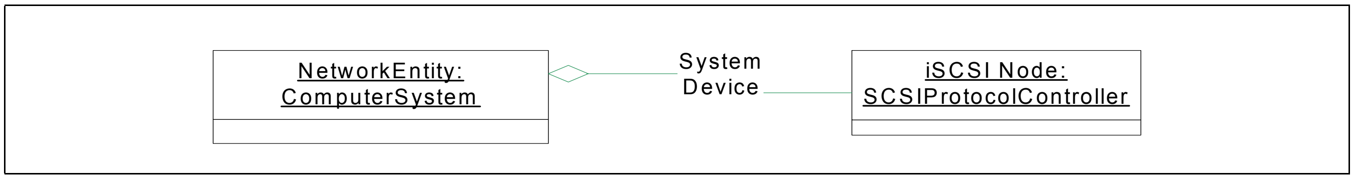
NetworkEntity: ComputerSystem
System Device
iSCSI Node: SCSIProtocolController

Figure 35 - 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, 1.3.0 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, 1.3.0 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, 1.3.0 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 100 defines the possible states that shall be supported for EthernetPort.OperationalStatus. The main OperationalStatus shall be the first element in the array

Table 100 - 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 101 - Supported Profiles for iSCSI Initiator

Registered Profile Names	Mandatory	Version
iSCSI Initiator Ports	No	1.2.0

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, 1.3.0 Rev 6*.

9.5.2 Find the health of an initiator

See 9.7.9 in *Storage Management Technical Specification, Part 2 Common Profiles, 1.3.0 Rev 6*.

9.5.3 Enable/disable header and data digest

See 9.5.3 in *Storage Management Technical Specification, Part 2 Common Profiles, 1.3.0 Rev 6*.

9.6 Registered Name and Version

iSCSI Initiator version 1.1.0

9.7 CIM Elements

Table 102 describes the CIM elements for iSCSI Initiator.

Table 102 - 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	Mandatory	
9.7.22 CIM_SystemDevice	Mandatory	
9.7.23 CIM_SystemDevice	Mandatory	

Table 102 - CIM Elements for iSCSI Initiator

Element Name	Requirement	Description
9.7.24 CIM_iSCSIConnection	Optional	
9.7.25 CIM_iSCSISession	Mandatory	
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_PortController	Optional	PortController (HBA) Creation
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_PorController	Optional	PortController (HBA) Removal

9.7.1 CIM_BindsTo (TCPProtocolEndpoint to IPProtocolEndpoint)

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

Table 103 describes class CIM_BindsTo (TCPProtocolEndpoint to IPProtocolEndpoint).

Table 103 - 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 104 describes class CIM_BindsTo (iSCSIProtocolEndpoint to TCPProtocolEndpoint).

Table 104 - SMI Referenced Properties/Methods for CIM_BindsTo (iSCSIProtocolEndpoint to TCPProtocolEndpoint)

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 105 describes class CIM_ComputerSystem.

Table 105 - 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 106 describes class CIM_ControlledBy.

Table 106 - 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 107 describes class CIM_DeviceSAPImplementation (EthernetPort to IPProtocolEndpoint).

Table 107 - 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 108 describes class CIM_DeviceSAPImplementation (EthernetPort to iSCSIProtocolEndpoint).

Table 108 - 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 109 describes class CIM_ElementSoftwareIdentity.

Table 109 - 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 110 describes class CIM_EndpointOfNetworkPipe (Between iSCSI Session and iSCSIProtocolENdpoint).

Table 110 - 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 111 describes class CIM_EndpointOfNetworkPipe (Between iSCSI connection and TCPProtocolENdpoint).

Table 111 - 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 112 describes class CIM_InstalledSoftwareIdentity.

Table 112 - 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 113 describes class CIM_NetworkPipeComposition.

Table 113 - 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 114 describes class CIM_PhysicalPackage.

Table 114 - 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 115 describes class CIM_PortController.

Table 115 - 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 116 describes class CIM_Product.

Table 116 - 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 117 describes class CIM_ProductPhysicalComponent.

Table 117 - 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 118 describes class CIM_ProtocolControllerForPort.

Table 118 - 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 119 describes class CIM_Realizes.

Table 119 - 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 120 describes class CIM_SAPAvailableForElement.

Table 120 - 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 121 describes class CIM_SCSIProtocolController.

Table 121 - SMI Referenced Properties/Methods for CIM_SCSIProtocolController

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

9.7.20 CIM_SoftwareIdentity

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

Table 122 describes class CIM_SoftwareIdentity.

Table 122 - 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

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

Table 123 describes class CIM_SystemDevice.

Table 123 - SMI Referenced Properties/Methods for CIM_SystemDevice

Properties	Flags	Requirement	Description & Notes
PartComponent		Mandatory	
GroupComponent		Mandatory	

9.7.22 CIM_SystemDevice

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

Table 124 describes class CIM_SystemDevice.

Table 124 - SMI Referenced Properties/Methods for CIM_SystemDevice

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	
PartComponent		Mandatory	

9.7.23 CIM_SystemDevice

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

Table 125 describes class CIM_SystemDevice.

Table 125 - SMI Referenced Properties/Methods for CIM_SystemDevice

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 126 describes class CIM_iSCSIConnection.

Table 126 - 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 126 - 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 127 describes class CIM_iSCSISession.

Table 127 - 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 127 - SMI Referenced Properties/Methods for CIM_iSCSISession

Properties	Flags	Requirement	Description & Notes
MaxConnectionsPer Session		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.

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 36.

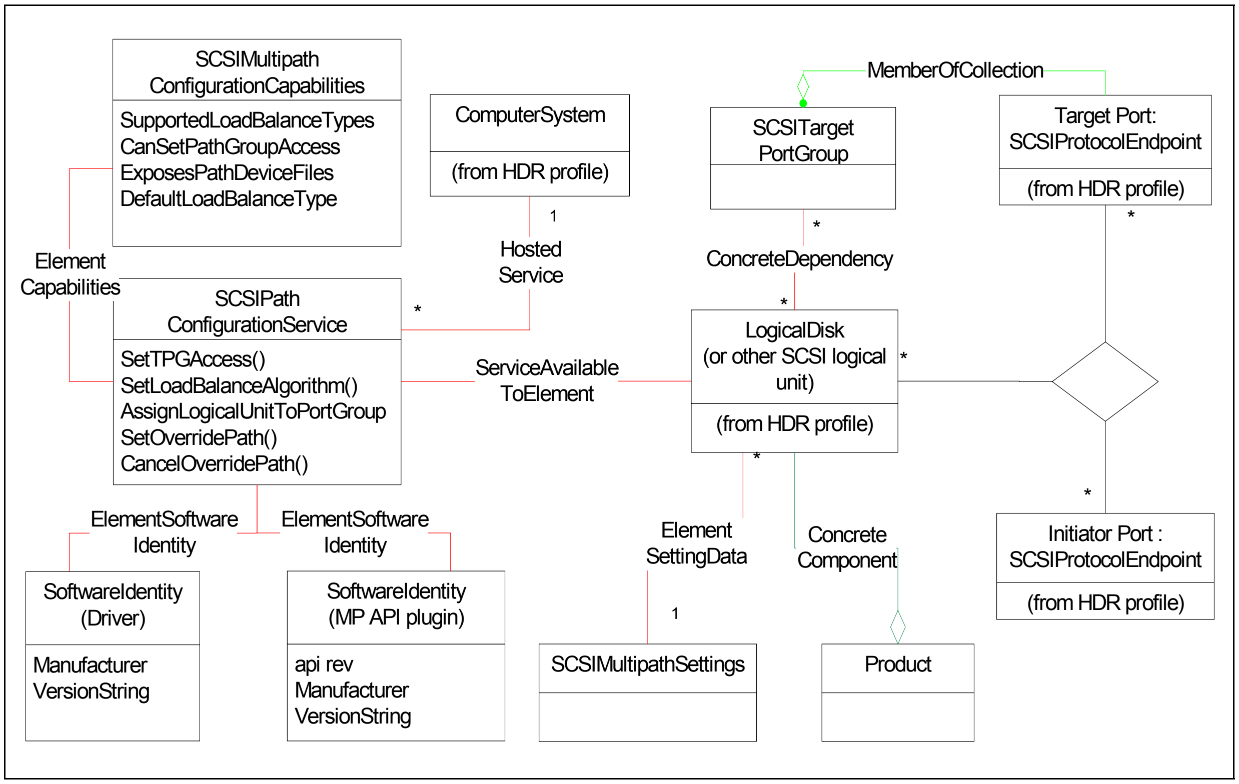

Figure 36 - Multipath Management Class Diagram

.Figure 37 shows the relationship of target and initiator ports (SCSIProtocolEndpoint instances) and a disk

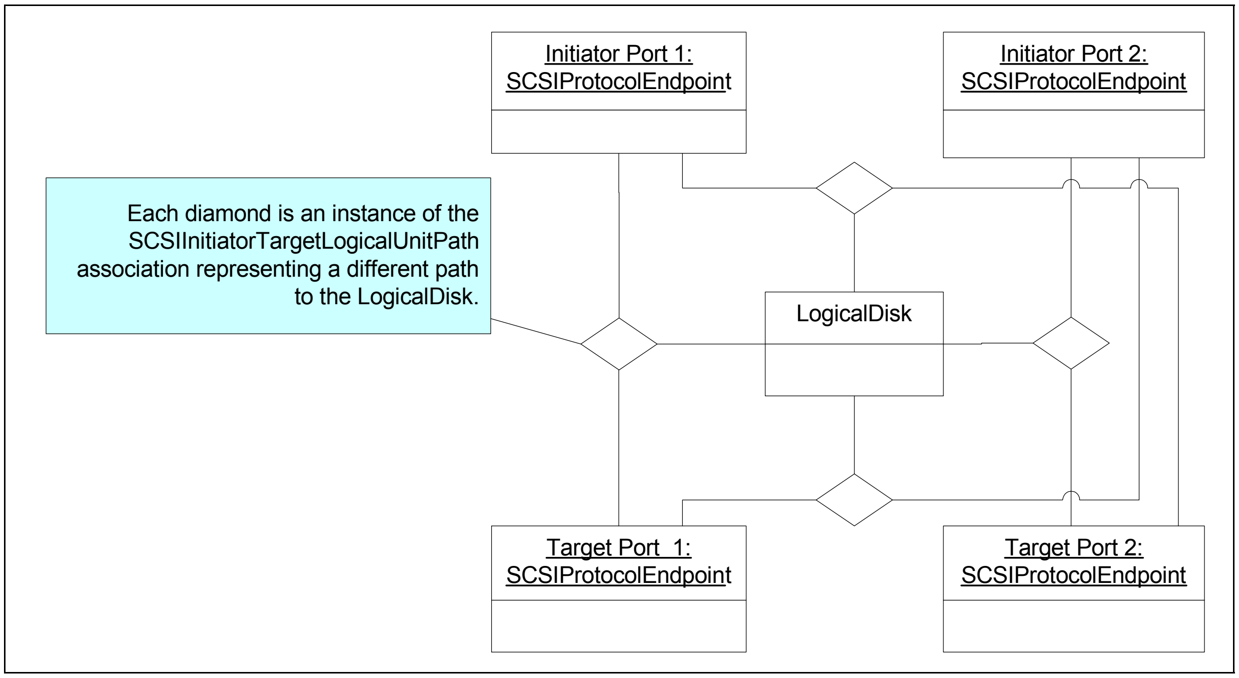

Figure 37 - Four Path Instance Diagram

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

In the past, devices exposed vendor-specific SCSI multipath interfaces, so drivers with device-specific logic were shipped with target devices, logical volume managers, and HBAs. The SPC-3 have 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 the 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 of 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 SCSIMultipathConfigurationCapabilities SupportsLuAssignment is set. The parameters are:

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

10.5.4 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 SCSIInitiatorTargetLogicalUnitPath.

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 SCSIInitiatorTargetLogicalUnitPath.

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 proptocol 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 iniitiator
             // 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 locad 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 128 describes the CIM elements for SCSI Multipath Management.

Table 128 - CIM Elements for SCSI Multipath Management

Element Name	Requirement	Description
10.8.1 CIM_SCSIInitiatorTargetLogicalUnitPath	Mandatory	Associates initiator and target SCSIProtocolEndpoints to a SCSI logical unit (LogicalDevice)
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_SCSIInitiatorTargetLogicalUnitPath	Mandatory	Path creation
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_SCSIInitiatorTargetLogicalUnitPath	Mandatory	Path deletion

Table 128 - CIM Elements for SCSI Multipath Management

Element Name	Requirement	Description
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_SCSIInitiatorTargetLogicalUnitPath AND SourceInstance.State <> PreviousInstance.State	Mandatory	Path State change
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_SCSIInitiatorTargetLogicalUnitPath AND SourceInstance.AdministrativeWeight <> PreviousInstance.AdministrativeWeight	Mandatory	Path AdminsitrativeWeight change
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_SCSIInitiatorTargetLogicalUnitPath AND SourceInstance.AdministrativeOverride<> PreviousInstance.AdministrativeOverride	Mandatory	Path AdministrativeOverride change

10.8.1 CIM_SCSIInitiatorTargetLogicalUnitPath

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

Table 129 describes class CIM_SCSIInitiatorTargetLogicalUnitPath.

Table 129 - 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

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 38 shows the relationship of target and initiator ports.

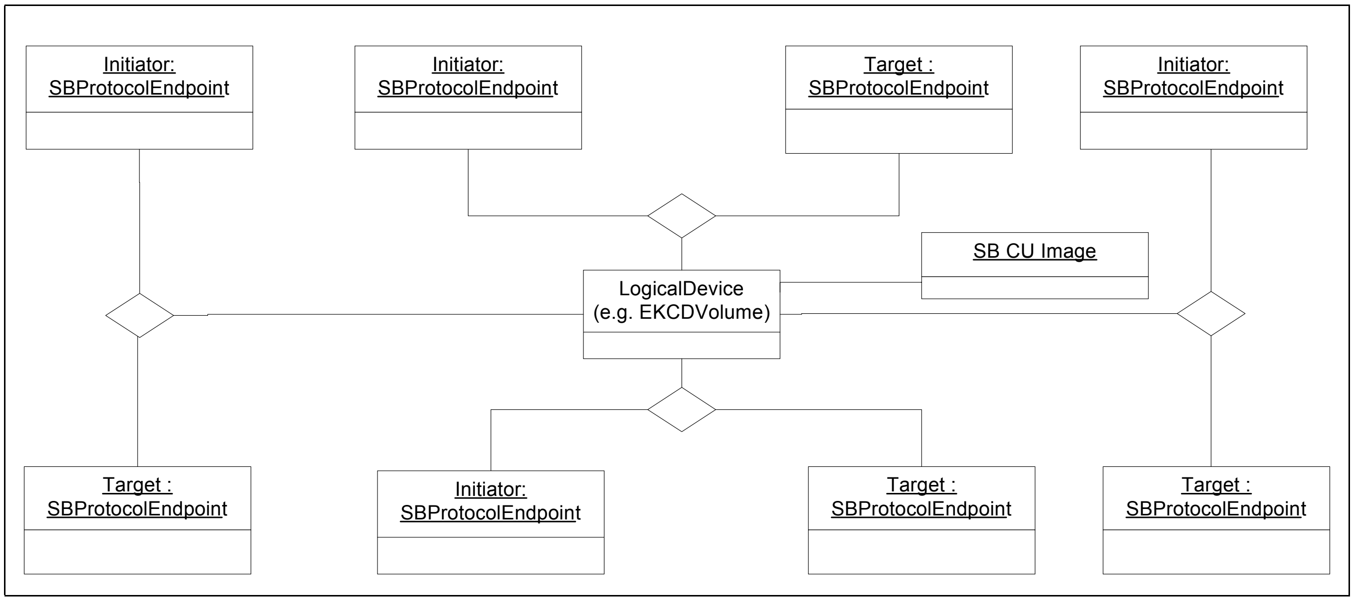
Initiator: SBProtocolEndpoint
Initiator: SBProtocolEndpoint
Target : SBProtocolEndpoint
Initiator: SBProtocolEndpoint
SB CU Image
LogicalDevice (e.g. EKCDVolume)
Target : SBProtocolEndpoint
Initiator: SBProtocolEndpoint
Target : SBProtocolEndpoint
Target : SBProtocolEndpoint

Figure 38 - Four SB Channel Instance Diagram

Figure 39 shows the relationship of multiple initiator ports to multiple target ports with multiple devices in a single control unit image.

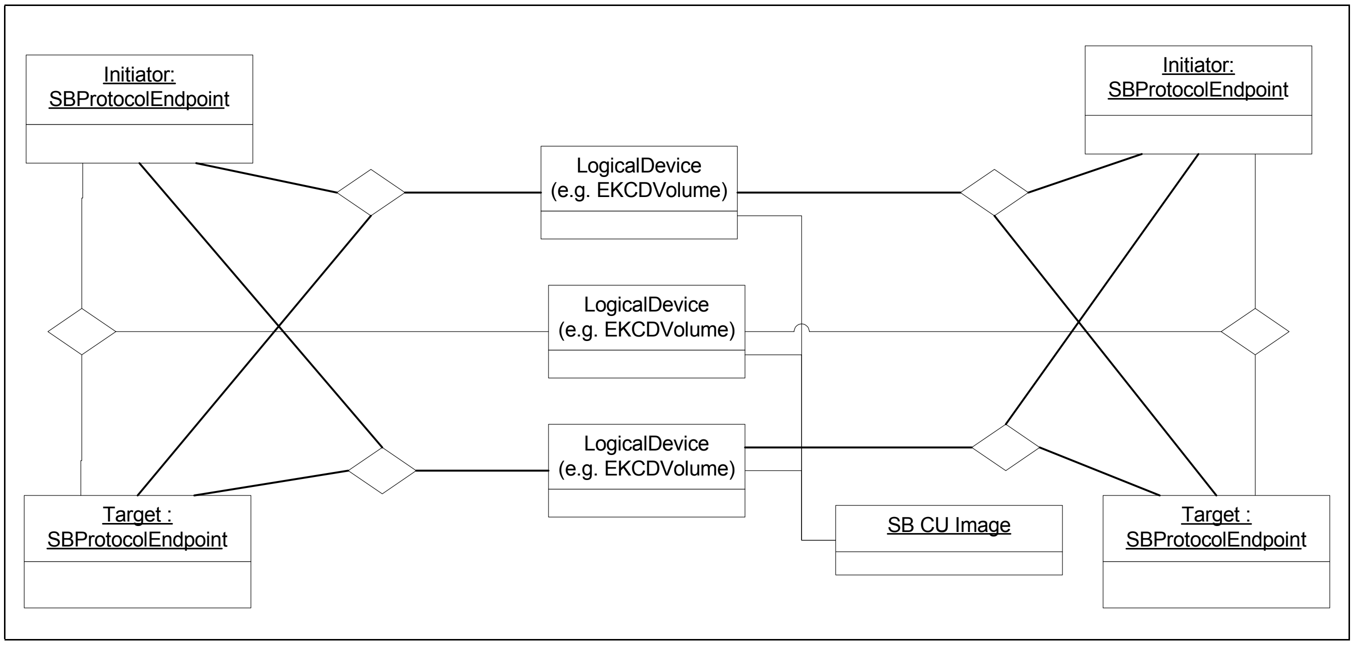
Initiator: SBProtocolEndpoint
LogicalDevice (e.g. EKCDVolume)
Initiator: SBProtocolEndpoint
LogicalDevice (e.g. EKCDVolume)
LogicalDevice (e.g. EKCDVolume)
Target : SBProtocolEndpoint
SB CU Image
Target : SBProtocolEndpoint

Figure 39 - 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 130 describes the CIM elements for SB Multipath Management.

Table 130 - 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 131 describes class SNIA_SBInitiatorTargetLogicalUnitPath.

Table 131 - 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