INTERNATIONAL STANDARD

ISO/IEC 24775

First edition
2007-06

Information technology – Storage management



Reference number
ISO/IEC 24775:2007(E)

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale



THIS PUBLICATION IS COPYRIGHT PROTECTED

Copyright © 2007 ISO/IEC, Geneva, Switzerland

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either IEC or IEC's member National Committee in the country of the requester.

If you have any questions about ISO/IEC copyright or have an enquiry about obtaining additional rights to this publication, please contact the address below or your local IEC member National Committee for further information.

IEC Central Office
3, rue de Varembé
CH-1211 Geneva 20
Switzerland
Email: inmail@iec.ch
Web: www.iec.ch

About the IEC

The International Electrotechnical Commission (IEC) is the leading global organization that prepares and publishes International Standards for all electrical, electronic and related technologies.

About IEC publications

The technical content of IEC publications is kept under constant review by the IEC. Please make sure that you have the latest edition, a corrigenda or an amendment might have been published.

- Catalogue of IEC publications: www.iec.ch/searchpub
The IEC on-line Catalogue enables you to search by a variety of criteria (reference number, text, technical committee,…). It also gives information on projects, withdrawn and replaced publications.

- IEC Just Published: www.iec.ch/online_news/justpub
Stay up to date on all new IEC publications. Just Published details twice a month all new publications released. Available on-line and also by email.

- Customer Service Centre: www.iec.ch/webstore/custserv
If you wish to give us your feedback on this publication or need further assistance, please visit the Customer Service Centre FAQ or contact us:

Email: csc@iec.ch
Tel.: +41 22 919 02 11
Fax: +41 22 919 03 00

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

INTERNATIONAL STANDARD

ISO/IEC 24775

First edition
2007-06

Information technology – Storage management



PRICE CODE XM

For price, see current catalogue

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

FOREWORD 22
INTRODUCTION 23
1 Scope 24
2 Normative References 25
2.1 International standards 25
2.2 Other specifications 25
3 Definitions, symbols, abbreviations, and conventions 26
3.1 Definitions 26
3.1.1 34
3.2 Symbols and abbreviations 40
3.3 Keywords 40
3.4 Conventions 41
4 Introduction 43
4.1 Preamble 43
4.2 Business rationale 43
4.3 Interface definition 43
4.4 Technology trends 45
4.5 Management environment 47
4.6 Architectural objectives 48
5 Overview 50
5.1 Base capabilities 50
5.1.1 General 50
5.1.2 Object Oriented 50
5.1.3 Messaging-based interface 52
5.2 Capabilities of this version 54
5.2.1 Overview 54
5.2.2 Determine and monitor the configuration of a SAN 54
5.2.3 Monitoring the health of key resources in a SAN 55
5.2.4 Monitoring the available performance of interconnections in a SAN 55
5.2.5 Monitoring and controlling the zones in a SAN 55
5.2.6 Discovering/monitoring/controlling the storage volumes in a SAN 55
5.2.7 Requiring authenticated clients in a SAN 55
5.3 Operational environment 56
5.4 Using this standard 57
5.5 Language bindings 57
6 Transport and reference model 58
6.1 Introduction 58
6.1.1 Overview 58
6.1.2 Language requirements 58
6.1.3 Communications requirements 58
6.1.4 XML Message Syntax and Semantics 58
6.2 Transport stack 59

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

6.3 Reference model 60
6.3.1 Overview....60
6.3.2 Roles for Interface Constituents 60
6.3.2.1 Client 60
6.3.2.2 Agent 60
6.3.2.3 CIM Server 61
6.3.2.4 Provider 61
6.3.2.5 Lock Manager....61
6.3.2.6 Directory Server....61
6.3.3 Cascaded Agents 61
7 Object model 62
7.1 Model overview (key resources) 62
7.1.1 Overview....62
7.1.2 Introduction to CIM UML Notation 62
7.2 Techniques 63
7.2.1 CIM fundamentals 63
7.2.2 Modelling profiles....66
7.2.3 Naming 67
7.2.4 Correlatable and Durable Names 68
7.2.4.1 Overview....68
7.2.4.2 Correlatable Names Formation....70
7.2.4.3 Testing equality of Correlatable Names....70
7.2.4.4 Standard Formats for Correlatable Names....70
7.2.4.5 Case Sensitivity 72
7.2.4.6 Preferred Correlatable Names....72
7.2.4.7 Concatenation 72
7.2.5 Events – CIM indications 73
7.2.5.1 Background 73
7.2.5.2 Using indications 73
7.2.5.3 Indication hierarchy 75
7.2.5.4 Agent/Provider Considerations....76
7.2.5.5 Client Considerations 77
7.2.5.6 Requirements 78
7.2.5.7 Implementation Considerations 78
7.2.6 Device Credentials 78
7.2.7 Recipe Conventions 79
7.2.7.1 Recipe Definition 79
7.2.7.2 Recipe Pseudo Code Conventions....79
7.2.7.3 Common Recipes 84
7.3 Profiles....85
7.3.1 Profile content....85
7.3.1.1 Profile and subprofile definition 85
7.3.1.2 Format for Profile Specifications....85
7.3.1.3 Registry of Profiles and Subprofiles 87
7.3.2 Common CIM Packages....89
7.3.2.1 Description....89
7.3.2.2 Physical Package “package” 89

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.2.3 Software Package 95
7.3.3 Common subprofiles....99
7.3.3.1 Overview....99
7.3.3.2 Access Points subprofile 99
7.3.3.3 Cluster Subprofile 103
7.3.3.4 Disk Drive subprofile....108
7.3.3.5 Extent Mapping subprofile 120
7.3.3.6 Location subprofile 124
7.3.3.7 Software Subprofile 127
7.3.3.8 Copy Services Subprofile 128
7.3.3.9 Job Control Subprofile 154
7.3.3.10 Pool Manipulation Capabilities and Settings Subprofile 160
7.3.3.11 LUN Creation Subprofile....183
7.3.3.12 Device Credentials subprofile....203
7.3.3.13 Backend Ports subprofile....208
7.3.3.14 LUN Masking and Mapping 215
7.3.4 Fabric....248
7.3.4.1 Fabric Profile 248
7.3.4.2 Switch Profile....297
7.3.5 Hosts 311
7.3.5.1 FC HBA Profile 311
7.3.6 Storage....325
7.3.6.1 Array Profile....325
7.3.6.2 In-Band Virtualization Profile 347
7.3.6.3 Storage Library Profile....362
7.3.7 Server Profile....384
7.3.7.1 Description....384
7.3.7.2 Standard dependencies....384
7.3.7.3 Profile dependencies....384
7.3.7.4 CIM Server requirements 385
7.3.7.5 Instance Diagram 386
7.3.7.6 Durable names and other correlatable IDs....387
7.3.7.7 Methods....387
7.3.7.8 Client considerations 388
7.3.7.9 Recipes....388
7.3.7.10 Instrumentation requirements....397
7.3.7.11 Required CIM Elements 399
7.3.7.12 Required properties for CIM Elements 399
7.3.7.13 Optional Subprofiles and Profiles. 406
7.3.7.14 ObjectManagerAdapter Subprofile 406
7.4 Cross Client considerations....410
7.4.1 Overview....410
7.4.1.1 HBA model 410
7.4.1.2 Switch model 411
7.4.1.3 Array model 411
7.4.1.4 Out of band virtualization model 413
7.4.1.5 Durable names 413

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.4.1.6 Fabric Topology (HBA, Switch, Array) 414
7.4.1.7 Storage Connections (FC HBA, Array) 420
7.4.1.8 Zoning 420
7.4.1.9 Fabric Route Discovery 421
7.4.1.10 Durable names 421
7.4.2 General recipes 421
7.4.2.1 Indications' Status 421
7.4.2.2 Listenable Instance Notification 422
7.4.2.3 Life Cycle Event Subscription description 422
7.4.2.4 Subscription for alert indications 423
7.4.2.5 Listenable Interface Modification Notification 423
7.4.2.6 Subscribe for Lifecycle Events where OperationalStatus Changes 423
7.4.2.7 Subscription for alert indications 424
8 Security 425
8.1 Introduction 425
8.2 Background 425
8.3 Requirements 426
8.3.1 General 426
8.3.2 Certificate usage with SSL 3.0 and TLS 426
8.3.2.1 Functional goals 426
8.3.2.2 Requirements 427
8.4 Modelling device credentials 428
8.4.1 General requirements 428
8.4.2 Instrumentation requirements 428
9 Service Discovery 429
9.1 Objectives 429
9.2 Overview 429
9.3 SLP messages 430
9.4 Scopes 431
9.5 Services definition 432
9.5.1 General 432
9.5.2 Service Type 432
9.5.3 Service Attributes 432
9.6 User Agents (UA) 434
9.7 Service Agents (SAs) 434
9.8 Directory Agents (DAs) 434
9.9 Service Agent Server (SA Server) 435
9.9.1 General Information 435
9.9.2 SA Server (SAS) Implementation 435
9.9.3 SA Server (SAS) Clients 435
9.9.3.1 Description 435
9.9.3.2 SAS Client Requests – SA Server Responses 436
9.9.4 SA Server configuration 436
9.9.4.1 Overview 436
9.9.4.2 SLP Configuration File 436
9.9.4.3 Programmatic configuration 436

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

9.9.4.4 DHCP configuration 437
9.9.4.5 Scope 437
9.9.5 SA Server discovery 437
9.9.6 SAS Client registration 437
9.10 'Standard WBEM' Service Type Templates 438
10 SMI-S Roles 442
10.1 Introduction 442
10.2 SMI-S Client 443
10.2.1 Overview 443
10.2.2 SLP functions 443
10.2.3 CIM-XML protocol functions 443
10.2.4 Security considerations 443
10.2.5 Lock management functions 443
10.3 Dedicated SMI-S Server 443
10.3.1 Overview 443
10.3.2 SLP functions 444
10.3.3 CIM-XML Protocol Functions 444
10.3.3.1 General 444
10.3.3.2 Required intrinsic methods 444
10.3.3.3 Required model support 445
10.3.4 Security considerations 445
10.3.5 Lock management functions 445
10.4 General Purpose SMI-S Server 445
10.4.1 Overview 445
10.4.2 SLP functions 445
10.4.3 CIM-XML Protocol functions 445
10.4.3.1 General 445
10.4.3.2 Required intrinsic methods 445
10.4.3.3 Required model support 446
10.4.3.4 Security considerations 446
10.4.4 Lock management functions 446
10.4.5 Provider Subrole 446
10.4.5.1 Overview 446
10.4.5.2 Required model support 446
10.5 Directory Server 446
10.5.1 General 446
10.5.2 SLP functions 446
10.5.3 CIM-XML Protocol functions 446
10.5.4 Security considerations 446
10.5.5 Lock management functions 446
10.6 Combined Roles on a Single System 446
10.6.1 Overview 446
10.6.2 General Purpose SMI-S Server as a Profile Aggregator 447
10.6.2.1 SLP functions 447
10.6.2.2 CIM-XML Protocol functions 447
10.6.2.3 Security considerations 447
10.6.2.4 Lock management functions 447

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

11 Installation and upgrade 448
11.1 Introduction 448
11.2 Role of the Administrator 448
11.3 Goals 448
11.3.1 Non-Disruptive Installation and De-installation 448
11.3.2 Plug-and-Play 448
11.4 Installing device support 449
11.4.1 General 449
11.4.2 Installation 449
11.4.3 Discovery and initialization of Device support 449
11.4.4 Removal/Update 450
11.4.5 Reconfiguration 450
11.4.6 Failure 451
11.5 Object manager 451
11.5.1 Installation 451
11.5.2 Multiple CIMOMs on a Single Server 451
11.5.3 Removal, reconfiguration and upgrade 452
11.5.4 Failure 452
11.6 Client 452
11.6.1 Removal 452
11.6.2 Reconfiguration 452
11.6.3 Failure 452
11.7 Directory Server 452
11.7.1 Installation 452
11.7.2 Removal/Failure 452
11.8 Management domains 452
11.8.1 General 452
11.8.2 Initial configuration 452
11.8.3 Reconfiguration 453
11.9 Lock manager 453
Annex A(informative) Futures 454
A.1 Overview 454
A.2 HBA LUN masking and persistent binding 454
A.3 Managed Hub Section 454
A.4 IP Storage 454
A.5 Multi-Path Modelling 454
A.6 Provider Modelling 454
A.7 Non-Fibre Fabrics 454
A.8 Compliance notification 454
A.9 Cascaded Agents 455
A.10 Network Storage 455
A.11 Synchronization of file system elements through copy services 455
A.12 Model size distinctions in Disk Drive 455
A.13 Expanded extent mapping 455
A.14 Locking 455
A.15 Policy management 456
Annex B(informative) Experimental Profiles 459

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

B.1 Overview 459
B.2 Common Profiles and Subprofiles....459
B.3 SML Subprofiles....462
B.4 Extender profile 498
B.5 Management Appliance profile....513
B.6 Out of Band Virtualizer Profile....521
B.7 JBOD Profile 546
B.8 Extra Capacity Set Subprofile 557
B.9 Host Discovered Resources Profile (EXPERIMENTAL) 563
B.10 Router Profile 584
Annex C(informative) Mapping CIM Objects to SNMP MIB Structures....601
C.1 Purpose of this Annex 601
C.2 CIM-to-MIB Mapping Overview 601
C.3 CIM-to-MIB mapping methodology 601
C.4 Example mapping 604
Annex D(normative) Compliance with this standard 622
D.1 Compliance statement 622
D.2 Compliance declaration....622
D.3 Server Profile and compliance 622
D.4 Example 622
Annex E(informative) Optional Profiles and Subprofiles 624
E.1 Introduction 624
E.2 Provider Subprofile....624
Bibliography 631

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table 1 – Standard dependencies for SMI-S....25
Table 2 – SLP properties....66
Table 3 – Standardized Name Formats....71
Table 4 – Profile components....86
Table 5 – Registry of profiles and subprofiles....88
Table 6 – Required CIM Elements....92
Table 7 – Required Properties for SystemPackaging....93
Table 8 – Required Properties for PhysicalPackage....93
Table 9 – Required Properties for Product....94
Table 10 – Required Properties for ProductPhysicalComponent....94
Table 11 – Required Properties for Container....95
Table 12 – Required Properties for ProductParentChild....95
Table 13 – Required Properties for Realizes....95
Table 14 – Required CIM Elements....98
Table 15 – Required Properties for InstalledSoftwareIdentity....98
Table 16 – Required Properties for SoftwareIdentity....99
Table 17 – RemoteAccessPoint.InfoFormat and AccessInfo Properties....101
Table 18 – Required CIM Elements....102
Table 19 – Required Properties for HostedAccessPoint....102
Table 20 – Required Properties for SAPAvailableForElement....103
Table 21 – Required Properties for RemoteServiceAccessPoint....103
Table 22 – Optional Profiles or Subprofiles....103
Table 23 – OperationStatus for Component ComputerSystem....106
Table 24 – Required CIM Elements....107
Table 25 – Required Properties for ComponentCS....107
Table 26 – Required Properties for ComputerSystem....108
Table 27 – Optional Profiles or Subprofiles....108
Table 28 – Required Functional Profiles....109
Table 29 – DiskDrive Status....111
Table 30 – Required CIM Elements....114
Table 31 – Required Properties for BasedOn....115
Table 32 – Required Properties for ConcreteComponent....115
Table 33 – Required Properties for Container....116
Table 34 – Required Properties for ProductParentChild....116
Table 35 – Required Properties for ElementSoftwareIdentity....116
Table 36 – Required Properties for DiskDrive....117
Table 37 – Required Properties for MediaPresent....117
Table 38 – Required Properties for PhysicalMedia....118
Table 39 – Required Properties for Realizes....118
Table 40 – Required Properties for RealizesExtent....119
Table 41 – Required Properties for SoftwareIdentity....119
Table 42 – Required properties for StorageExtent....120
Table 43 – Optional profiles or subprofiles....120
Table 44 – Required Functional Profiles....121
Table 45 – Required CIM Elements....123

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table 46 – Required Properties for BasedOn 123
Table 47 – Required Properties for ConcreteComponent 123
Table 48 – Required Properties for StorageExtent 124
Table 49 – Optional Profiles or Subprofiles 124
Table 50 – Required CIM Elements 126
Table 51 – Required Properties of Location 126
Table 52 – Required Properties for PhysicalElementLocation 127
Table 53 – Optional Profiles or Subprofiles 127
Table 54 – Required CIM Elements 128
Table 55 – Optional Profiles or Subprofiles 128
Table 56 – Copy Services Standard Dependencies 129
Table 57 – Required Functional Profiles 129
Table 58 – Name Formats 132
Table 59 – Subprofile Required Classes, Associations, Methods and Indications 146
Table 60 – Required Properties for ElementCapabilities 147
Table 61 – Required Properties for HostedService 147
Table 62 – Required Properties for StorageConfigurationService 148
Table 63 – Required Properties for StorageConfigurationCapabilities 148
Table 64 – Required Properties for StorageSynchronized 149
Table 65 – Required Properties for StorageCapabilities 150
Table 66 – Required Properties for ElementSettingData 153
Table 67 – Required Properties for StorageSetting 153
Table 68 – Copy Services Optional Subprofiles and Profiles 154
Table 69 – Job Control Services Standard Dependencies 155
Table 70 – Required Functional Profiles 155
Table 71 – Subprofile Required Classes, Associations, Methods and Indications 158
Table 72 – AffectedJobElement Required Properties 158
Table 73 – Required Properties for ConcreteJob 159
Table 74 – Required Properties for OwningJobElement 160
Table 75 – Optional Profiles or Subprofiles 160
Table 76 – Pool Manipulation Capabilities and Settings Standard Dependencies 161
Table 77 – Required Functional Profiles 162
Table 78 – Example RAID Mapping Table 169
Table 79 – Required CIM Elements 173
Table 80 – Required Properties for ElementCapabilities 174
Table 81 – Required Properties for StorageConfigurationService 174
Table 82 – Required Properties for StorageConfigurationCapabilities 174
Table 83 – Required Properties for StorageCapabilities 176
Table 84 – Required Properties for ElementSettingData 179
Table 85 – Required Properties for StorageSetting 179
Table 86 – Required Properties for StorageSettingWithHints 181
Table 87 – HostedService Required Properties 183
Table 88 – Optional Profiles or Subprofiles 183
Table 89 – LUN Creation Standard Dependencies 184

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table 90 – Required Functional Profiles 184
Table 91 – Required CIM Elements 202
Table 92 – Required Properties for StorageConfigurationService 202
Table 93 – Optional Profiles or Subprofiles 203
Table 94 – Device Credentials Standard Dependencies 203
Table 95 – Required Functional Profiles 204
Table 96 – Required CIM Elements 206
Table 97 – Required Properties for SharedSecretService 206
Table 98 – Required Properties for SharedSecret 206
Table 99 – SharedSecretIsShared Required Properties 207
Table 100 – HostedService Required Properties 207
Table 101 – Optional Profiles or Subprofiles 208
Table 102 – Device Credentials Standard Dependencies 208
Table 103 – Required Functional Profiles 209
Table 104 – Required CIM Elements 211
Table 105 – Required Properties for FCPort 211
Table 106 – Required Properties from ProtocolControllerForPort 213
Table 107 – Required Properties from ProtocolControllerAccessesUnit 213
Table 108 – Required Properties for SCSIProtocolController 213
Table 109 – Required Properties for StorageExtent 214
Table 110 – Required Properties for SystemDevice 214
Table 111 – Optional Profiles or Subprofiles 214
Table 112 – LUN Masking Standard Dependencies 215
Table 113 – Required Functional Profiles 216
Table 114 – Subprofile Required Classes, Associations, Methods and Indications 239
Table 115 – Required Properties for AuthorizedPrivilege 240
Table 116 – Required Properties for AuthorizedSubject 241
Table 117 – Required Properties for AuthorizedTarget 241
Table 118 – Required Properties for ConcreteDependency 241
Table 119 – Required Properties for ControllerConfigurationService 241
Table 120 – Required Properties for ElementSettingData 242
Table 121 – Required Properties for HostedCollection 242
Table 122 – Required Properties for HostedDependency 242
Table 123 – Required Properties for ProtoclControllerMaskingCapabilities 243
Table 124 – Required Properties for PrivilegeManagementService 245
Table 125 – Required Properties for StorageClientSettingData 246
Table 126 – Required Properties for StorageHardwareID 246
Table 127 – Required Properties for StorageHardwareIDManagementService 247
Table 128 – Required Properties for SystemSpecificCollection 247
Table 129 – Fabric Standard Dependencies 250
Table 130 – Required Functional Profiles 250
Table 131 – Durable Names Usage 254
Table 132 – Port OperationalStatus 254
Table 133 – OperationalStatus for ComputerSystem 254
Table 134 – Required CIM Elements 257
Table 135 – Required Properties for ActiveConnection 259

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table 136 – Required Properties for AdminDomain 259
Table 137 – Required Properties for Component 259
Table 138 – Required Properties for ComputerSystem 260
Table 139 – Required Properties for ContainedDomain 260
Table 140 – Required Properties for DeviceSAPImplementation 260
Table 141 – Required Properties for ElementCapabilities 261
Table 142 – Required Properties for ElementSettingData 261
Table 143 – Required Properties for FCPort 261
Table 144 – HostedAccessPoint 264
Table 145 – Required Properties for HostedCollection 264
Table 146 – Required Properties for ConnectivityCollection 264
Table 147 – Required Properties for LogicalPortGroup 264
Table 148 – Required Properties for MemberOfCollection 265
Table 149 – Required Properties for MemberOfCollection 265
Table 150 – Required Properties for ProtocolEndpoint 266
Table 151 – Required Properties for SystemDevice 266
Table 152 – Required Properties for Zone 266
Table 153 – Required Properties for ZoneCapabilities 267
Table 154 – Required Properties for ZoneMembershipSettingData 268
Table 155 – Required Properties for ZoneSet 268
Table 156 – Optional Profiles or Subprofiles 268
Table 157 – Required Functional Profiles 269
Table 158 – Required CIM Elements 283
Table 159 – Required Properties for HostedService 283
Table 160 – Required Properties for ZoneService 283
Table 161 – Optional Profiles or Subprofiles 284
Table 162 – Required CIM Elements 289
Table 163 – Required Properties for HostedCollection 289
Table 164 – Required Properties of MemberOfCollection 289
Table 165 – Required Properties for NamedAddressCollection 289
Table 166 – Required Properties for ZoneService 290
Table 167 – Optional Profiles or Subprofiles 290
Table 168 – Required CIM Elements 292
Table 169 – Required Properties for ControlledBy 292
Table 170 – Required Properties for ElementSoftwareIdentity 293
Table 171 – Required Properties for FCPort 293
Table 172 – Required Properties for LogicalPortGroup 295
Table 173 – Required Properties of MemberOfCollection 295
Table 174 – Required Properties for PortController 295
Table 175 – Required Properties of ProtocolControllerForPort 296
Table 176 – Required Properties for SCSIProtocolController 296
Table 177 – Switch Standard Dependencies 297
Table 178 – Required Functional Profiles 297
Table 179 – Required CIM Elements 300
Table 180 – Required Properties for ComputerSystem 300
Table 181 – Required Properties for ElementStatisticalData 302
Table 182 – Required Properties for FCPort 302

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table 183 – Required Properties for FCPortRateStatistics....304
Table 184 – Required Properties for FCPortStatistics....304
Table 185 – Required Properties for SystemDevice....306
Table 186 – Optional Profiles or Subprofiles....306
Table 187 – Required CIM Elements....309
Table 188 – Required Properties for LogicalModule....309
Table 189 – Required Properties for ModulePort....310
Table 190 – Required Properties for SystemDevice....310
Table 191 – Optional Profiles or Subprofiles....310
Table 192 – HBA Standard Dependencies....311
Table 193 – Required Functional Profiles....311
Table 194 – Required CIM Elements....316
Table 195 – Required Properties for ComputerSystem....317
Table 196 – Required Properties for ControlledBy....317
Table 197 – Required Properties for ElementSoftwareIdentity....317
Table 198 – Required Properties for ElementStatisticalData....317
Table 199 – Required Properties for FCPort....318
Table 200 – Required Properties for FCPortStatistics....319
Table 201 – Required Properties for HostedCollection....319
Table 202 – Required Properties for InstalledSoftwareIdentity....320
Table 203 – Required Properties for LogicalPortGroup....320
Table 204 – Required Properties of MemberOfCollection....320
Table 205 – Required Properties for PhysicalPackage....320
Table 206 – Required Properties for PortController....321
Table 207 – Required Properties for Product....321
Table 208 – Required Properties for ProductPhysicalComponent....322
Table 209 – Required Properties of ProtocolControllerForPort....322
Table 210 – Required Properties for Realizes....322
Table 211 – Required Properties for SCSIProtocolController....323
Table 212 – Required Properties for SystemDevice....323
Table 213 – Required Properties for SoftwareIdentity....323
Table 214 – Optional Profiles or Subprofiles....324
Table 215 – Array Standard Dependencies....325
Table 216 – Required Functional Profiles....326
Table 217 – OperationalStatus for ComputerSystem....329
Table 218 – OperationalStatus for StorageVolume....330
Table 219 – Port State/Status....331
Table 220 – Required CIM Elements....335
Table 221 – Required Properties for AllocatedFromStoragePool....336
Table 222 – Required Properties for ElementCapabilities....336
Table 223 – Required Properties for ElementSettingData....336
Table 224 – Required Properties for ComputerSystem....336
Table 225 – Required Properties for FCPort....338
Table 226 – Required Properties from HostedStoragePool....340
Table 227 – Required Properties from ProtocolControllerForPort....340
Table 228 – Required Properties from ProtocolControllerForUnit....340
Table 229 – Required Properties for SCSIProtocolController....340

--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table 230 – Required Properties from StorageCapabilities....342
Table 231 – Required Properties for StoragePool....342
Table 232 – Required Properties from StorageSetting....343
Table 233 – Required Properties for StorageVolume....343
Table 234 – Required Properties for SystemDevice....345
Table 235 – Optional Profiles or Subprofiles....346
Table 236 – In-Band Virtualizer Standard Dependencies....347
Table 237 – Required Functional Profiles....347
Table 238 – Required CIM Elements....351
Table 239 – Required Properties for AllocatedFromStoragePool....353
Table 240 – Required properties for ComputerSystem....353
Table 241 – Required properties for ConcreteComponent....353
Table 242 – Required properties for ElementCapabilities....354
Table 243 – Required properties for ElementSettingData....354
Table 244 – Required properties for FCPort....354
Table 245 – Required properties from HostedStoragePool....356
Table 246 – Required properties for ProtocolControllerAccessesUnit....356
Table 247 – Required Properties from ProtocolControllerForPort....357
Table 248 – Required Properties from ProtocolControllerForUnit....357
Table 249 – Required properties for SCSIProtocolController....357
Table 250 – Required properties from StorageCapabilities....358
Table 251 – Required properties for StorageExtent....358
Table 252 – Required Properties for StoragePool....359
Table 253 – Required properties from StorageSetting....359
Table 254 – Required properties for StorageVolume....359
Table 255 – Required properties for SystemDevice....361
Table 256 – Optional Profiles or Subprofiles....361
Table 257 – Storage Library Standard Dependencies....362
Table 258 – Required Functional Profiles....363
Table 259 – Required CIM Elements....371
Table 260 – Required Properties for ChangerDevice....373
Table 261 – Required properties for Chassis....374
Table 262 – Required properties for Container....374
Table 263 – Required properties for ProtocolControllerForUnit....375
Table 264 – Required properties for SCSIProtocolController....375
Table 265 – Required properties for DeviceSoftware....375
Table 266 – Required properties for LibraryPackage....376
Table 267 – Required properties for MediaAccessDevice....376
Table 268 – Required properties for PackagedComponent....376
Table 269 – Required properties for PhysicalMedia....377
Table 270 – Required properties for PhysicalMediaInLocation....377
Table 271 – Required properties for ProductPhysicalComponent....377
Table 272 – Required properties for Realizes....378
Table 273 – Required properties for SoftwareIdentity....378
Table 274 – Required Properties for StorageLibrary....379
Table 275 – Required Properties for StorageMediaLocation....379
Table 276 – Required properties for SystemDevice....380

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table 277 – Optional Profiles or Subprofiles 380
Table 278 – Required CIM Elements 382
Table 279 – Required Properties for LimitedAccessPort 382
Table 280 – Optional Profiles or Subprofiles 383
Table 281 – CIM Server Standard Dependencies 384
Table 282 – Required Functional Profiles 385
Table 283 – Profile Required Classes, Associations, Methods and Indications 399
Table 284 – Required Properties for ObjectManager 400
Table 285 – Required Properties for System 400
Table 286 – Required Properties for HostedService 401
Table 287 – Required Properties for CIMXMLCommunicationMechanism.....401
Table 288 – Required Properties for HostedAccessPoint.....402
Table 289 – Required Properties for CommMechanismForManager 402
Table 290 – Required Properties for Namespace.....403
Table 291 – Required Properties for NamespaceInManager.....403
Table 292 – Required Properties for RegisteredProfile 404
Table 293 – Required Properties for RegisteredSubProfile.....404
Table 294 – Required Properties for ReferencedProfile 405
Table 295 – Required Properties for SubProfileRequiresProfile.....405
Table 296 – Required Properties for ElementConformsToProfile.....406
Table 297 – CIM Server Profile Optional Subprofiles and Profiles 406
Table 298 – Subprofile Required Classes, Associations, Methods and Indications.....408
Table 299 – Required Properties for ObjectManagerAdapter.....408
Table 300 – Required Properties for CommMechanismForObjectManagerAdapter 409
Table 301 – Cross-Profile Durable Names 413
Table 302 – Cross Profile Durable Names 421
Table 303 – Message Types.....431
Table 304 – Required Configuration Properties for SA as DA.....436
Table 305 – Required Configuration Properties for SA.....436
Table 306 – Functional Profiles 444
Table B.1 – Required CIM Elements 461
Table B.2 – Required Properties for IsSpare 461
Table B.3 – Required Properties of MemberOfCollection.....461
Table B.4 – Required Properties for SparedSet 461
Table B.5 – Optional Profiles or Subprofiles 462
Table B.6 – Required CIM Elements 464
Table B.7 – Required Properties for InterLibraryPort.....464
Table B.8 – Required Properties for LibraryExchange 464
Table B.9 – Optional Profiles or Subprofiles 465
Table B.10 – Required CIM Elements 467
Table B.11 – Required Properties for DeviceServicesLocation 467
Table B.12 – Optional Profiles or Subprofiles 467
Table B.13 – Required CIM Elements 470
Table B.14 – Required Properties for FCPort 470
Table B.15 – Required Properties for ProtocolControllerForPort.....472

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table B.16 – Required CIM Elements 474
Table B.17 – Required Properties for DeviceServicesLocation 474
Table B.18 – Optional Profiles or Subprofiles 474
Table B.19 – Required CIM Elements 476
Table B.20 – Required Properties for ConfigurationCapacity 476
Table B.21 – Required Properties for ElementCapacity 476
Table B.22 – Optional Profiles or Subprofiles 476
Table B.23 – Required Properties for AlertIndication.....478
Table B.24 – LibraryAlert Property Settings.....478
Table B.25 – Vendor Specific Properties of LibraryAlert.....479
Table B.26 – Variable Alert Properties for LibraryAlert 479
Table B.27 – SCSI TapeAlert-based Properties 479
Table B.28 – LibraryAlert AlertIndication Properties 480
Table B.29 – Optional Profiles or Subprofiles 497
Table B.30 – Extender Standard Dependencies.....498
Table B.31 – Required Functional Profiles 499
Table B.32 – Required CIM Elements 502
Table B.33 – Required Properties for ActiveConnection for ATM.....504
Table B.34 – Required Properties for ActiveConnection for FC.....504
Table B.35 – Required Properties for BindsTo 504
Table B.36 – Required Properties for ComputerSystem.....504
Table B.37 – Required Properties for DeviceSAPImplementation.....506
Table B.38 – Required Properties for FCPort 506
Table B.39 – Required Properties for ForwardingService.....508
Table B.40 – Required Properties for ForwardsAmong 508
Table B.41 – Required Properties for HostedNetworkPipe.....508
Table B.42 – Requited Properties for IPProtocolEndpoint.....508
Table B.43 – Required Properties for NetworkPort.....510
Table B.44 – Required Properties for Network 511
Table B.45 – Required Properties for NetworkPipe 511
Table B.46 – Requited Properties for ProtocolEndpoint 511
Table B.47 – Requited Properties for TCPPrototocolEndpoint 511
Table B.48 – Required Properties for SystemDevice.....512
Table B.49 – Optional Profiles or Subprofiles 512
Table B.50 – Management Appliance Standard Dependencies 513
Table B.51 – Required Functional Profiles 513
Table B.52 – Required CIM Elements 516
Table B.53 – Required Properties for ComputerSystem.....516
Table B.54 – Required Properties for FCPort 517
Table B.55 – Required Properties from HostedService 518
Table B.56 – Required Properties from Installed SoftwareElement.....518
Table B.57 – Required Properties for LogicalPortGroup 518
Table B.58 – Required Properties for MemberOfCollection.....518
Table B.59 – Required Properties for RemoteServiceAccessPoint 519
Table B.60 – Required Properties for ServiceAvailableToElement.....519
Table B.61 – Required Properties for SoftwareElement 519
Table B.62 – Optional Profiles or Subprofiles 520

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table B.63 – OutofBand Virtualizer Standard Dependencies 522
Table B.64 – Required Functional Profiles 523
Table B.65 – Required CIM Elements 529
Table B.66 – Required Properties for AllocatedFromStoragePool.....531
Table B.67 – Required Properties for BasedOn 531
Table B.68 – Required Properties for Component.....531
Table B.69 – Required Properties for ComputerSystem - Metadata Controller .531
Table B.70 – Required Properties for ComputerSystem - Translation Engine...532
Table B.71 – Required Properties for ElementCapabilities.....532
Table B.72 – Required Properties for FCPort 533
Table B.73 – Required Properties from HostedCollection 534
Table B.74 – Required Properties from HostedStoragePool 534
Table B.75 – Required Properties from LogicalPortGroup.....534
Table B.76 – Required Properties for MemberOfCollection.....535
Table B.77 – Required Properties from ProtocolControllerForPort.....535
Table B.78 – Required Properties from ProtocolControllerForUnit.....535
Table B.79 – Required Properties for SCSIProtocolController 535
Table B.80 – Required Properties from StorageCapabilities 536
Table B.81 – Required Properties for StoragePool.....537
Table B.82 – Required Properties from StorageSetting.....537
Table B.83 – Required Properties for StorageExtent.....537
Table B.84 – Required Properties for StorageVolume.....542
Table B.85 – Required Properties for SystemDevice.....542
Table B.86 – Optional Profiles or Subprofiles 542
Table B.87 – JBOD Standard Dependencies 546
Table B.88 – Required Functional Profiles 546
Table B.89 – Required CIM Elements 548
Table B.90 – Required Properties for ComputerSystem.....548
Table B.91 – Required Properties for ComputerSystemPackage.....551
Table B.92 – Required Properties for FCPort 551
Table B.93 – Required Properties for ConcreteIdentity 554
Table B.94 – Required Properties for SCSIProtocolController 554
Table B.95 – Required Properties for ProtocolControllerForUnit.....556
Table B.96 – Required Properties for SystemDevice.....556
Table B.97 – Optional Profiles or Subprofiles 556
Table B.98 – OperationStatus for Component ComputerSystem 559
Table B.99 – Required CIM Elements 560
Table B.100 – Required Properties for ComputerSystem.....561
Table B.101 – Required Properties for ExtraCapacitySet.....561
Table B.102 – Required Properties for ConcreteIdentity 562
Table B.103 – Required Properties of MemberOfCollection.....562
Table B.104 – Optional Profiles or Subprofiles 562
Table B.105 – HostDiscoveredResources Standard Dependencies.....564
Table B.106 – Required Functional Profiles 564
Table B.107 – SCSI Device Type Mapping 568
Table B.108 – Required CIM Elements 569
Table B.109 – Required Properties for AdminDomain.....570

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table B.110 – Required Properties for Component 570
Table B.111 – Required Properties for DeviceSAPImplementation 570
Table B.112 – Required Properties for FCPort 570
Table B.113 – Required Properties for HostedCollection 575
Table B.114 – Required Properties of ProtocolControllerForPort 575
Table B.115 – Required Properties for LogicalNetwork 575
Table B.116 – Required Properties of MemberOfCollection 575
Table B.117 – Requited Properties for ProtocolEndpoint 576
Table B.118 – Required Properties for SystemDevice 576
Table B.119 – Required Properties for StorageVolume 576
Table B.120 – Optional Profiles or Subprofiles 578
Table B.121 – Required CIM Elements 579
Table B.122 – Required Properties for ProtocolControllerForPort 579
Table B.123 – Required Properties for SCSIProtocolController 579
Table B.124 – Required Properties for ProtocolControllerAccessesUnit 580
Table B.125 – Optional Profiles or Subprofiles 580
Table B.126 – Required CIM Elements 582
Table B.127 – Required Properties for SCSIProtocolController 582
Table B.128 – Required Properties for ProtocolControllerForUnit 582
Table B.129 – Optional Profiles or Subprofiles 583
Table B.130 – Router Standard Dependencies 584
Table B.131 – Required Functional Profiles 584
Table B.132 – Required CIM Elements 587
Table B.133 – Required Properties for ComputerSystem 588
Table B.134 – Required Properties for ComputerSystemPackage 590
Table B.135 – Required Properties for FCPort 591
Table B.136 – Required Properties for LogicalDevice 596
Table B.137 – Required Properties for ConcreteIdentity 596
Table B.138 – Required Properties for LogicalPortGroup 596
Table B.139 – Required Properties of MemberOfCollection 596
Table B.140 – Required Properties for SCSIProtocolController 597
Table B.141 – Required Properties for ProtocolControllerAccessesUnit 599
Table B.142 – Required Properties for ProtocolControllerForUnit 599
Table B.143 – Optional Profiles or Subprofiles 600
Table C.1 – CIM/SNMP Data Type Mapping 602
Table E.1 – Subprofile Required Classes, Associations, Methods and Indications 626
Table E.2 – Required Properties for Provider 626
Table E.3 – Required Properties for ProviderCapabilities 627
Table E.4 – Required Properties ProviderElementCapabilities 628
Table E.5 – Required Properties for ClassSupportForNamespace 628
Table E.6 – Required Properties for ProviderModule 628
Table E.7 – Required Properties for ProviderInModule 629
Table E.8 – Required Properties for IndicationFilter 629
Table E.9 – Required Properties for FiltersSupported 629
Table E.10 – Required Properties for ObjectManagerIsProviderRequired 630

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Figure 1 – Interface functions 44
Figure 2 – Large SAN Topology 47
Figure 3 – Example Client Server Distribution in a SAN 48
Figure 4 – SMI-S Modelling Conventions 50
Figure 5 – Object Model/Server Relationship 51
Figure 6 – Canonical inheritance 52
Figure 7 – Sample CIM-XML Message 53
Figure 8 – Operational environment 56
Figure 9 – Transport stack 59
Figure 10 – Reference Mode 60
Figure 11 – Cluster model 65
Figure 12 – Common elements 66
Figure 13 – Server Profile Instance Diagram 67
Figure 14 – Volume shared across namespaces 68
Figure 15 – Indication Filter Schema 74
Figure 16 – Indications Schema 75
Figure 17 – Physical Package Instance 90
Figure 18 – Software Instance Diagram 97
Figure 19 – Access Point Instance Diagram 100
Figure 20 – Cluster Instance 105
Figure 21 – Disk Drive Instance Model 110
Figure 22 – Extent Mapping Instance 122
Figure 23 – Location Instance 125
Figure 24 – Instance Diagram for Copy Services 130
Figure 25 – StorageSynchronized Association 131
Figure 26 – State Diagram for Snapshot 142
Figure 27 – State Diagram for Mirrors 144
Figure 28 – Job Control Subprofile Model 156
Figure 29 – Storage Configuration 157
Figure 30 – Pool Manipulation Instance Diagram 163
Figure 31 – Storage Configuration 168
Figure 32 – Pool Creation - Initial State 170
Figure 33 – Pool Creation - Step 2 170
Figure 34 – Pool Creation - Step 3 171
Figure 35 – Pool Creation - Step 4 171
Figure 36 – LUN Creation Instance Diagram 185
Figure 37 – Storage Pool Example 187
Figure 38 – Volume Creation - Initial State 188
Figure 39 – Volume Creation - Step 1 188
Figure 40 – Volume Creation - Step 2 189
Figure 41 – Volume Creation - Step 3 190
Figure 42 – DeviceCredentials Subprofile Model 204
Figure 43 – Back-end Ports Instance 210
Figure 44 – Generic System with no ConfigurationService 217
Figure 45 – Generic System with ControllerConfiguration Service 217
Figure 46 – Authorization and Access Rights 219
Figure 47 – Initiator Setting Data Example 220

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Figure 48 – Entire Model....221
Figure 49 – Simple StorageVolume Model....226
Figure 50 – Two view/Two LogicalDevice Use Case....226
Figure 51 – Volume used in multiple views....227
Figure 52 – Fabric Instance Diagram....251
Figure 53 – Zoning Instance Diagram (AdminDomain)....252
Figure 54 – Zoning Instance Diagram (ComputerSystem)....253
Figure 55 – Switch Instance Diagram....298
Figure 56 – Switch Blade Instance Diagram....307
Figure 57 – FC HBA Instance Diagram....312
Figure 58 – Array Profile Instance Diagram....327
Figure 59 – Array Packages Diagram....345
Figure 60 – In Band Virtualization Overview Diagram....348
Figure 61 – In Band Virtualization System Instance....349
Figure 62 – StorageLibrary-centric Instance Diagram....364
Figure 63 – MediaAccessDevice-centric Instance Diagram....365
Figure 64 – ChangerDevice-centric Instance Diagram....365
Figure 65 – Physical View Instance Diagram....366
Figure 66 – StorageMediaLocation Instance Diagram....367
Figure 67 – LimitedAccessPort Linkages....381
Figure 68 – Server Model....386
Figure 69 – ObjectManagerAdapter Subprofile Model....407
Figure 70 – System Diagram....410
Figure 71 – Host Bus Adapter Model....410
Figure 72 – Switch Model....411
Figure 73 – Array Instance....412
Figure 74 – Virtualization Instance....413
Figure 75 – Fabric Topology....414
Figure 76 – SA Server Configuration....437
Figure 77 – Complete Reference Model....442
Figure 78 – Configuration Administration....450
Figure A.1 – Reference Model with Policy Server....457
Figure A.2 – Policy Components....458
Figure B.1 – Sparing Instance....460
Figure B.2 – InterLibraryPort Connection Instance Diagram....463
Figure B.3 – Virtual ChangerDevices....466
Figure B.4 – Instance Diagram for Fibre Channel Connection....468
Figure B.5 – Virtual ChangerDevices Sharing a Chassis....473
Figure B.6 – Library Capacity Instance Diagram....475
Figure B.7 – Extender Instance Diagram....500
Figure B.8 – Management Appliance Subprofile Diagram....514
Figure B.9 – Management Appliance Instance Diagram....514
Figure B.10 – Out-of-band Virtualization Block Diagram....522
Figure B.11 – Out-of-Band Virtualization Subprofile Diagram....524
Figure B.12 – Metadata System Instance Diagram....525
Figure B.13 – Translation Engine Instance Diagram....527
Figure B.14 – Translation Engine Back-end Ports Instance....528

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Figure B.15 – Virtualizer/Fabric Interaction Overview.....544
Figure B.16 – Extra Capacity Set Instance Diagram 558
Figure B.17 – Host Discovered Resources Instance Diagram 1.....565
Figure B.18 – Host Discovered Resources Instance Diagram 2.....566
Figure B.19 – Router Instance Diagram 585
Figure E.1 – Provider Subprofile Model 625

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

INFORMATION TECHNOLOGY — STORAGE MANAGEMENT

FOREWORD

1) ISO (International Organization for Standardization) and IEC (International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards. Their preparation is entrusted to technical committees; any ISO and IEC member body interested in the subject dealt with may participate in this preparatory work. International governmental and non-governmental organizations liaising with ISO and IEC also participate in this preparation.

2) In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

3) The formal decisions or agreements of IEC and ISO on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC and ISO member bodies.

4) IEC, ISO and ISO/IEC publications have the form of recommendations for international use and are accepted by IEC and ISO member bodies in that sense. While all reasonable efforts are made to ensure that the technical content of IEC, ISO and ISO/IEC publications is accurate, IEC or ISO cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.

5) In order to promote international uniformity, IEC and ISO member bodies undertake to apply IEC, ISO and ISO/IEC publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any ISO/IEC publication and the corresponding national or regional publication should be clearly indicated in the latter.

6) ISO and IEC provide no marking procedure to indicate their approval and cannot be rendered responsible for any equipment declared to be in conformity with an ISO/IEC publication.

7) All users should ensure that they have the latest edition of this publication.

8) No liability shall attach to IEC or ISO or its directors, employees, servants or agents including individual experts and members of their technical committees and IEC or ISO member bodies for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication of, use of, or reliance upon, this ISO/IEC publication or any other IEC, ISO or ISO/IEC publications.

9) Attention is drawn to the normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.

10) Attention is drawn to the possibility that some of the elements of this International Standard may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

International Standard ISO/IEC 24775 was prepared by subcommittee 25: Interconnection of Information technology equipment, of ISO/IEC joint technical committee 1: Information technology.

This International Standard has been approved by vote of the member bodies and the voting results may be obtained from the address given on the second title page.

This publication has been drafted in accordance with the ISO/IEC Directives, Part 2.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

INTRODUCTION

The SNIA (Storage Networking Industry Association) Storage Management Initiative Specification standard (SM-S) is divided into a number of clauses and annexes.

Clause 1 defines the scope of this standard and places it in context of other standards and standards projects.

Clause 2 enumerates the normative references that apply to this standard.

Clause 3 specifies definitions, symbols, abbreviations, and document conventions.

Clause 4 describes the underlying concepts of the interface, including its anticipated usage model and business rationale.

Clause 5 identifies high-level requirements and capabilities addressed by this standard.

Clause 6 defines the transport layer and reference model employed in the standard.

Clause 7 defines the object model underlying the standard.

Clause 8 defines the security model for the standard.

Clause 9 defines the discovery mechanism assumed by the standard.

Clause 10 defines the users roles assumed by the standard.

Clause 11 defines the installation and upgrade process defined by the standard.

Annex A describes potential enhancements and future developments for the standard (informative).

Annex B defines profiles which are currently in development (informative).

Annex C defines a mapping between the CIM object model and SNMP MIB structures. (informative).

Annex D defines a compliance procedure for the standard (normative).

Annex E defines optional profiles in the context of the standard (informative).

The bibliography provides useful context for the understanding of this standard.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

INFORMATION TECHNOLOGY – STORAGE MANAGEMENT

1 Scope

This International Standard defines the a method for the interoperable management of a heterogeneous Storage Area Network (SAN).

This International Standard describes the information available to a WBEM Client from an SMI-S compliant CIM Server.

This International Standard describes an object-oriented, XML-based, messaging-based interface designed to support the specific requirements of managing devices in and through Storage Area Networks (SANs).

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

2 Normative references

2.1 International standards

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.

ISO/IEC 14165-414, *Information technology - Fibre Channel - Part 414: Generic Services - 4 (FC-GS-4)*

ISO/IEC 14776-452, *Information technology - Small Computer System Interface (SCSI) - Part 452: Primary Commands-2 (SPC-2) [ANSI INCITS 351-2001]*

ITU-T Rec. X.509: May 2000, *The Directory: Public-key and attribute certificate frameworks (DER encoded)*

2.2 Other specifications

All references in this subclause were correct at the time of approval of this International Standard. The provisions of the referenced specifications, as identified in this subclause, are valid within the context of this in International standard. The reference to a specification within this International Standard does not give it any further status within ISO or IEC. In particular, it does not give the referenced specification the status of an International Standard.

Table 1 - Standard dependencies for SMI-S

Standard	Version	Organization
CIM Specification	2.2	DMTF
CIM Operations over HTTP	1.1	DMTF
CIM Schema	2.8 Preliminary	DMTF
CIM-XML		DMTF
UML		OMG
SLP		IETF
Key words for use in RFCs to Indicate Requirement Levels		IETF (RFC2119)
Hypertext Transfer Protocol - HTTP	1.0 (1.1)	IETF (RFC1945, RFC2068)
An Extension to HTTP: Digest Access Authentication		IETF (RFC2069)
Secure Sockets Layer (SSL)	3.0	IETF
Multipurpose Internet Mail Extensions (MIME) - Part One: Format of Internet Message Bodies	November, 1996	IETF (RFC2045)
PKCS #12: Personal Information Exchange Syntax	1.0	RSA Laboratories

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

3 Definitions, symbols, abbreviations, and conventions

3.1 Definitions

3.1.1
Address masking
Address masking is a function of a host I/O controller (device driver) that filters access to certain storage resources on the SAN. It puts the responsibility of segregating I/O paths on the individual server system in the SAN and requires coordination of all servers to avoid access collisions. Also called Host-based LUN Masking

3.1.2
Addressable Unit
Storage addressable unit (e.g. LUN, Virtual Disk, Logical Disk, Logical Volume, Volume Set)

3.1.3
Agent
Object Manager that includes the provider service for a limited set of resources

NOTE An Agent may be embedded or hosted and can be an aggregator for multiple devices.

3.1.4
Aggregation
strong form of an association. For example, the containment relationship between a system and the components that make up the system can be called an aggregation. An aggregation is expressed as a Qualifier on the association class. Aggregation often implies, but does not require, that the aggregated objects have mutual dependencies

3.1.5
ATM
acronym for Asynchronous Transfer Mode

3.1.6
Attributes
collection of tags and values describing the characteristics of a service

3.1.7
Attribute Reply (AttrRply)
reply to an Attribute Request (optional)

3.1.8
Attribute Request (AttrRqst)
request for attributes of a given type of service or attributes of a given service (optional)

3.1.9
Cardinality
the number of values that may apply to an attribute for a given entity. Refer to UML Standards

3.1.10
CIM
acronym for Common Information Model. An object oriented description of the entities and relationships in a business' management environment maintained by the Distributed Management Task Force. Abbreviated CIM. CIM is divided into a Core Model and Common Models. The Core Model addresses high-level concepts (such as systems and devices), as well as fundamental relationships (such as dependencies). The Common

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Models describe specific problem domains such as computer system, network, user or device management. The Common Models are subclasses of the Core Model and may also be subclasses of each other

3.1.11

Client
process that issues requests for service. Formulating and issuing requests may involve multiple client processes distributed over one or more computer systems. The collection of client processes involved in formulating and issuing requests is known as a *consumer*

3.1.12

Completion Semantics
specifies how a method notifies its caller that its operations have completed. To this end, notification of completion is accomplished in either of two ways:

Asynchronous notification: Upon return of the method, its operations may not have yet completed. The caller is then required to employ some other mechanism to determine when the operations complete. Events, callbacks and polling are examples of mechanisms available to the caller in this regard.

Synchronous notification: The thread calling the method blocks until the method's operations succeed or fail.

Completion semantics refer to the operations executed by the method and not the method completion itself. For example, suppose we write a method to resync a split-mirror. We recognize that this could take an indeterminate amount of time, so we design a method, *resync()*, to spawn a task to manage the set of operations required for the resynchronization and then return to the caller. When the method, resync(), completes and returns to the caller, the resynchronization of the mirrors will most likely not have completed. So, the method has completed but its operations have not.

3.1.13

Consumer
host, identified by HBA WWN or other identifier, that is allowed access to a storage addressable unit

3.1.14

Control Software
body of software that provides common control and management for one or more disk arrays or tape arrays. Control software presents the arrays of disks or tapes it controls to its operating environment as one or more virtual disks or tapes. Control software may execute in a disk controller or intelligent host bus adapter or in a host computer. When it executes in a disk controller or adapter, control software is often referred to as firmware

3.1.15

Concurrency Control Protocol
set of rules for identifying and resolving resource conflicts between multiple, non-cooperating clients. The three most common concurrency protocols are the following.

Lock ordering: Transactions are ordered according to the order of arrival of their operations at the resource(s).

Optimistic ordering: Transactions proceed until they are ready to commit, whereupon a check is made to see whether they have performed conflicting operations.

Timestamp ordering: Transactions are ordered according to the time they were initiated.

3.1.16

Cooperating Clients
set of consumer processes that are aware of each other and are able to coordinate access to (and control of) resources among themselves

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

3.1.17

DA Advertisements (DAAdvert)
solicited (unicast) or unsolicited (multicast) advertisement of Directory Agent availability

3.1.18

Data Invariant
data invariant is the name given to the consistency-state of shared data. A data invariant shall always be TRUE. When the data invariant is violated, the invariant shall be protected via mutual exclusion. For example, suppose I have a list of records and a record pointer, `i`, that is always set to point to the last record in the list. In this example, the invariant is *the record pointer always points to the last record*

But observe what happens when I append a record to the list as follows:

(a) Add record to record[i];

(b) i += 1.

After (a) completes, but before (b) is invoked, *i* no longer points to the last record in the list. Now, suppose another thread comes along and attempts to read the last record in the list. In this case, the thread will get the penultimate record, not the last one – Because *`i`* has not yet been updated. The solution to this problem is to serialize access to both operations using a lock or a semaphore.

BEGIN LOCK

(a) Add record to record[i]

(b) i += 1;

END LOCK

3.1.19

Device
storage system that is addressable from the SAN

3.1.20

DHCP
acronym for dynamic host control protocol. An Internet protocol that allows nodes to dynamically acquire ("lease") network addresses for periods of time rather than having to pre-configure them. Abbreviated DHCP. DHCP greatly simplifies the administration of large networks, and networks in which nodes frequently join and depart

3.1.21

Directory
repository of information about objects that may be accessed via a Directory Service

3.1.22

Directory Agent (DA):
in the context of SLP, a process that caches SLP service advertisements registered by Service Agents and forwards the service advertisements to User Agents on demand

3.1.23

Discovery
discovery provides information about what physical and logical SAN entities have been found within the management domain. Enough information is provided to support the creation of correct Topology maps. This information changes dynamically, as SAN entities are added, moved or removed

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

3.1.24

DLT
acronym for Digital Linear Tape. A family of tape device and media technologies developed by Quantum Corporation

3.1.25

DRM
the Disk Resource Management (DRM) Work Group is defining standard data and interfaces for the management of disk storage facilities, as well as creating guidelines for implementing well-managed block-oriented storage systems.

3.1.26

DMTF
Distributed Management Task Force. An industry organization that develops management standards for computer system and enterprise environments. DMTF standards include WBEM, CIM, DMI, DEN and ARM. Abbreviated DMTF. The DMTF has a web site at www.dmtf.org

3.1.27

Enclosure
box or cabinet

3.1.28

Enumerate
operation that is used to enumerate subclasses, subclass names, instances and instance names in the target Namespace. If successful, the method returns zero or more requested elements that meet the required criteria

3.1.29

Extent
set of consecutively addressed FBA disk blocks that is allocated to consecutive addresses of a single file;

set of consecutively located tracks on a CKD disk that is allocated to a single file;

set of consecutively addressed disk blocks that is part of a single virtual disk-to-member disk array mapping. A single disk may be organized into multiple extents of different sizes and may have multiple (possibly) non-adjacent extents that are part of the same virtual disk-to-member disk array mapping. This type of extent is sometimes called a logical disk

3.1.30

Extrinsic Method
method defined as part of CIM Schema. Extrinsic methods are invoked on a CIM Class (if static) or Instance (otherwise). An extrinsic method call is represented in XML by the <METHODCALL> element and the response to that call represented by the <METHODRESPONSE> element. cf. Intrinsic Method

3.1.31

Fabric
any interconnect between two or more Fibre Channel N_Ports, including point-to-point, loop and Switched Fabric

Switched Fabric: fabric comprised of one or more Switches

3.1.32

FC-GS-3
Fibre Channel - Generic Services 3. Abbreviation FC-GS-3 or GS-3

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

3.1.33

FIP
Acronym for Federal Information Processing Standard. Standards (and guidelines) produced by NIST for government-wide use in the specification and procurement of Federal computer systems

3.1.34

Grammar
formal definition of the syntactic structure of a language (see syntax), normally given in terms of production rules that specify the order of constituents and their sub-constituents in a sentence (a well-formed string in the language). Each rule has a left-hand side symbol naming a syntactic category (e.g. "noun-phrase" for a natural language grammar) and a right-hand side that is a sequence of zero or more symbols. Each symbol may be either a terminal symbol or a non-terminal symbol. A terminal symbol corresponds to one "lexeme" - a part of the sentence with no internal syntactic structure (e.g. an identifier or an operator in a computer language). A non-terminal symbol is the left-hand side of some rule

3.1.35

GS-3
Refer FC-GS-3

3.1.36

HBA
host bus adapter, card that contains ports for host systems

3.1.37

Host
computer running an O/S

3.1.38

HTTP
request-reply protocol called the Hypertext Transfer Protocol, HTTP

3.1.39

Hub
interconnect element that supports a ring topology

3.1.40

Inheritance Relationship
refer to UML Standards

3.1.41

Interconnect Element
non-terminal network elements (Switches, hubs, routers, directors)

3.1.42

Interface Definition Language (IDL)
high-level declarative language that provides the syntax for interface declarations
Some examples of IDLs in common usage today are:

- DCE’s RPC IDL
- Microsoft’s DCOM IDL (based on the DCE IDL)
- OMG IDL (used to define the DOM XML interface)
- DMTF MOF (an IDL-derived specification).

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

3.1.43

Intrinsic Method
Operations made against a CIM server and a CIM Namespace independent of the implementation of the schema defined in the server. Examples of intrinsic methods in XML include the <IMETHODCALL> element and the response to that call represented by the <IMETHODRESPONSE> element, cf. Extrinsic Method

3.1.44

Language-Binding
association of a programming language (e.g., C++, Java, C) with an interface definition language. For example, OMG IDL supports many language bindings because it can be compiled into a variety of programming languages (C, C++, Java, ADA, COBOL, etc.). By contrast, Microsoft's DCOM IDL only supports one language binding, C++. Similarly, Java IDL also supports only one language binding (Java).

Some IDLs do not support any formal language bindings. DMTF's MOF, for example, is derived from OMG's IDL but is used as a data modeling language more in the spirit of SQL than programmatic interfaces.

3.1.45

Lock Manager
short name for Lock Management Server

3.1.46

Logical Unit Number (LUN)
SCSI identifier of a logical unit within a Target

3.1.47

LTO
acronym for Linear Tape Open

3.1.48

LUN Mapping
process of creating a disk resource and defining its external access paths, by configuring LUs (Logical Units) from the disk array logical disk volumes - either by grouping them as a single larger LU or by creating partitions. The "Logical Unit Number (LUN)" is then be mapped to an external ID descriptor (for example: a SCSI Port, Target ID and LU Number). An LU may be mapped for access from multiple ports and/or multiple target IDs, providing alternate paths for nonstop data availability.

LUN Mapping is a necessary task to be able to export the LUN to the Fabric/Server/etc. It can be done independent of any knowledge of the intended use of the LUN. Only LUNs that are exposed via a "Port" are available for access.

3.1.49

LUN Masking
process of configuring software in SAN nodes to determine which hosts have access to exported drives. LUN masking can be either server-based address masking or storage based port mapping, cf. Port Mapping

3.1.50

MAN
acronym for Metropolitan Area Network. A network that connects nodes distributed over a metropolitan (city-wide) area as opposed to a local area (campus) or wide area (national or global). Abbreviated MAN. From a storage perspective, MANs are of interest because there are MANs over which block storage protocols (e.g., ESCON, Fibre Channel) can be carried natively, whereas most WANs that extend beyond a single metropolitan area do not currently support such protocols

--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

3.1.51

Marshalling
set of operations by which a message is converted into a transfer syntax. In HTTP, requests and replies are marshaled into formatted ASCI-text strings

3.1.52

Method
name of one or more operation(s) performed by an instance of an object class. Methods are distinguished from operations as follows:
A method is a name for one or more operations that may execute when the method is invoked. For example, when the method, `printSelf()`, is called, the operation of printing the state of the reference object is executed.

Synonyms are: Function, procedure or subroutine. Usage of these terms should be deprecated.

In most models, a method is characterized by its name, return-type, parameters, completion semantics (asynchronous or synchronous) and side-effects (e.g., event generation, message propagation, etc.).

Methods are *specified* in an IDL.

Methods are *declared* in source header files of a programming language (.h files, Java Interface files, etc.,).

Methods are *defined* (or *implemented*) in source implementation files (e.g., .cpp, . java, class files).

Method specifications are language independent. Method declarations and implementations are, by construction, language dependent.

3.1.53

Monitoring
monitoring provides management information about the current state of individual logical and physical SAN entities. This information changes dynamically, as SAN entities perform their functions, are serviced, experience errors, etc. Monitoring can only be done on SAN entities that are known via Discovery

3.1.54

NAA
acronym for Network Address Authority. A four bit identifier defined in FC-PH to denote a network address authority (i.e., an organization such as ITU or IEEE that administers network addresses).

3.1.55

NDMP
acronym for Network Data Management Protocol. A communications protocol that allows intelligent devices on which data is stored, robotic library devices and backup applications to intercommunicate for the purpose of performing backups. Abbreviated NDMP.

An open standard protocol for network-based backup of NAS devices. Abbreviated NDMP. NDMP allows a network backup application to control the retrieval of data from, and backup of, a server without third-party software. The control and data transfer components of backup and restore are separated. NDMP is intended to support tape drives, but can be extended to address other devices and media in the future. The Network Data Management Task Force has a web site at HTTP://www.ndmp.org

3.1.56

N_Port
refer to Port. Node

a collection of Ports. A Fiber channel device with a group of ports

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

An addressable entity connected to an I/O bus or network. Used primarily to refer to computers, storage devices and storage subsystems. The component of a node that connects to the bus or network is a port

3.1.57

Non-cooperating clients
set of consumer processes that are independent of each other, compete for resources and execute independently of the other. User processes on a multi-user machine are non-cooperating clients with respect to the operating system

3.1.58

Operation
action executed within the body of a method (AKA procedure, function or subroutine). Operations are distinct from methods (see Method)

3.1.59

Out-of-Band
transmission of management information for Fibre Channel components outside of the Fibre Channel network, typically over Ethernet

3.1.60

PKI
acronym for public key infrastructure. A framework established to issue, maintain and revoke public key certificates accommodating a variety of security technologies

3.1.61

Platform
collection of Nodes

3.1.62

Port
connection point for links

N_Port: A hardware entity that includes a Link_Control_Facility. It may act as an Originator, a Responder or both.

N_Port identifier: A Fabric unique address identifier by which an N_Port is uniquely known. The identifier may be assigned by the Fabric during the initialization procedure. The identifier may also be assigned by other procedures not defined in FC-FS.

Port_Name: As defined in FC-FS.

3.1.63

Port Mapping
function of a storage subsystem to define which hosts have access to exported drives. This configuration authorizes specified server HBA WWNs to access the secured LU while preventing other unauthorized servers/hosts from either seeing the secured LU or accessing the data contained on the secured LU, cf. LUN Masking

3.1.64

Protocol
set of rules that define and constrain data, operations or both. For example, xmlCIM uses XML as its transfer syntax and HTTP as the request-reply protocol HTTP is layered over the TCP/IP network protocol

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

3.1.65

Provider
COM server that communicates with managed objects to access data and event notifications from a variety of sources, such as the system registry or an SNMP device. Providers forward this information to the CIM Object Manager for integration and interpretation

class provider: A COM server that supplies class definitions. Class providers can support data retrieval, modification, deletion, enumeration and query processing.

property provider: A type of provider that supports the retrieval and modification of the CIM properties.

3.1.66

Relationship
SOURCE (DMTF)

refer to UML Standards.

3.1.67

Required Reference
SOURCE (DMTF)

refer to UML Standards

3.1.68

SA Advertisement (SAAdvert)
information describing a service that consists of the Service Type, Service Access Point, lifetime and Attributes

3.1.69

SAN
Acronym for storage area network.

Acronym for Server Area Network that connects one or more servers.

Acronym for System Area Network for an interconnected set of system elements.

A group of fabrics that have common leaf elements.

3.1.70

Scope
set of services, typically making up a logical administrative group

3.1.71

Semantics
meaning or behavior associated with an entity. For example, we might say the semantics of the method, `resync_mirror()`, is encoded in the method name. By contrast, the semantics of the UNIX `ioctl()` method is encoded in the command parameter.

3.1.72

Server
process that fields and/or dispatches requests. Honoring a request may involve more than one server process distributed over one or more computer systems. The collection of server processes that are involved in honoring a request are known as *service providers*

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

3.1.73

Service Access Point
network address and port number of a process offering a service

3.1.74

Service Acknowledgement (SrvAck)
reply to a SrvReg request

3.1.75

Service Agent (SA)
in the context of SLP, this refers to a process working on behalf of one or more services to advertise the services in the network

3.1.76

Service Agent Server (SAServer)
in the context of SLP, this refers to a process working on behalf of one or more Service Agents to listen on a particular port number for SLP service requests

3.1.77

Service Deregister (SrvDereg)
request to deregister a service or some attributes of a service (optional)

3.1.78

Service Register (SrvReg)
request to register a service or some attributes of a service

3.1.79

Service Reply (SrvRply)
reply to a Service Request

3.1.80

Service Request (SrvRqst)
request for a service on the network

3.1.81

SES
ISO/IEC 14776-371 (under study and based on ANSI INCITS 305-1998 (R2003); ANSI INCITS 305-1998/AM-1-2000 (R2003)) for management of environmental factors such as typewriter, power, voltage, etc.

3.1.82

Service Type
class of a network service represented by a unique string (for example a namespace assigned by IANA)

3.1.83

Service Type Reply (SrvTypeRply)
reply to a Service Type Request (optional)

3.1.84

Service Type Request (SrvTypeRqst)
request for all types of service on the network (optional)

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

3.1.85

Service Type Template
formaized, computer-readable description of a Service Type

3.1.86

Service URL
Uniform Resource Locator for a service containing the service type name, network family, Service Access Point and any other information needed to contact the service

3.1.87

SLP
acronym for Service Location Protocol

3.1.88

SNIA
acronym for Storage Networking Industry Association. An association of producers and consumers of storage networking products whose goal is to further storage networking technology and applications

3.1.89

SNMP
acronym for Simple Network Management Protocol. An IETF protocol for monitoring and managing systems and devices in a network. The data being monitored and managed is defined by a MIB. The functions supported by the protocol are the request and retrieval of data, the setting or writing of data and traps that signal the occurrence of events

3.1.90

SNMP Trap
a type of SNMP message used to signal that an event has occurred

3.1.91

Soft Zone
Zone consisting of Zone Members that are made visible to each other through Client Service requests. Typically, Soft Zones contain Zone Members that are visible to devices via Name Server exposure of Zone Members. The Fabric does not enforce a Soft Zone. Note that well known addresses are implicitly included in every Zone

3.1.92

SRM
acronym for storage resource management. Management of physical and logical storage resources, including storage elements, storage devices, appliances, virtual devices, disk volume and file resources.

3.1.93

SSL
acronym for Secure Sockets Layer. A suite of cryptographic algorithms, protocols and procedures used to provide security for communications used to access the world-wide web. The characters "https:" at the front of a URL cause SSL to be used to enhance communications security. More recent versions of SSL are known as TLS (Transport Level Security) and are standardized by the Internet Engineering Task Force (IETF)

3.1.94

SSP
acronym for Storage Service Provider

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

3.1.95

Switch
fibre channel interconnect element that supports a mesh topology

3.1.96

Symmetric Virtualization Appliance
synonym for an appliance that provides in-band virtualization. In-band virtualization appliance is the preferred term

3.1.97

Synchronous
method that blocks the calling thread until all operations have completed or failed

3.1.98

Syntax
(The structure of strings in some language. A language's syntax is described by a grammar. For example, the syntax of a binary number could be expressed as

binary_number = bit [binary_number]

bit = "0" | "1"

Meaning that a binary number is a bit optionally followed by a binary number and a bit is a literal zero or one digit. The meaning of the language is given by its semantics

3.1.99

TLS
acronym for Transport Layer Security

3.1.100

Transfer Syntax
formal rules (i.e., the protocol) governing the format (or representation) of messages as they are transferred between clients and servers

3.1.101

UDP
acronym for User Datagram Protocol. An Internet protocol that provides connectionless datagram delivery service to applications. Abbreviated UDP. UDP over IP adds the ability to address multiple endpoints within a single network node to IP

3.1.102

UML Standards
appendix D of the *Common Information Model (CIM) Specification, V2.0* (March 3, 1998)

Class - represented by a **rectangle**.

class name either stands alone in the rectangle or is in the uppermost segment. If present, the segment below the segment containing the name contains the properties of the class. If present, a third region indicates the presence of methods.

Lines indicate:

- **Inheritance** relationships (blue lines with arrows) – Otherwise known as “is-a” relationships

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

- **Aggregation/component** relationships (green lines with a diamond shape at the “aggregating” end) - Otherwise known as “has-a” relationships
- **Dependency and other** relationships (red lines) – Some of which are “uses-a” relationships
- **Relationship Labels -** Inheritance relationships are not specifically labelled or named, while all other associations are named
- **Cardinality** - the cardinalities of the references on both sides of an association are indicated by numeric values or an asterisk (*) at the endpoints of the association

The following cardinalities are typically used in the CIM Schema:

0..1 - Indicates an optional single-valued reference

1 - Indicates a **required**, single-valued reference

1..n or 1..* - Indicates either a single or multi-valued reference, that is **required***, 0..n or 0..* - Indicates an optional, single or multi-valued reference

Required Reference - the object and the association shall exist (or be instantiated) when the *other* referenced class is defined.

Weak Reference – indicated by the symbol, “w”, indicates that the referenced endpoint or class is “weak” or “weakly associated” with respect to the *other* class participating in the association. This means that the referenced class is scoped or named relative to the other class and the identifying keys of the other class are placed as properties in the “weak” class.

NOTE Note that this is not standard UML convention, but an added symbol in CIM diagrams.

3.1.103

Universal Markup Language (UML)
refer to UML Standards

3.1.104

URL
Uniform Resource Locator

3.1.105

User Agent (UA)
in the context of SLP, a process that attempts to establish contact with one or more services. A User Agent retrieves service information from Service Agents or Directory Agents

3.1.106

VAR
Value Added Remarketeer

3.1.107

Volume Set
synonym for virtual disk

3.1.108

VPD
Vital Product Data, a portion of the SCSI-2 specification (see 2.1, “International standards”)

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

3.1.109

WAN
acronym for Wide Area Network. A communications network that is geographically dispersed and that includes telecommunications links

3.1.110

Weak Reference
refer to UML Standards

3.1.111

WBEM
acronym for Web Based Enterprise Management. Web-Based Enterprise Management is an initiative in the DMTF. Abbreviated WBEM. It is a set of technologies that enables interoperable management of an enterprise. WBEM consists of CIM, an XML DTD defining the tags (XML encodings) to describe the CIM Schema and its data, and a set of HTTP operations for exchanging the XML-based information. CIM joins the XML data description language and HTTP transport protocol with an underlying information model, CIM to create a conceptual view of the enterprise

3.1.112

W3C
World-Wide Web Consortium.

3.1.113

XML
acronym for eXtensible Markup Language. A universal format for structured documents and data on the World-Wide Web. Abbreviated XML. The World-Wide Web Consortium is responsible for the XML specification. cf. http://www.w3.org/XML/

3.1.114

XML-CIM Listener
server application that receives and processes XML-CIM Export Message requests and issues CIM Export Message responses

3.1.115

XML-CIM Server
server that receives and processes XML-CIM Operation Requests and issues XML-CIM Operation Responses

3.1.116

Zone
group of ports and switches that allow access. Defined by a zone definition. cf. Hard Zone, Soft Zone

A collection of Zone Members. Zone Members in a Zone are made aware of each other, but not made aware of devices outside the Zone. A Zone can be defined to exist in one or more Zone Sets

3.1.117

Zone Definition
the parameters that define a Zone: the Zone Name, number of Zone Members and Zone Member definition

3.1.118

Zone Member
N_Port (or NL_Port) to be included in a Zone, as specified by its Zone Member Definition. N_Ports at well known addresses shall not be specified as Zone Members

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

3.1.119

Zone Member Definition
parameter by which a Zone Member is specified. A Zone Member may be specified by:

- a port on a Switch, (specifically by Domain_ID and port number); or,
- the device's N_Port_Name; or,
- the device's address identifier; or,
- the device's Node_Name.

3.1.120

Zone Set
one or more Zones that may be activated or deactivated as a group.

Zone Set Name: The name assigned to a Zone Set.

Zone Set State: The state of a Zone Set, which may be either activated or deactivated.

Active Zone Set: The Zone Set that is currently activated. Only one Zone Set may be activated at any time.

3.2 Symbols and abbreviations

||	concatenate
= or EQ	equal
≈	approximately equal
API	application programming interface
CDB	command descriptor block
FC	Fibre Channel
HBA	host bus adapter
IDL	interface definition language
IETF	Internet Engineering Task Force
IMA	iSCSI Management API
IP	Internet Protocol
iSCSI	Internet SCSI
iSD	iSCSI Direct
LSB	least significant bit
OS	operating system
RFC	Request for comments
SCSI	Small Computer System Interface
SAM-3	SCSI architecture model
SAN	storage area network
SNIA	Storage Networking Industry Association
SNMP	simple network management protocol
SPC-3	SCSI primary commands-3
TC	Technical Committee
TCP	Transmission control protocol
UDP	User datagram protocol

3.3 Keywords

3.3.1

expected
keyword used to describe the behavior of the hardware or software in the design models presumed by this standard. Other hardware and software design models may also be implemented

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

3.3.2

invalid
keyword used to describe an illegal or unsupported bit, byte, word, field or code value. Receipt of an invalid bit, byte, word, field or code value shall be reported as an error

3.3.3

mandatory
keyword indicating an item that is required to be implemented as defined in this standard to claim compliance with this standard

3.3.4

may
keyword that indicates flexibility of choice with no implied preference

3.3.5

may not
keywords that indicates flexibility of choice with no implied preference

3.3.6

obsolete
keyword indicating that an item was defined in prior standards but has been removed from this standard

3.3.7

opaque
keyword indicating that a value has no semantics or internal structure

3.3.8

optional
keyword that describes features that are not required to be implemented by this standard. However, if any optional feature defined by this standard is implemented, it shall be implemented as defined in this standard

3.3.9

reserved
keyword referring to bits, bytes, words, fields and code values that are set aside for future standardization. Their use and interpretation may be specified by future extensions to this or other standards. A reserved bit, byte, word or field shall be set to zero or in accordance with a future extension to this standard. Recipients are not required to check reserved bits, bytes, words or fields for zero values. Receipt of reserved code values in defined fields shall be reported as an error

3.3.10

shall
keyword indicating a mandatory requirement. Designers are required to implement all such requirements to ensure interoperability with other products that conform to this standard

3.3.11

should
keyword indicating flexibility of choice with a preferred alternative, equivalent to the phrase “it is recommended”

3.4 Conventions

Certain words and terms used in this International Standard have a specific meaning beyond the normal English meaning. These words and terms are defined either in clause 3 or in the text where they first appear.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Fields containing only one bit are usually referred to as the name bit instead of the name field.

Numbers that are not immediately followed by lower-case b or h are decimal values.

Numbers immediately followed by lower-case b (xxb) are binary values.

Numbers immediately followed by lower-case h (xxh) are hexadecimal values.

Hexadecimal digits that are alphabetic characters are upper case (i.e., ABCDEF, not abcdef).

Hexadecimal numbers may be separated into groups of four digits by spaces. If the number is not a multiple of four digits, the first group may have fewer than four digits (e.g., AB CDEF 1234 5678h)

Decimal fractions are initiated with a comma (e.g., two and one half is represented as 2,5).

Decimal numbers having a value exceeding 999 are separated with a space(s) (e.g., 24 255).

An alphanumeric list (e.g., a, b, c or A,B,C) of items indicate the items in the list are unordered.

A numeric list (e.g., 1, 2, 3) of items indicate the items in the list are ordered (i.e., item 1 shall occur or complete before item 2).

In the event of conflicting information the precedence for requirements defined in this standard is

1) text,
2) tables,
3) figures.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4 Introduction

4.1 Preamble

Storage Area Networks (SANs) are emerging as a prominent layer of IT infrastructure in enterprise class and midrange computing environments. Applications and functions driving the emergence of SAN technology include:

- Sharing of vast storage resources between multiple systems,
- LAN free backup,
- Remote, disaster tolerant, on-line mirroring of mission critical data,
- Clustering of fault tolerant applications and related systems around a single copy of data.

To accelerate the emergence of SANs in the market, the industry requires a standard management interface that allows different classes of hardware and software products supplied by multiple vendors to reliably and seamlessly interoperate for the purpose of monitoring and controlling resources. The SNIA Storage Management Initiative (SMI) was created to develop this specification (SMI-Specification or SMI-S), the definition of that interface. This standard provides for heterogeneous, functionally rich, reliable and secure monitoring/control of mission critical global resources in complex and potentially broadly distributed multi-vendor SAN topologies. As such, this interface overcomes the deficiencies associated with legacy management.

4.2 Business rationale

This interface is targeted at creating broad multi-vendor management interoperability and thus increasing customer satisfaction. To that end, this specification defines an “open” and extensible interface that allows subsystems and devices within the global context of a SAN to be reliably and securely managed by overlying presentation frameworks and management systems in the context of the rapidly evolving multi-vendor market. In specific, SAN integrators (like end-users, VARs and SSPs) can, via this standardized SAN management interface, more flexibly select between multiple vendors when building the hierarchy of software systems required to manage a large SAN independent of the underlying hardware systems. Additionally, SAN integrators can more flexibly select between alternate hardware vendors when constructing SAN configurations. Broad adoption of the standards defined and extended in this specification will provide increased customer satisfaction and will

- more rapidly expand the acceptance of SAN,
- accelerate customer acquisition of SAN technology;
- expand the total market.

Additionally, a single common management interface allows SAN vendors and integrators to decrease the time required to bring new more functional technology, products and solutions to market.

4.3 Interface definition

This management interface allows storage management systems to reliably identify, classify, monitor and control physical and logical resources in a SAN. The fundamental relationship of this interface to storage

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

management software, presentation frameworks, user applications, SAN physical entities (i.e., devices), SAN discovery systems and SAN logical entities is illustrated in Figure 1.

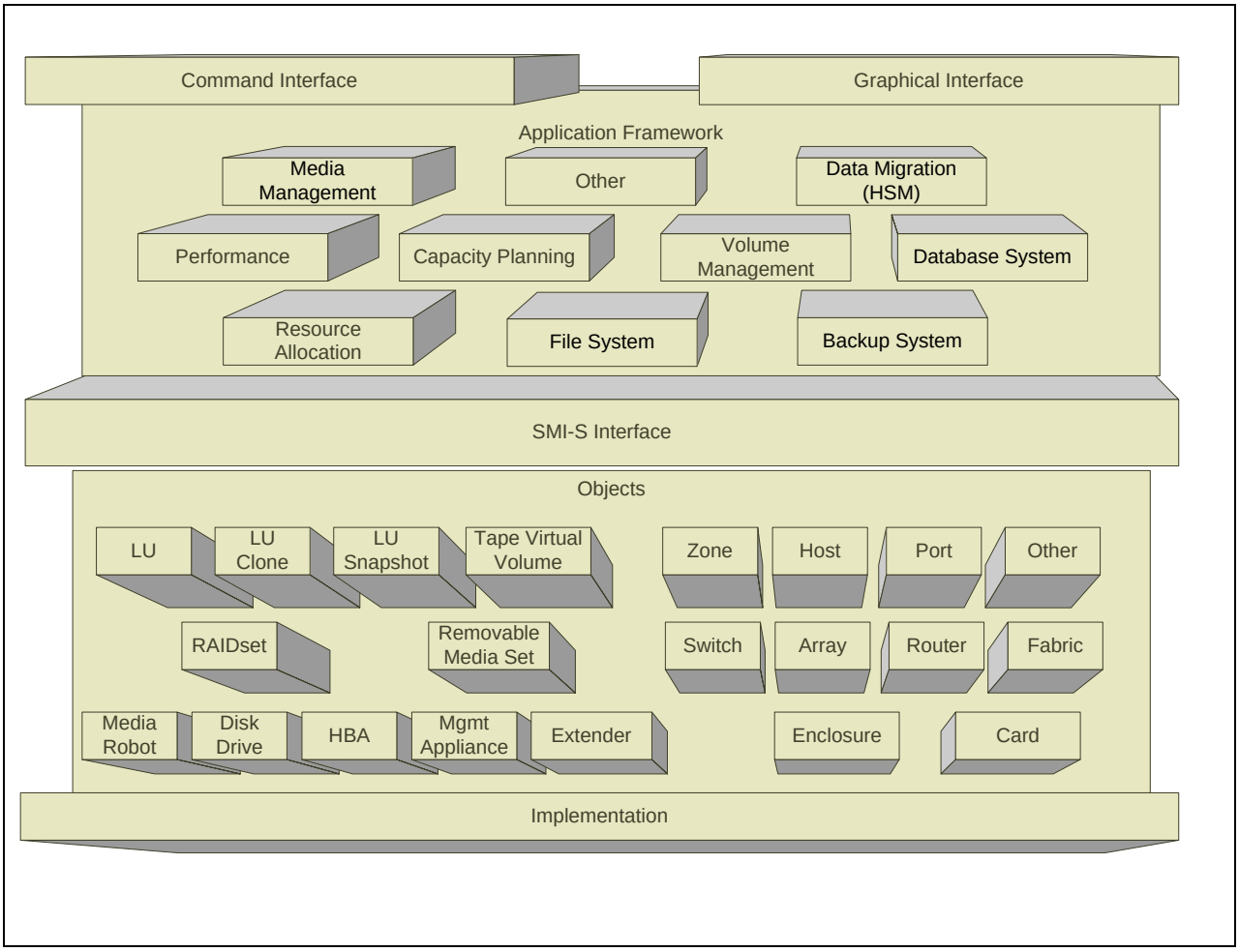


Figure 1 - Interface functions

The diagram illustrates that functions of the interface can be distributed across multiple SAN devices (i.e., Switches or Array Controllers) and/or software systems (i.e., Discovery Systems). While the functionality of the interface is likely to be distributed within or across a SAN, to insure that monitoring and control operations by clients are consistent and reliable, care should be taken to prevent the state of a given resource being simultaneously available to clients from multiple unsynchronized sources.

Example: A request by an SRM application and a backup engine for the bandwidth available on a given Fibre Channel path will need to be coordinated by a single monitoring entity to insure information consistency. Should the SRM application and Backup engine obtain different available bandwidth information for a given Fibre Channel path from multiple unsynchronized sources, conflicts can arise and degrade the efficiency of the environment.

Satisfying this requirement is the responsibility of parties configuring Storage and Network management clients in conjunction with the primitives defined in the specification.

NOTE Within this architecture (as depicted by the illustration above) entities like an appliance-based volume manager have the option to act as both a client and a server to the interface.

Example: A Host-based volume manager wants to construct a large storage pool from multiple SAN appliance based volumes, as well as volumes/LUNs originating from array

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

controllers. In this case, the host based volume manager shall inspect the characteristics of the volumes on both the SAN appliance and array controller prior to allocation. Additionally, the SAN appliance (which runs a volume manager) shall inspect the properties of storage devices when building its volumes. As such, the SAN appliance in this case is both a client and server in the management environment, depending on the action being performed.

Relative to Figure 1, examples of long-term functional requirements for the interface to properly satisfy the needs of clients using it include the following.

a) Clients shall be able to obtain sufficient information to discern the topology of the SAN.

b) Clients shall be able to reliably identify resources that have experienced an error/fault condition that has resulted in degraded/disabled operation.

c) Clients shall be able to construct a zone of allocation around a select group of host and storage resources.

d) Clients shall be able to identify nonvolatile storage resources available to a storage management system, to allow them to construct a storage pool of a consistent level of performance and availability.

e) Clients shall be able to identify third-party copy engines (and associated media libraries/robots) available to a cooperating backup engine, allowing it to allocate an engine/library/robot to a given backup task.

f) Clients shall be able to dynamically allocate non-volatile storage resources.

NOTE Each volume to be utilized is subject to strict availability and performance requirements. As a result, the file system needs to inspect the properties of each volume prior to allocation.

g) Clients shall be able to access sufficient topology and component information to allow a Storage Resource Management (SRM) application like a SAN performance monitor to examine topology and line utilization, in such a way that performance bottlenecks can be exposed.

h) Clients shall be able to employ appropriate data reporting and tracking to allow capacity planning system to identify each storage pool in the SAN and then interact with the manager of each pool to assess utilization statistics.

i) Clients shall be provided with adequate controls for a privileged, user-written application to restrict the use of a volume to a specific host, set of hosts or set of controller communication ports.

j) Clients shall be assured of timely propagation of data concerning the health and performance of the devices and subsystems in the SAN to fault isolation and analysis systems.

Example non-goals for this interface include the following.

a) Select a logical communication port over which to send/receive data.

b) Read/Write data to a volume.

c) Identify and recover from data communication errors and failures.

d) Synchronization message between two cluster nodes.

e) Log a new communication device into a network.

4.4 Technology trends

To be broadly embraced and long lived this management interface should respect and leverage key technology trends evolving within the industry. These include:

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

a) **Improved connectivity:** Whether available in-band (i.e., over Fibre Channel) or available out-of-band (i.e., over a LAN/MAN/WAN) or available over a mix of both, virtually all devices in a SAN have (or soon will have), access to a common communication transport suitable for carrying management information content (e.g., TCP/IP) that is used to transmit a standardized encoding (e.g., CIM-XML) of recognized semantics (e.g., CIM);

b) **Increased device manageability:** Through a common, general-purpose network transport and, where necessary, the use of proxy services through another resource (e.g., general purpose computer system), devices can support a standardized management interface;

Example: A legacy array controller is incapable of running the software necessary to implement a management server for this interface and uses a proxy server on a SAN appliance to communicate within the management environment.

Example: An HBA is incapable of running the software necessary to implement a management server for this interface and uses a proxy server on its host system to communicate within the management environment.

c) **XML Standardization:** XML is providing management protocols with an extensible, platform independent, human readable, content describable communication language for the first time. These protocols provide appropriate abstraction by separating the definition of the object model from the semantics/syntax of the protocol. Additionally, the transport-independent, content-description (i.e., markup) nature of XML allows it to be utilized by both web-enabled application and appliances;

d) **Increased SAN Complexity:** SANs are being configured with diverse classes of components and widely distributed topologies. Management clients and servers in the environment need to anticipate being widely distributed on systems, appliances and devices throughout large SAN topologies, while maintain-

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

ing real-time distributed state for logical entities. Figure 2 below provides an example of a single SAN built from multiple classes of components spanning three physical locations (i.e., Sites A, B and C).".

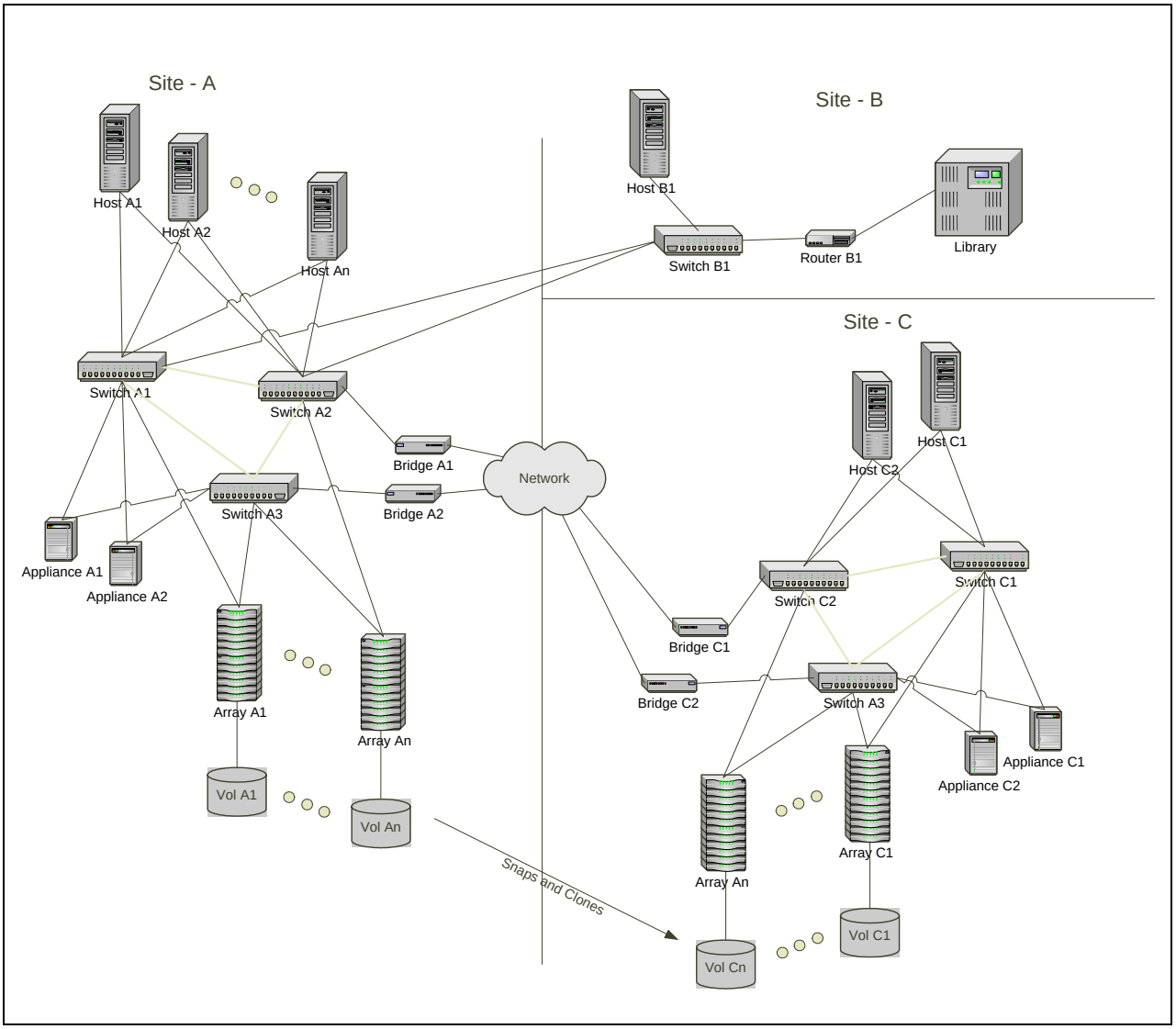


Figure 2 - Large SAN Topology

4.5 Management environment

Clients and Servers of this interface can be widely distributed on systems, appliances and devices across a network that includes one or more large SAN topologies.

The configuration in Figure 3 provides an example client/server distribution using in-band TCP/IP communication, out of band TCP/IP communication or employing proxy services to bridge legacy and/or proprietary communication interfaces. The device "Old Array Controller" is incapable of appropriate communication with clients and servers in the management environment to provide management access (i.e., a CIM Server). Access to the communication transport that clients and servers share for communication is

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

achieved via a proxy service on the host computer in the upper right hand corner of the illustration. All other clients and servers communicate via direct access to a common communication transport.

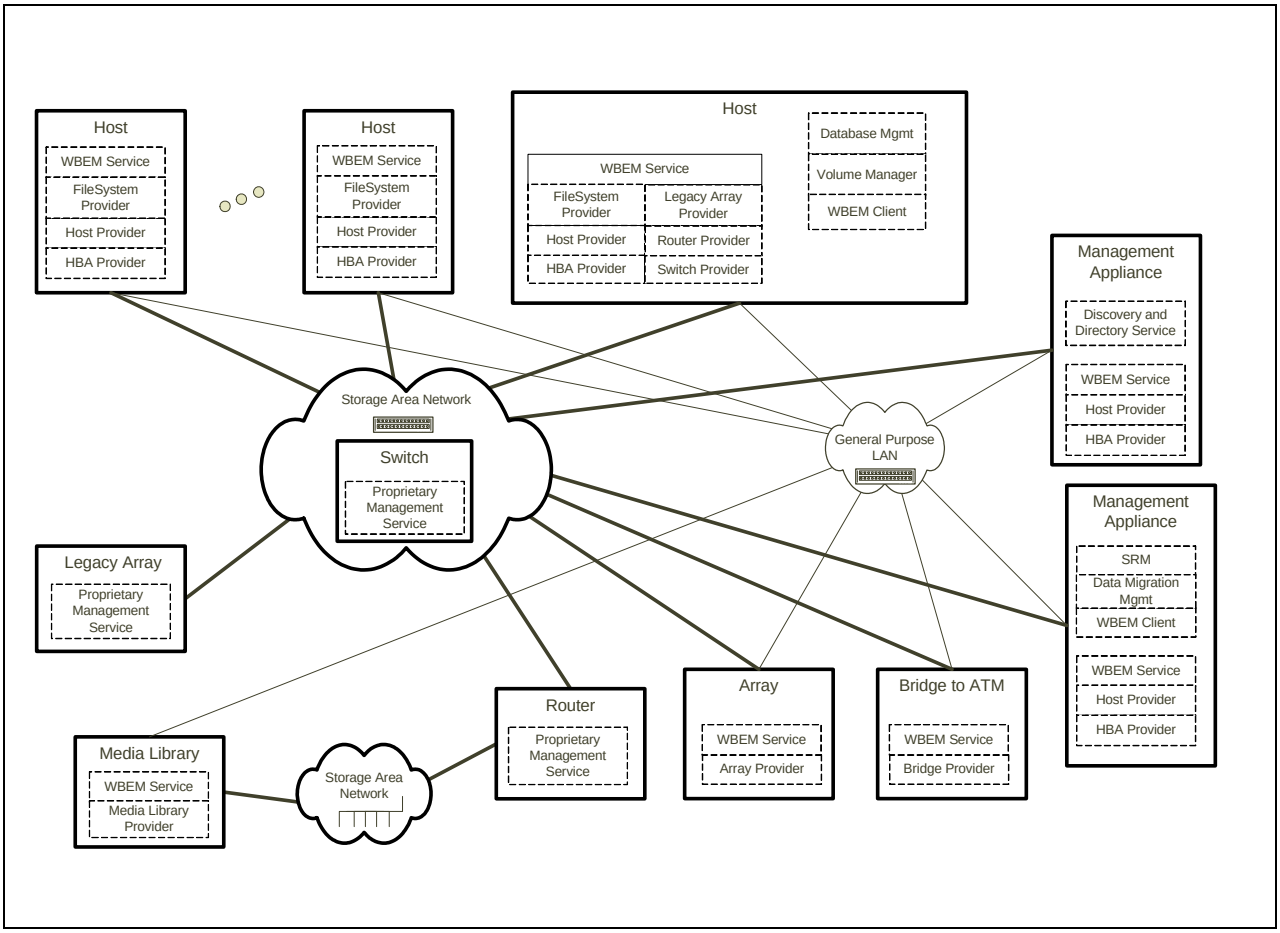


Figure 3 - Example Client Server Distribution in a SAN

4.6 Architectural objectives

The following reflect architectural objectives of the interface. Some of these capabilities are not present in the initial release of the interface, but are inherent in its architecture and intended extensibility. They are intended to provide guidance concerning the present and future direction of development of the Storage Management Initiative Specification.

a) Consistency: State within an object and between objects remains consistent independent of the number of clients simultaneously exerting control, the distribution of objects in the environment, or the management action being performed.

b) Isolation: A client that needs to execute an atomic set of management actions against one or more objects is able to do so in isolation of other clients, who are simultaneously executing management actions against those same objects.

c) Durability: Atomicity, consistency and isolation are preserved independent of the failure of any entity or communication path in the management environment.

d) Consistent Name Space: Managed objects in the SAN adhere to a consistent naming convention independent of state or reliability of any object, device or subsystem in the SAN.

e) Distributed Security: Monitoring and control operations are secure. The architecture supports:

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

1) client authentication;
2) privacy (encryption) of the content of the messages in this protocol;
3) client authorization.

f) Physical Interconnect Independence: The interface functions independent of any particular SAN physical interconnect, supplier or topology.

g) Multi-vendor Interoperability: Clients and servers should use a common communication transport and message/transfer syntax to promote seamless plug compatibility between heterogeneous multi-vendor components that implement the interface.

h) Scalability: The size, physical distribution or heterogeneity of the SAN does not degrade the quality or function of the management interface.

i) Vendor Unique Extension: The interface allows vendors to implement proprietary functionality to distinguish their products and services in the market independent of the release of a new version of the interface.

j) Volatility of State: This interface does not assume that objects are preserved in non-volatile repositories. Clients and servers have the option to preserve object state across failures, but are not required to do so.

k) Replication: This interface provides no support for the automatic replication of object state within the management environment.

l) Functional Layering Independence: The design of this interface is independent of any functional layering a vendor chooses to employ in constructing the storage management systems (hardware and software) necessary to manage a SAN.

m) Asynchronous or Synchronous execution: Management actions have the option to execute either asynchronously or synchronously.

n) Events: This interface provides for the reliable asynchronous delivery of events to one or more registered clients.

o) Cancelable Management Actions: Long running synchronous or asynchronous directives shall be capable of being cancelled by the client. Cancellation shall result in the termination of work by the server and resource consumed being released.

p) Durable Reference: Object classes that persist across power cycles and need to be monitored and controlled independent of SAN reconfiguration (i.e., logical volumes) shall be identified via “Durable Names” to insure consistent reference by clients;

q) Dynamic installation and reconfiguration: New clients and servers shall be capable of being added to or removed from a SMI-S management environment without disrupting the operation of other clients or servers. In most cases, clients will be capable of dynamically managing new servers that have been added to a SMI-S environment.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

5 Overview

5.1 Base capabilities

5.1.1 General

To achieve the architectural objectives and support the key technological trends in clause 4, this document describes an object-oriented, XML-based messaging based interface designed to support the specific requirements of managing devices in and through Storage Area Networks. To quickly become ubiquitous, SMI-S seeks to the greatest extent possible to leverage existing enterprise management specifications like

- the Distributed Management Task Force (DMTF) authored Common Information Model (CIM) and Web Based Enterprise Management (WBEM) standards,
- the international standards on Fibre Channel and SCSI, see bibliography,
- the World-Wide Web Consortium (W3C) for standards on XML,
- the Internet Engineering Task Force (IETF) for standards on HTTP and SLP,
- the standards emerging from the Storage Networking Industry Association (SNIA) on volume and array management.

5.1.2 Object Oriented

A hierarchy of object classes with properties (also known as attributes) and methods (also known as directives) linked via the Universal Modelling Language (UML) modelling constructs of inheritance and associations define most of the capabilities of the SMI-S. Figure 7 provides a simple example of UML using CIM classes for reference. Implementers of this specification are encouraged to consult one of the many publicly available texts on UML or the uml.org web site (www.uml.org) to develop an understanding of UML. A brief tutorial on UML is provided in the introduction material on Object Model in clause 7.

Each SMI-S server in a SAN provides one or more object classes (and related instances) to clients for monitoring and control per Figure 4.

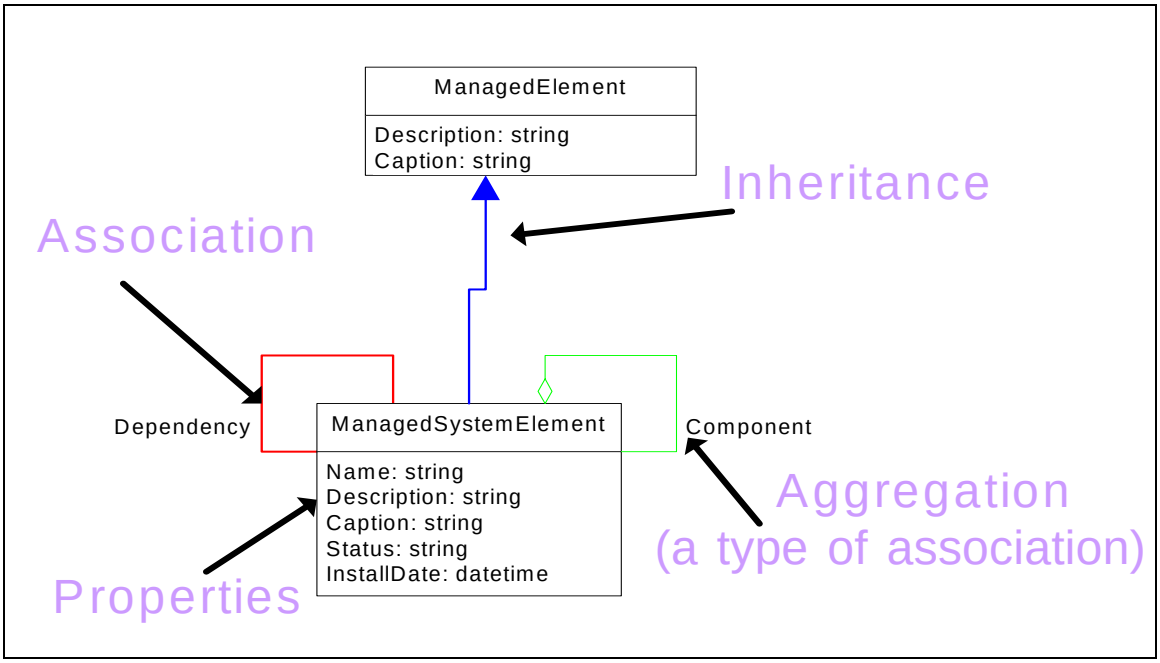


Figure 4 - SMI-S Modelling Conventions

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

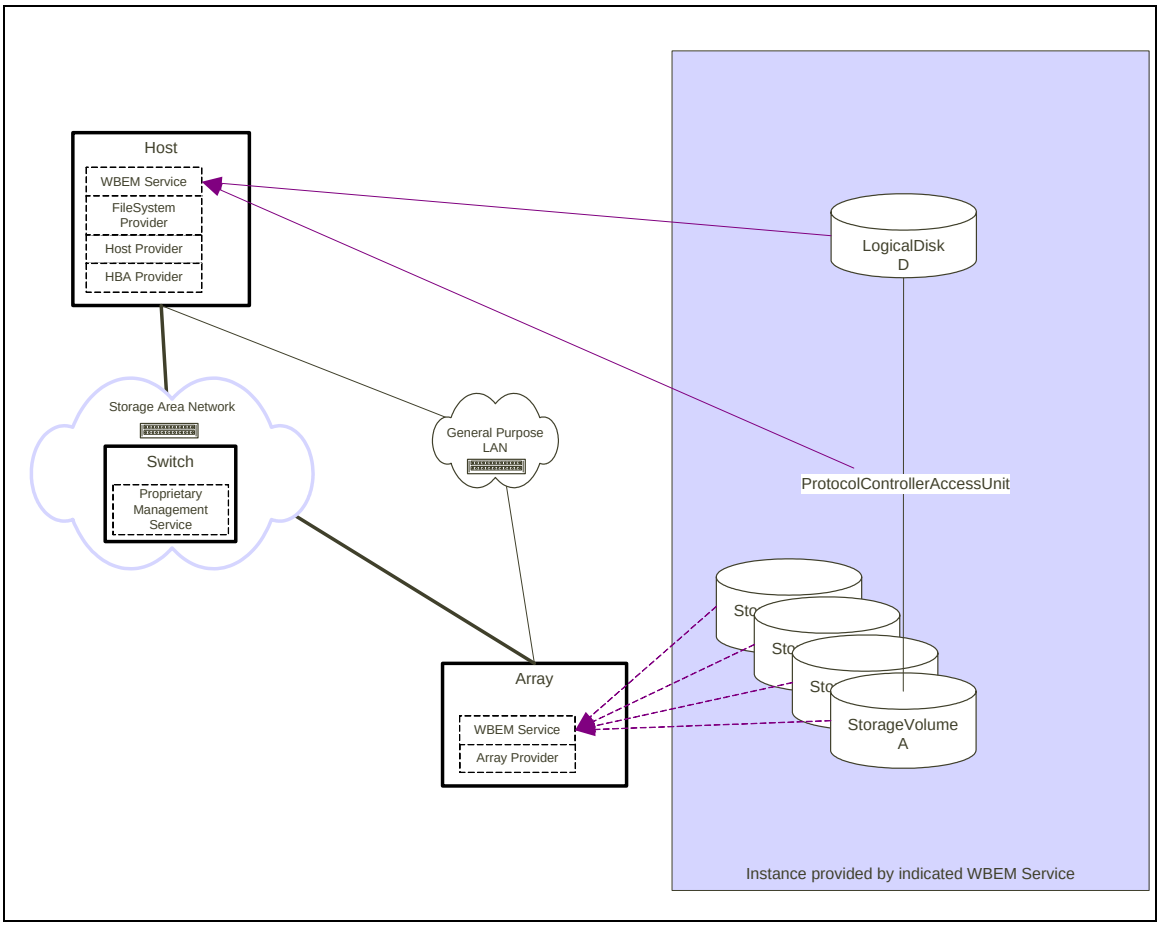


Figure 5 - Object Model/Server Relationship

In Figure 5, a SMI-S client obtains object classes and instances that it can use to manage the storage. At this level of discussion, we have SMI-S conformant WBEM Clients and Servers. The WBEM Servers have providers for the various components that are responsible for the class and association instances that allow the underlying component implementation to be managed.

A standard object oriented interface, together with a standard interface protocol, allows WBEM Clients to discover, monitor and control storage and network devices, regardless of the underlying implementation of those devices.

The goal of this document is to clearly and precisely describe the information expected to be available to a WBEM Client from an SMI-S compliant WBEM Service. It relies upon UML diagrams, easy-to-use tables and machine-readable CIM compliant Managed Object Format (MOF) (through the CIM model maintained at the DMTF). This is intended to ease the task of client implementation and to ease the task of using existing WBEM Servers. It should be noted that the MOF Interface Description Language is a precise representation of the object model in this specification and developers are encouraged to learn this means of expression when implementing this interface. Programmers implementing this interface should reference MOF representations of the object model when faced with implementation decisions.

SMI-S compliant WBEM Servers shall provide instances in a manner conformant to one or more SMI-S profiles (see 7.3.1). The object model supporting these instances can be extended by the vendor as long as it remains conformant to the relevant SMI-S profiles. Generally, a vendor-unique code is required in order for a WBEM client to take advantage of vendor-defined model extensions. However, regardless of the presence of vendor extensions, a generic WBEM client is free to leverage all SMI-S features defined for a supported profile.

--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Figure 6 illustrates this requirement.

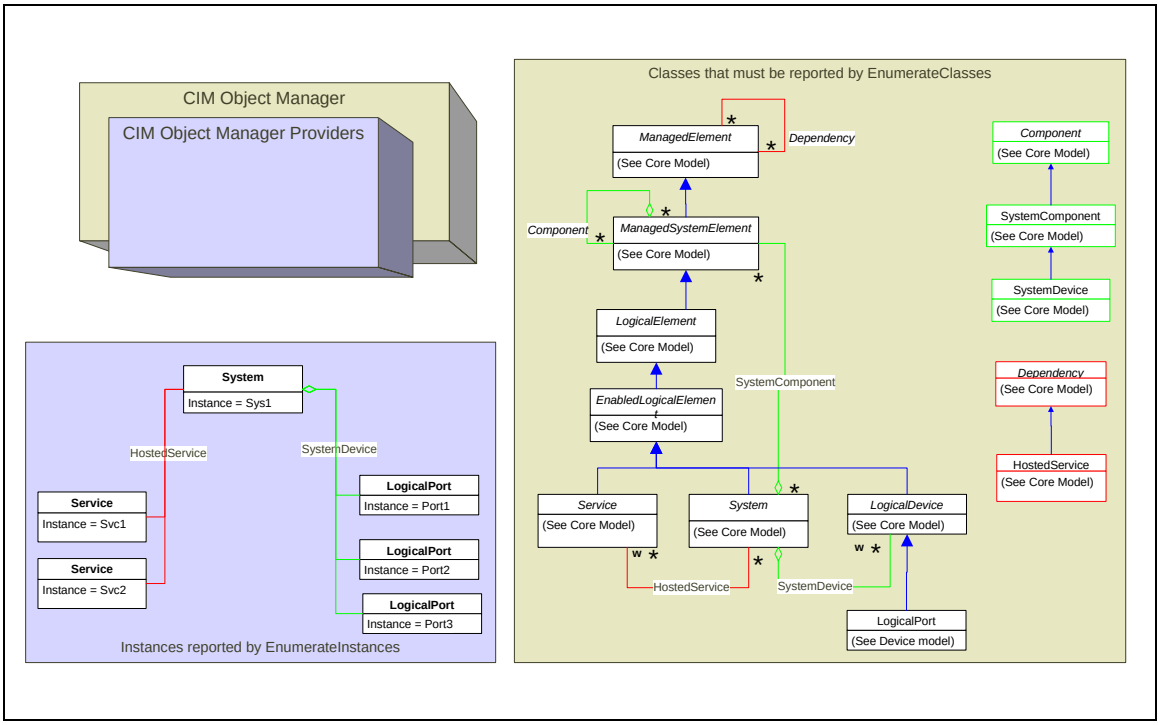


Figure 6 - Canonical inheritance

Figure 6 illustrates that even though a given Fibre Channel Switch only reports instances and allows associated method execution for certain objects, it shall, when asked by a client to enumerate its Object Classes, report the entire hierarchy of classes in its tree. Similarly a server that instantiates an array controller shall report the complete set of object classes that links it to the base canonical object of the SMI-S model. It is this single canonical root that allows any SMI-S client to discover, map and operate upon the complete set of objects in a given SAN.

The object model presented in this specification is intended to facilitate interoperability but not limit the expression of unique features that differentiate manufacturers in the market. For this reason, the object model herein only serves as a ”core” to compel multi-vendor interoperability. In the interest of gaining a competitive advantage, a given vendor’s implementation of the interface is free to include additional object classes, properties, methods, events and associations around this “core”. These vendor-unique extensions to the object model may, in select cases (e.g., extrinsic methods), require the modification of client code above and beyond that required to support the core.

5.1.3 Messaging-based interface

A messaging-based interface, rather than a more traditional procedure call interface, was selected so that platform and language independence could be achieved across the breadth of devices, clients and manufacturers that may implement the interface. This messaging-based environment also eases the task of transporting management actions over different communication transports and protocols that may emerge as the computer industry evolves. An example fragment of an SMI-S CIM-XML message is provided in Figure 7.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
<!DOCTYPE CIM SYSTEM HTTP://www.dmtf.org/cim-v2.dtd/>
<CIMVERSION=”2.0” DTDVersion=”2.0”>
     <CLASS NAME=”ManagedSystemElement”>
        <QUALIFIER NAME=”abstract”></QUALIFIER>
        <PROPERTY NAME=”Caption” TYPE=”string”>
           <QUALIFIER NAME=”MaxLen” TYPE=”sint32”>
              <VALUE>64</VALUE>
           </QUALIFIER>
        </PROPERTY>
        <PROPERTY NAME=”Description” TYPE=”string”></PROPERTY>
        <PROPERTY NAME=”InstallDate” TYPE="datetime”>
           <QUALIFIER NAME=”MappingStrings” TYPE=”string”>
              <VALUE>MIF.DMTF|ComponentID|001.5</VALUE>
           </QUALIFIER>
        </PROPERTY>
        <PROPERTY NAME=”Status” TYPE=”string”>
           <QUALIFIER NAME=”Values” TYPE=”string” ARRAY=”TRUE”>
              <VALUE>OK</VALUE>
              <VALUE>Error</VALUE>
              <VALUE>Degraded</VALUE>
              <VALUE>Unknown</VALUE>
           </QUALIFIER>
        </PROPERTY>
     </CLASS>
</CIM>
```

Figure 7 - Sample CIM-XML Message

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

5.2 Capabilities of this version

5.2.1 Overview

This subclause establishes requirements for this version of the Storage Management Initiative Specification. These requirements are stated as a prioritized list of functional capabilities that are provided by the interface. A compliant WBEM Client shall be able to

a) identify key resources in a SAN;

b) identify interconnects between key resources in a SAN;

c) receive asynchronous notification that the configuration of a SAN has changed;

d) identify the health of key resources in a SAN;

e) receive asynchronous notification that the health of a SAN resource has changed;

f) identify the available performance of interconnects in a SAN;

g) receive asynchronous notification that the performance of a SAN interconnect has changed;

h) identify the zones being enforced in a SAN;

i) create/delete and enable/disable zones in a SAN;

j) identify the storage volumes in a SAN;

k) create/delete/modify storage volumes in a SAN;

l) identify the connectivity and access rights to Storage Volumes in a SAN;

m) create/delete and enable/disable connectivity and access rights to Storage Volumes in a SAN;

n) require the use of authenticated clients.

5.2.2 Determine and monitor the configuration of a SAN

Functional capabilities (a) to (c) allow a client to determine and monitor the configuration of a SAN. The configuration of a SAN consists of the “key resources in a SAN” and the interconnections between them. Functional capabilities (a) and (b) provide a means of establishing a baseline configuration of a SAN and functional capability (c) reduces the need for a client to poll the key resources in a SAN for the purpose of monitoring the configuration of the SAN.

The key resources in a SAN include

- Hosts,
- Management Appliances,
- Interconnection Devices, including switches, routers, etc.,
- Storage Subsystems, including virtualization systems,
- Storage Volumes,
- HBAs and Ports,
- Links,
- Media Libraries,
- Tape Drives.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

NOTE No distinction is made between older (legacy) and newer key resources in a SAN. The Storage Managment Initiative Specification supports both.

5.2.3 Monitoring the health of key resources in a SAN

Functional capabilities (d) and (e) allow a client to monitor the ongoing health of the key resources in a SAN. Functional capability (d) provides a means of establishing a baseline “health” reading for each key resource in a SAN and functional capability (e) reduces the need for a client to poll the key resources in a SAN for the purpose of monitoring their ongoing health.

5.2.4 Monitoring the available performance of interconnections in a SAN

Similarly, functional capabilities (f) and (g) allow a client to monitor the ongoing available performance of interconnections in a SAN. Functional capability #5 provides a means of establishing a baseline available performance reading for each interconnection in a SAN and functional capability (g) reduces the need for a client to poll the available performance of interconnections in a SAN for the purpose of monitoring their ongoing health.

5.2.5 Monitoring and controlling the zones in a SAN

Functional capabilities (h) and (i) allow a client to monitor and exercise control over the zones in a SAN. This is the only functional capability that is specific to Fibre Channel SANs. (It is recognized that this version of the Storage Managment Initiative Specification only handles Fibre Channel SANs.)

5.2.6 Discovering/monitoring/controlling the storage volumes in a SAN

Functional capabilities (j) and (k) allow a client to discover, monitor and exercise control over the storage volumes. Functional capability (j) provides a way for clients to discover existing storage volumes in a SAN, and functional capability (k) provides a means of changing the population and/or characteristics of storage volumes in a SAN. The storage pools within storage arrays are not specifically handled by functional capabilities here, since any necessary handling of such storage pools is implied by functional capability (k).

Functional capabilities (l) and (m) allow a client to discover, monitor and exercise control over the access of storage volumes in a SAN by other systems within the SAN.

5.2.7 Requiring authenticated clients in a SAN

Functional capability (n) allows a site to require clients to be authenticated before they are able to access Server Management Interface Specification functionality.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

5.3 Operational environment

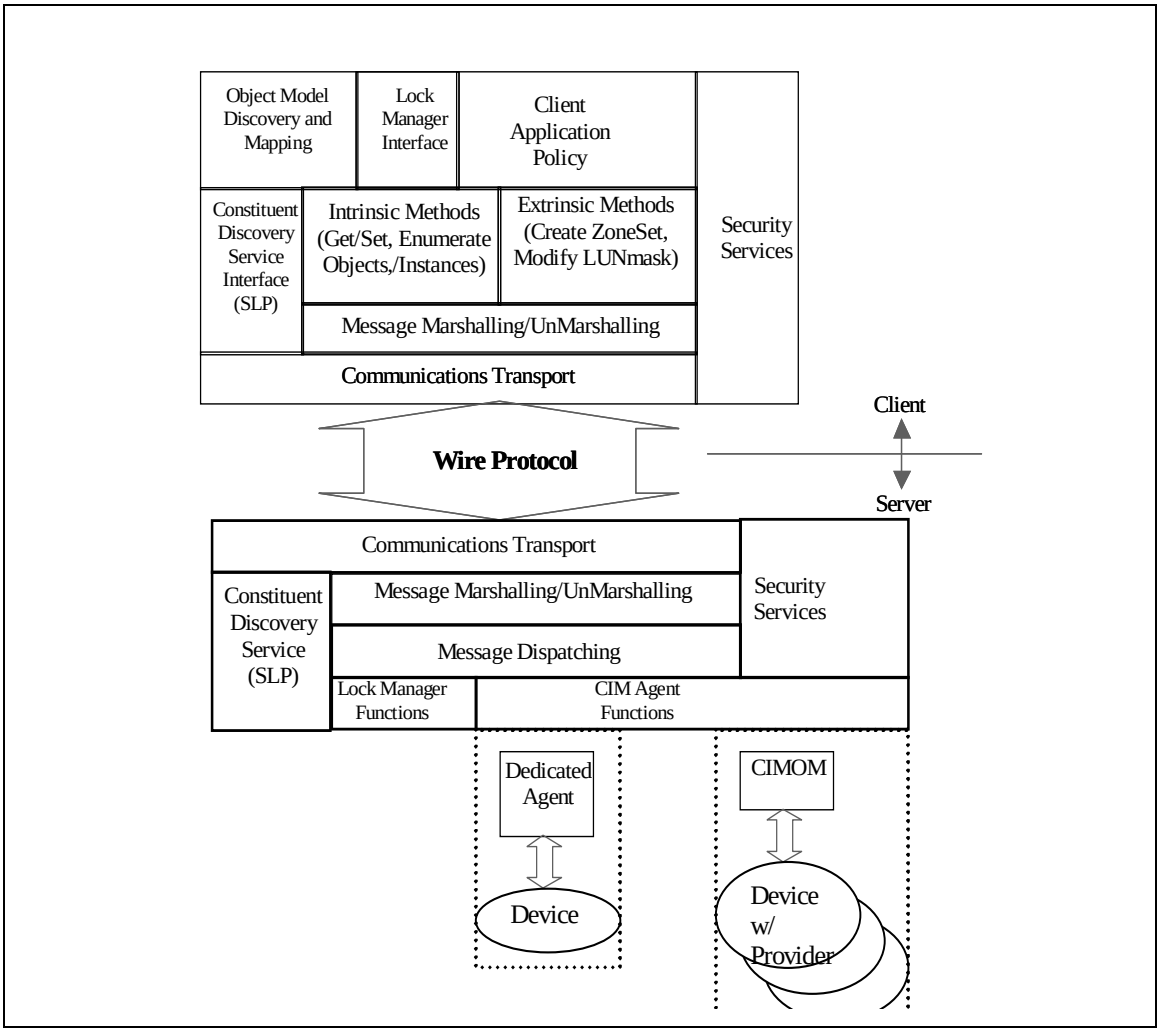


Figure 8 - Operational environment

Figure 8 illustrates activities that either clients or servers need to account for to provide facilities to support

- the discovery of constituents in the managed environment,
- the discovery of object classes as well as related associations, properties, methods, indications and return status codes that are provided by servers in the managed environment,
- the security or resources and communication in the environment,
- the locking of resources in the presence of non-cooperating clients,
- the marshalling/un-marshalling of communication messages,
- the execution of basic methods that are “intrinsic” to the construction, traversal and management of the object model provided by the distributed servers in a SAN,
- the execution of object specific “extrinsic” methods that provide clients the ability to change the state of entities in the SAN.

In addition, to facilitate ease of installation, startup, expansion and upgrade requirements for implementations are specified for the developers of clients and servers.

--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

5.4 Using this standard

This International Standard is insufficient as a single resource for the developers of SMI-S clients and servers. Developers are encouraged to first read the DMTF specifications on CIM, CIM Operations over HTTP and CIM-XML as well as obtaining familiarity with UML and the IETF specification on Service Location Protocol (SLP).

A developer implementing SMI-S clients/servers should read this standard in sequence noting that clause 7, “Object model” is intended principally as a reference relative to the particular device type that is being provided or managed in a SMI-S environment.

5.5 Language bindings

As a messaging interface this standard places no explicit requirements for syntax or grammar on the procedure call mechanisms employed to convert SMI-S messages into semantics consumable by modern programming languages. The syntax and grammar used to express the semantics is left at the discretion of each SMI-S developer.

Several open-source sources are available for programmers who wish to streamline the task of parsing SMI-S messages into traditional procedure call semantics and using the semantics to store object instances. Consult the WBEMsource initiative (http://wbemsource.org) for current language bindings available to implement the SMI-S interface.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

6 Transport and reference model

6.1 Introduction

6.1.1 Overview

The interoperable management of storage devices and network elements in a distributed storage network requires a common transport for communicating management information between constituents of the management system. This clause details the design of this transport, as well as the roles and responsibilities of constituents that use the common transport (i.e., a reference model).

6.1.2 Language requirements

To express management information across the interface a language is needed that

- can contain platform independent data structures,
- is self-describing and easy to debug,
- can be extended easily for future needs.

The World-Wide Web Consortium's (W3C) Extensible Markup Language (XML) was chosen for the language to express management information and related operations, as it meets the requirements above.

6.1.3 Communications requirements

Communications protocols to carry the XML based management information are needed that

- can take advantage of the existing ubiquitous IP protocol infrastructures,
- can be made to traverse inter- and intra-organizational firewalls,
- can easily be embedded in low cost devices.

The Hyper Text Transport Protocol (HTTP) was chosen for the messaging protocol and TCP was chosen for the base transfer protocol to carry the XML management information for this interface as it meets the requirements above.

6.1.4 XML Message Syntax and Semantics

In order to be successful, the expression of XML management information (messages) across this interface shall follow consistent rules for Semantics and Syntax. These rules are of sufficient quality, extensibility and completeness to allow their wide adoption by storage vendors and management software vendors in the industry. In addition, to facilitate rapid adoption, existing software that can parse, marshal, un-marshal and interpret these XML messages should be widely available in the market such that vendor implementations of the interface are accelerated. The message syntax and semantics selected should

- be available on multiple platforms,
- have software implementations that are Open source (i.e., collaborative code base),
- have software implementations available in Java and C++,
- leverage industry standards, where applicable,
- conform with W3C standards for XML use,
- be object model independent (i.e., be able to express any object model).

Virtually the only existing industry standard in this area is the WBEM standards *CIM Operations over HTTP* and *Specification for the Representation of CIM in XML* as developed and maintained by the DMTF. The WBEM source initiative is a collaboration of open source implementations which can be leveraged by storage vendors to prototype, validate and implement this interface in products. Specifically designed for transporting object model independent management information, the CIM-XML message syntax was chosen because it

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

meets the requirements of the storage industry as enumerated above. This specification augments the capabilities of CIM-XML in the area of discovery to facilitate ease of management.

6.2 Transport stack

The complete transport stack for this interface is illustrated below in Figure 9. It is the primary objective of this interface to drive seamless interoperability across vendors as communications technology and the object model underlying this interface evolves in time. Thus, it should be noted that the transport stack has been layered in such a way that, if required, other protocols can be added as technology evolves. For example, should Simple Object Access Protocol (SOAP) or Internet Inter-Orb Protocol (IIOP) become prominent the content in the stack below can be expanded with minimal changes to existing product implementations in the market.

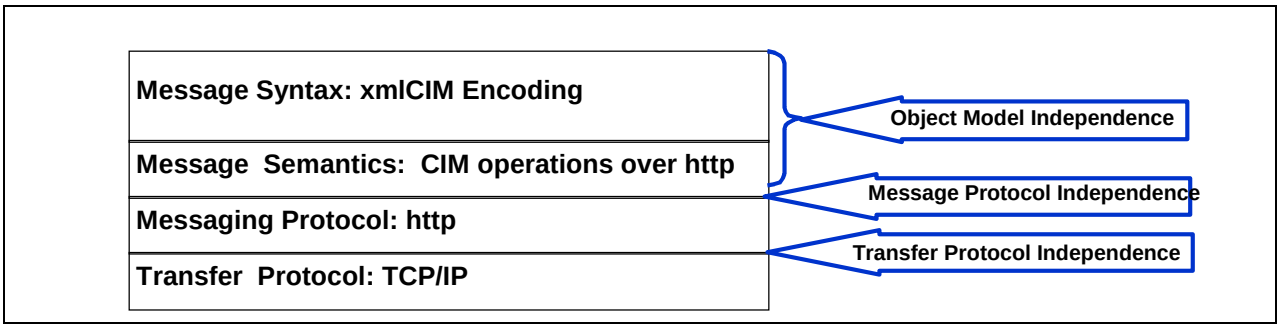


Figure 9 - Transport stack

Again, this interface uses two specifications from the DMTF to fully implement the message syntax and semantics for this interface.

The first specification, *CIM Operations over HTTP*, Version 1.1 details a basic set of directives (Semantics) needed to manage any schema over HTTP. The requirement for this basic set of directives is common to nearly all management frameworks (e.g., create object, delete object, create instance and delete instance). This class of directive is referred to in this document as “intrinsic methods”. *CIM Operations over HTTP* also provides a client the ability to execute directives that are unique to the specification of a particular object class within a schema (example: chop <method>, apple <object-class>). This class of directive is referred to in this standard as “extrinsic methods”.

The second specification, *Specification for the Representation of CIM in XML, Version 2.1* details the precise W3C compliant syntax and grammar for encoding CIM into XML.

While some vendors may choose alternate transfer and message protocols for unique implementations, the transport stack elements listed above shall be implemented for conformance with this standard.

It should be noted that this standard places no restriction on the physical network selected to carry this transport stack. For example, a vendor can choose to use in-band communication over Fibre-channel as the backbone for this interface. Another vendor could exclusively (and wisely) choose out-of-band communication over Ethernet to implement this management interface. Additionally, select vendors could choose a mix of in-band and out-of-band physical network to carry this transport stack.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

6.3 Reference model

6.3.1 Overview

As shown below in Figure 10, the Reference model shows all possible constituents of the management environment in the presence of the transport stack for this interface.

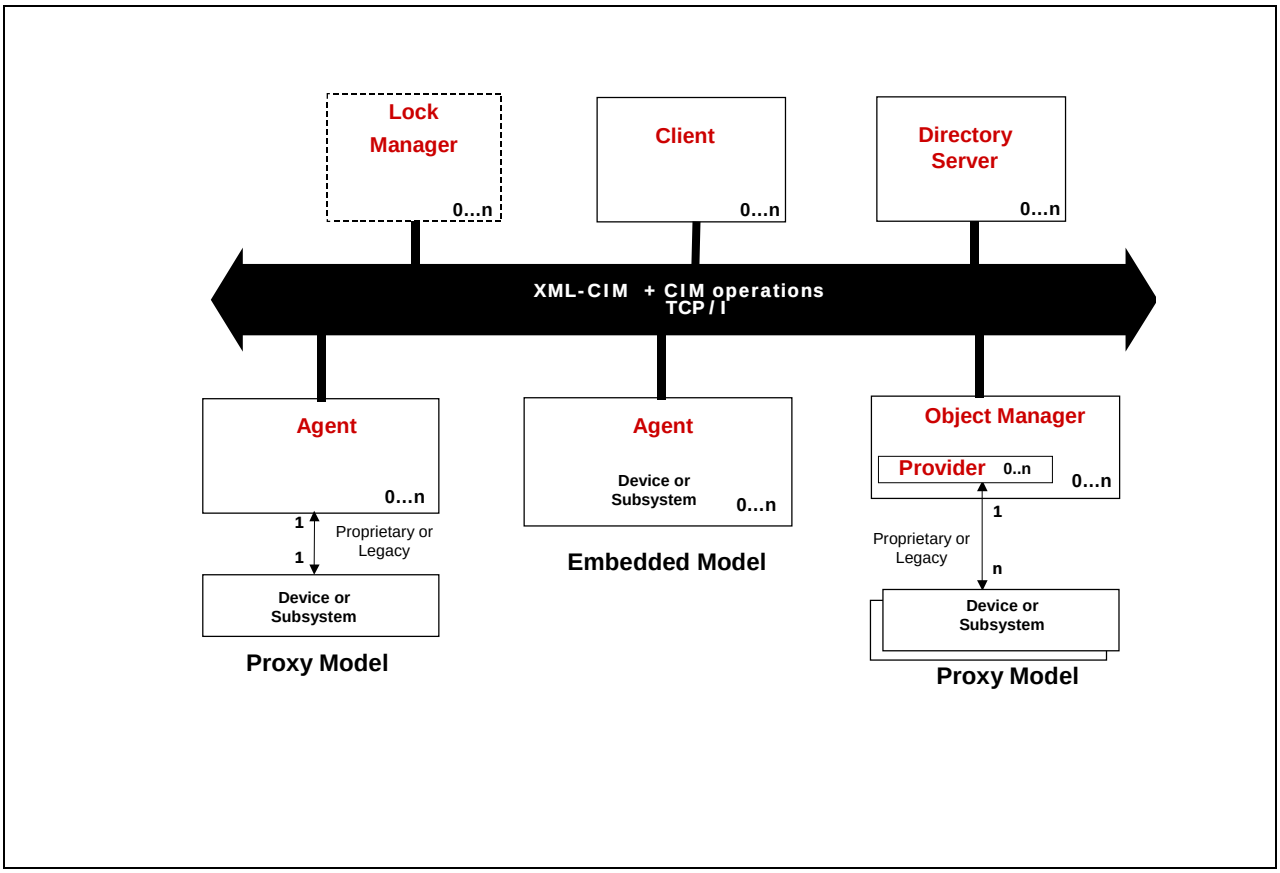


Figure 10 - Reference Mode

Figure 10 illustrates that the transport for this interface uses CIM Operations over HTTP with xm/CIM encoding and HTTP/TCP/IP to execute intrinsic and extrinsic methods aganist the schema for this interface.

NOTE It is envisioned that a more complete version of this reference model would include the Lock Manager. However, the Lock Manager in **SMI-S** Release 1 is preliminary and subject to change. As a result, it is shown as a dotted box to illustrate where the role would fit.

6.3.2 Roles for Interface Constituents

6.3.2.1 Client

A Client is the consumer of the management information in the environment. It provides an API (language binding in Java or C++, for example) for overlying management applications (like backup engines, graphical presentation frameworks and volume managers) to use.

6.3.2.2 Agent

An agent is a CIM Server. It shall implement those functional profiles, as defined in the DMTF specifications, necessary to satisfy the SMI-S profile with which it conforms. Often, an agent only controls only one device or subsystem and is incapable of providing support for complex intrinsic methods like schema traversal. An agent can be embedded in a device (like a Fibre Channel Switch) or provide a proxy on a host that communicates to a device over a legacy or proprietary interconnect (like a SCSI based array controller).

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Embedding an agent directly in a device or subsystem reduces the management overhead of a customer and eliminates the requirement for a stand-alone host (running the proxy agent) to support the device.

Embedded agents are the desired implementation for “plug and play” support in an SMI-S managed environment. However, proxy agents are a practical concession to the legacy devices that are already deployed in storage networked environments. In either case, the minimum CIM support for agents applies to either agent deployments.

6.3.2.3 CIM Server

A CIM Server is an object manager that serves management information from one or more devices or underlying subsystems through providers. As such, an Object Manager is an aggregator that enables proxy access to devices/subsystems and can perform more complex operations like schema traversals. An object manager typically includes a standard provider interface to which device vendors adapt legacy or proprietary product implementations.

6.3.2.4 Provider

A provider expresses management information for a given resource such as a storage device or subsystem exclusively to a CIM Server. The resource can be local to the host that runs the Object Manager and be remotely accessed through a distributed systems interconnect.

6.3.2.5 Lock Manager

This version of the specification does not support a lock manager.

6.3.2.6 Directory Server

A directory server provides a common service for use by clients for locating services in the management environment.

6.3.3 Cascaded Agents

This standard discusses constituents in the SMI-S environment in the context of Clients and Servers (Agents and Object Managers). This version of the specification does not allow constituents in a SMI-S management environment to function as both client and server.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7 Object model

7.1 Model overview (key resources)

7.1.1 Overview

The SMI-S object model is based on the Common Information Model (CIM), developed by the DMTF. Version 1 SMI-S Object Model is based on the 2.8 revision of the CIM schema. For a more complete discussion of the full functionality of CIM and its modelling approach, see http://www.dmtf.org/standards/cim).

Readers seeking a more complete understanding of the assumptions, standards and tools that assisted in the creation of the SMI-S object model are encouraged to review the following:

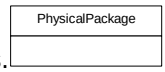
- CIM Tutorial
  (http://www.dmtf.org/education/tutorials/)
- CIM UML Diagrams and MOFs
  (http://www.dmtf.org/standards/cim)
- CIM System / Device Working Group Modeling Storage
  (http://www.dmtf.org/standards/published_documents)

Managed Object File (MOF) is a way to describe CIM object definitions in a textual form. A MOF can be encoded in either Unicode of UTF-8. A MOF can be used as input into an MOF editor, parser or compiler for use in an application.

The SMI-S model is divided into several *profiles*, each of which describes a particular class of SAN entity (such as disk arrays or Fibre Channel Switches). These profiles allow for differences in implementations but provide a consistent approach for clients to discover and manage SAN resources. IN DMTF parlance, a *provider* is the instrumentation logic for a profile. In many implementations, providers operate in context of a *CIM Server* that is the infrastructure for a collection of providers. A WBEM *client* interacts with one or more WBEM Servers.

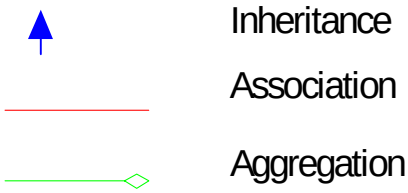
7.1.2 Introduction to CIM UML Notation

CIM diagrams use a subset of Unified Modelling Language (UML) notation.

Most classes are depicted in rectangles. 

. The class name is in the upper part and *properties* (also known as *attributes* or *fields*) are listed in the lower part. A third subdivision is added for *methods,* if they are included. A special type of class, called an *association*, is used to describe the relationship between two or more CIM classes.

Three types of lines connect classes.



The CIM documents generally follow the convention of using blue arrows for inheritance, red lines for associations and **green** lines for **aggregation**. The color-coding makes large diagrams much easier to read but is not a part of the UML standard.

--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

The ends of some associations have numbers (cardinality) indicating the valid count of object instances. Cardinality is expressed either as a single value (such as 1) or a range of values (0..1 or 1..4);"*" is shorthand for 0..n.

Some associations and aggregations are marked with a "W" at one end indicating that the identity of this class depends on the class at the other end of the association. For example, fans may not have world-wide unique identifiers. They are typically identified relative to a chassis.

This document uses two other UML conventions.

The UML Package symbol 

is used as a shortcut representing a group of classes that work together as an entity. For example, several classes model different aspects of a disk drive. After the initial explanation of these objects, a single disk package symbol is used to represent the entire group of objects.

Schema diagrams include all of a profile's classes and associations. The class hierarchy is included and each class is depicted one time in the schema diagram. Instance diagrams also contain classes and associations, but represent a particular configuration. Multiple instances of an object may be depicted in an instance diagram. An instance may be named with an instance name followed by a colon and a class name (underlined). For example,



represent an array and a switch – two instances of <COMPUTER SYSTEM> objects.

7.2 Techniques

7.2.1 CIM fundamentals

This subclause provides a rudimentary introduction to some of the modelling techniques used in CIM and is intended to speed understanding of the SMI-S object model.

Associations as Classes

CIM presents relationships between objects with specialized classes called *associations* and *aggregations.* In addition to references to the related objects, the association or aggregations may also contain domain-related properties. For example, "ControlledBy" associates a controller and a device. There is a many-to-many cardinality between controllers and devices (i.e., a controller may control multiple devices and multi-path devices connect to multiple controllers), each controller/device connection has a separate activity state. This state corresponds to the AcccessState property of "ControlledBy" association linking the device and the controller.

Logical and Physical Views

CIM separates physical and logical views of a system component, and represents them as different objects – the "realizes" association ties these logical and physical objects together.

Identity

Different agents may each have information about the same organic object and may need to instantiate different model objects representing the same thing. Access control is one example: a switch zone defines which host device ports may access a device port. The switch agent creates partially populated port objects that are also created by the HBA and storage system agents. The ConcreteIdentity association is used to indicate the associated object instances are the same thing. ConcreteIdentity is also used as a language-independent alternative to multiple inheritance. For example, a FibreChannel port inherits from a generic port and also has properties of a SCSI controller. CIM models this as "FCPort" and "ProtocolController" objects associated by ConcreteIdentity.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Extensibility

CIM makes allowances for additional values in enumerations that were not specified in the class Derivation by adding a property to hold arbitrary additional values for an enumeration. This property is usually named OtherXXXX (where XXXX is the name of the enumeration property) and specifying “other” as the value in the enumeration property indicates its use. For an example, see the ConnectorType and OtherTypeDescription properties of CIM_Slot in the CIM_Physical MOF.

Value/ValueMap Arrays

CIM uses a pair of arrays to represent enumerated types. ValueMap is an array of integers. Values is an array of strings that map to the equivalent entry in ValueMap. For example, PrinterStatus (in the CIM_Device MOF) is defined as follows:

```
ValueMap {“1”, “2”, “3”, “4”, “5”, “6”, “7”},
Values {“Other”, “Unknown”, “Idle”, “Printing”, “Warm-up,
“Stopped Printing”, “Offline”},
```

A status value of 6 means “Stopped Printing”. A client application can automatically convert the integer status value to a human-readable message using this information from the MOF.

Return Codes

When a class definition includes a method, the MOF includes Value/ValueMap arrays representing the possible return codes. These values are partitioned into ranges of values. Values from 0 to 0x1000 are used for return codes that may be common to various methods. Interoperable values that are specific to a method start at 0x1001, and vendor-specific values may be defined starting at 0x8000. Here’s an example of return codes for starting a storage volume.

```
 ValueMap {“0”, “1”, “2”, “4”, “5”, “.”, “0x1000”,
“0x1001”, “…”, “0x8000..”},
 Values {“Success”, “Not Supported”, “Unknown”, “Time-out,
“Failed”, “Invalid Parameter”, “DMTF_Reserved”,
“Method parameters checked - job started”,
“Size not supported”,
“Method_Reserved”, "Vendor_ Specific"}]
```

Model Conventions

This is a summary of objects and associations that are common to multiple profiles.

ComputerSystem: Most SAN products are modelled as ComputerSystem. The term “cluster” is used for systems with multiple loosely coupled processors, the individual processors known as “component” ComputerSystems. A cluster and each of its component nodes are modelled with ComputerSystem. ComponentCS associates the top-level ComputerSystem with each node

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

ComputerSystem. A ComputerSystem's dedicated property describes the functions provided by a system (e.g., host, storage system, switch).

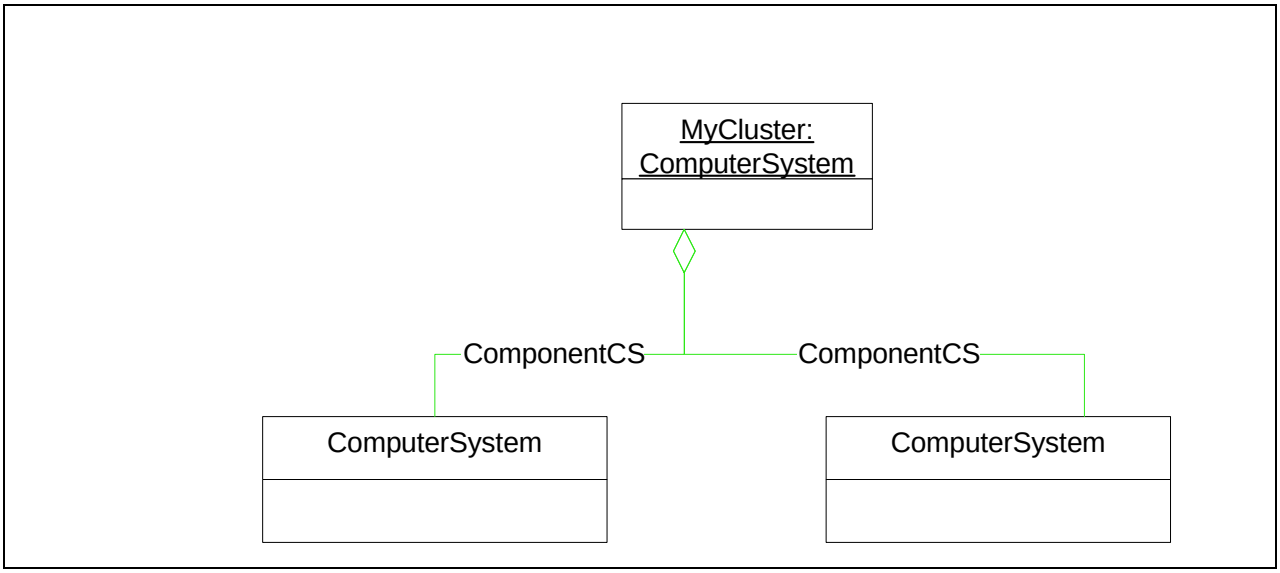


Figure 11 - Cluster model

"PhysicalPackage" represents the physical storage product. "PhysicalPackage" is sometimes subclassed to "ChangerDevice", but "PhysicalPackage" accommodates products deployed in multiple chassis.

"Product" models asset information including vendor and product names. **"**Product" is associated with "PhysicalPackage".

"SoftwareElement" models firmware and optional software packages. "InstalledSoftwareElement" associates "SoftwareElement" and "ComputerSystem", "DeviceSoftware" associates "SoftwareElement" and "LogicalDevice"s (a superclass of devices and ports).

"Service" models a configuration interface (for example, a switch zoning service or an array access control service). Services typically have methods and properties describing the capabilities of the service. A storage system may have multiple services. For example, an array may have separate services for LUN Masking and LUN creation. A client can test for the existence of a named service to see if the agent is providing this capability.

"LogicalDevice" is a superclass with device subclasses (e.g., DiskDrive and TapeDrive) and also intermediate nodes like Controller and FCPort. Each LogicalDevice subclass shall be associated to a ComputerSystem with a SystemDevice aggregation. Due to the large number of LogicalDevice subclasses, SystemDevice aggregations are often omitted in instance diagrams in this standard.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

The diagram (see figure 12) combines these common elements. This combination is used in several of the profiles.

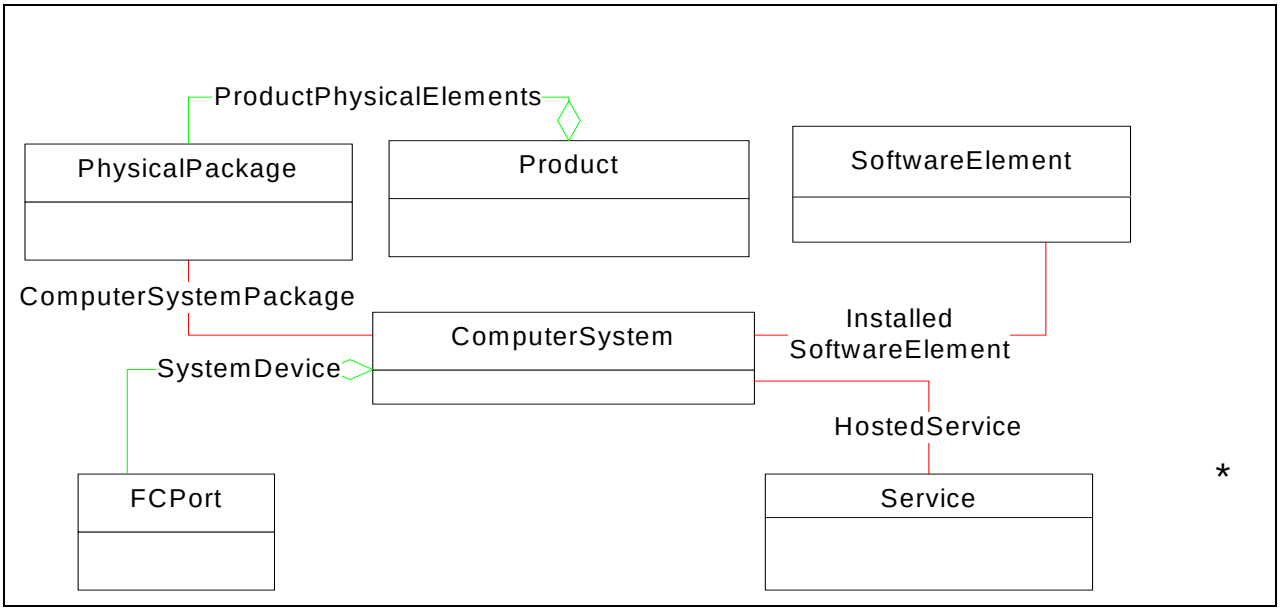


Figure 12 - Common elements

This standard covers many common storage models and management interfaces, but some implementations include other objects and associations not detailed in this standard. In some cases, these objects and associations are modelled by CIM schema elements, not covered by this document. When vendor-specific capabilities are needed, modelling subclasses of CIM objects is preferable. These subclasses are allowed to contain vendor-specific properties and methods and vendor-specific associations to other classes.

7.2.2 Modelling profiles

In addition to modelling SAN components, SMI-S servers shall model the profiles they provide. This piece of information is used two ways:

- clients can quickly determine which profiles are available;
- an SLP component can query the SMI-S Server and automatically determine the appropriate SLP Service Template information (see clause 9 and Table 2)

Table 2 - SLP properties

Property Name	Use
SupportedRegisteredProfiles	Defines the organization defining the profile, the RegisteredProfile and RegisteredSubprofile. Setting this to “SNIA” indicates that one of the SNIA SMI-S profiles applies.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

A client can traverse the Server Profile in each SMI-S server to see which Profiles (and objects) claim SMI-S compliance.

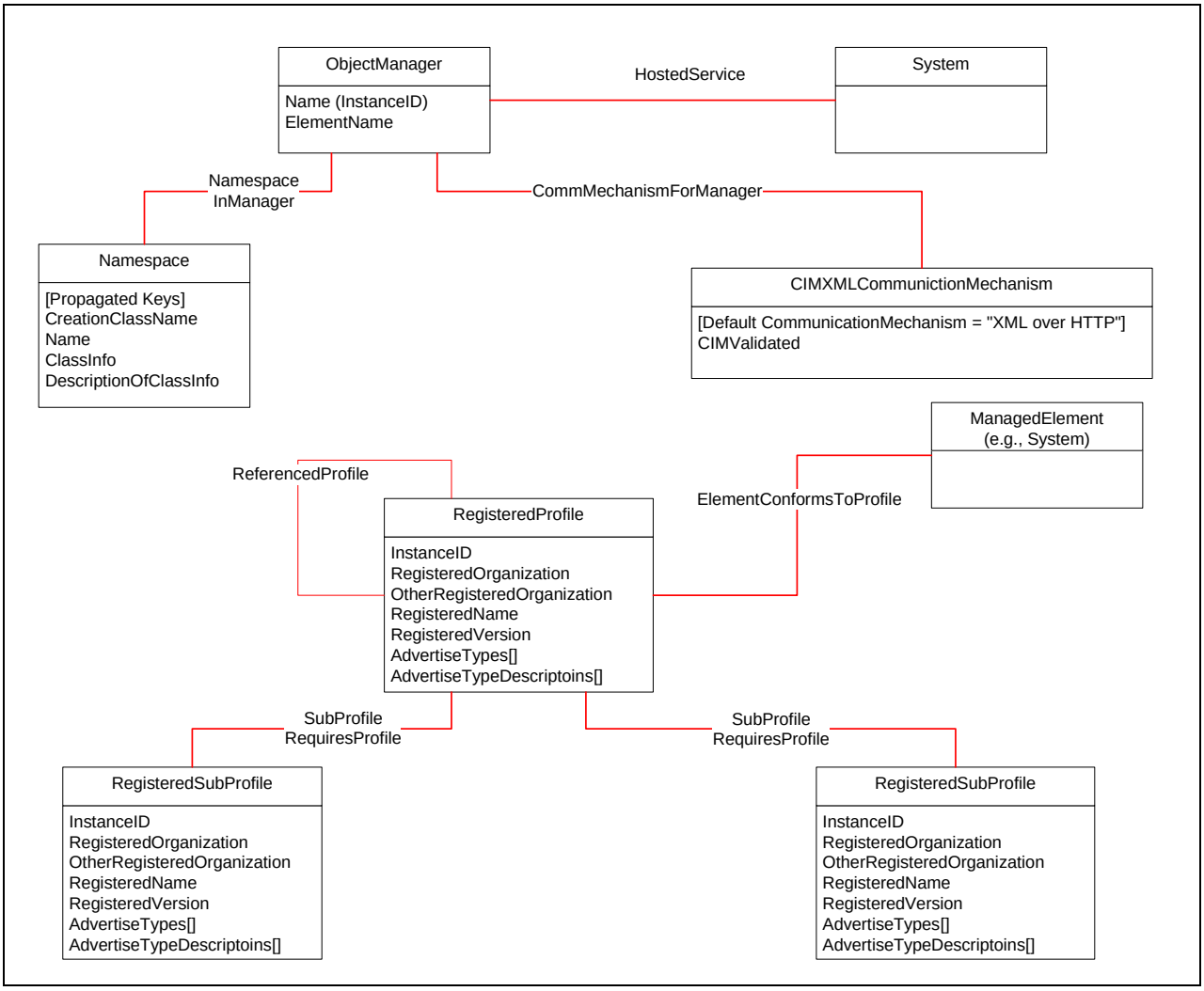


Figure 13 - Server Profile Instance Diagram

The RegisteredProfile describes the profiles that a CIM Server claims are supported. The RegisteredSubprofile is used to define the optional features supported by the system being modelled. A client can traverse the associations in the Server Profile, see which Profiles and subprofiles claim SMI-S compliance.

7.2.3 Naming

There can be multiple SMI-S Servers in any given storage network environment. It is not sufficient to think of the name of an object as just the combination of its key values. The name also serves to identify the Server that is responsible for the object. The name of an object (instance) consists of the Namespace path and the Model path. The Namespace path provides access to a specific SMI-S server implementation and is used to locate a particular namespace within a Server. The Model path provides full navigation within the CIM Schema and is the concatenation of the class name and key-qualified properties and values.

The namespace has special rules. Optimally, a namespace will uniquely identify a SMI-S Server. However, a SMI-S Server that supports multiple namespaces is acceptable. How an implementation defines Namespaces within a SMI-S server is not restricted. However, SMI-S implementations that manage all objects within a Profile in one Namespace simplify interoperability.

--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.2.4 Correlatable and Durable Names

7.2.4.1 Overview

Management applications need to read and write information about managed objects in multiple CIM namespaces. When an object in one namespace is associated with an object in another namespace, each namespace represents some amount of information about the same managed resource using different objects. A management application needs a way to understand when objects in different namespaces represent the same managed resource. A unique common identifier, referred to as a **correlatable name**, is designated as a required property for any objects representing managed resources that might be "seen" from multiple points of view. These durable names are used by a management application for object coordination

A related concept is **durability**. Some names may be correlatable at a particular point in time, but may change over time. An example of a durable name is a hardware-assigned port or volume name. An example of a correlatable, non-durable ID is an IP address. In a DHCP environemnt, IP addresses may change. No name is infinitely durable, even a name derived from hardware may change due to FRU replacement. A client application can generally expect that a stored durable name will be valid at a later time; a non-durable name is less likely to be valid over time.

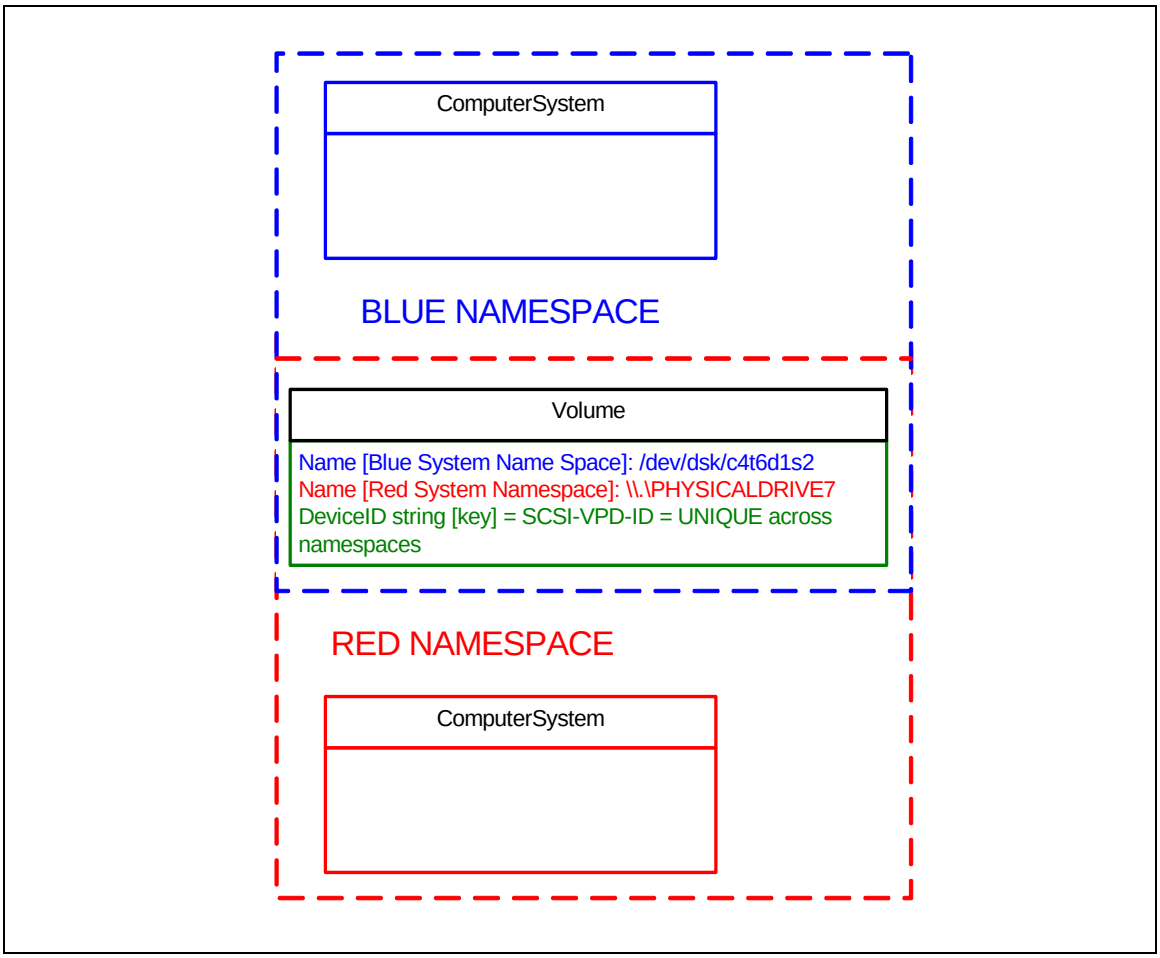


Figure 14 - Volume shared across namespaces

Correlatable names thus provide a means of reliably "stitching together" information from multiple sources about the same managed resource in a SAN. They also provide a means of stitching together information obtained at different points in time, such as when a managed resource is returned to a SAN after having been removed for some period of time.

--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

A necessary technique associated with correlatable names involves the use of NameFormat or NameNamespace properties in some classes. CIM key-value combinations are unique across all instances of a class within a single namespace, but CIM does not fully address cases where different types of identifiers are possible on different instances of an object. It is therefore necessary to ensure that multiple sources of information about managed resources use the same approach for forming correlatable names whenever different types of identifiers are possible.

When different types of identifiers are possible, the profile or subprofile text describes the possible name formats and namespaces for durable and correlatable IDS, the preferred order that each implementation shall use if multiple namespaces are available and the related properties that a client will use to determine the namespace.

Correlatable, durable names shall be implemented for the following objects.

- StorageVolume
- FCPort
- Fabric (AdminDomain)
- ComputerSystem objects with the following roles
    - Host
    - Management Appliance
    - CIM Server
    - Switch
    - Router
    - Bridge
    - Extender
    - Block Server
    - Virtualization Appliance
    - StorageLibrary Server
    - Enclosure Service

Note that CIM keys and correlatable names are not tightly coupled. For some classes, they may be the same thing, but this is not required as long as all correlatable names are unique and management applications can determine when objects in different namespaces are providing information about the same managed resource.

Storage Virtualization crosses different namespaces: Host virtualization layer may provide Logical Volumes that are based on Storage Volumes exported by a Virtualization Appliance that, in turn, may use Storage Volumes exported by RAID array. Management applications and clients can use the correlatable names defined in this subclause for unique identification of objects that cross namespace boundaries.

The common types of information used for correlatable names include SCSI Device Identifiers from the Inquiry Vital Product Data Page 83, Fibre Channel World-Wide Names, Fully Qualified Domain Names and IP Address information. The details for each class requiring correlatable names are provided in the correlatble names subclause of each Profile defined in this specification.

An overview of the information used to form correlatable names for objects is as follows.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.2.4.2 Correlatable Names Formation

- StorageVolume: **StorageVolume.Name** - Multiple valid identifier types exist; the **NameFormat** property shall be used. For Array and Virtualization system exported Storage Volumes, correlatable names are based upon SCSI VPD page information
- FCPort: **FCPort.PermanentAddress** - Usually the Fibre Channel World-Wide Name for the port.
- Fabric: **AdminDomain.Name** - Multiple valid identifier types exist and **NameFormat** shall be used. In Fibre Channel, this name is based on World-Wide Name of the principal switch. Note that this correlatable name can change under some circumstances, such as when the Fibre Channel fabric is partitioned or when the principal switch in a fabric fails.
- PhysicalPackage: Concatenation of PhysicalPackage properties: Manufacturer, Model and SerialNumber. See Table 3, “Standardized Name Formats,” for required format of this concatenated durable, correlatable identifier.
- ComputerSystem (roles “Switch”, “Router”, “Bridge”, “Extender”): **ComputerSystem.Name** - Multiple valid identifier types exist and **NameFormat** shall be used. Correlatable names are based upon a unique identifier native to the interconnect system. For systems supporting FibreChannel, this could be a Fibre Channel World-Wide Name.
- ComputerSystem: (roles “Block Server”, “StorageLibrary”, “Enclosure Service” (SES)): **ComputerSystem.Name** - Multiple valid identifier types exist and **NameFormat** can be used. Correlatable names are based upon NAA or IP Address information.
- ComputerSystem: (roles “Host”, “Virtualization Appliance”, “CIM Server”): **ComputerSystem.Name** - Multiple valid identifier types exist and **NameFormat** can be used. Correlatable names shall be based upon fully-qualified domain name (DNSName) or IP Address information. (See Table 3, “Standardized Name Formats,”). Note that these IDs can be administratively changed. A ComputerSystemPackage association with an appropriate PhysicalPackage or subclass can be used to satisfy asset management use cases (see PhysicalPackage above).

7.2.4.3 Testing equality of Correlatable Names

You can only compare objects of the same class (or parent class). For objects that do not require the use of additional properties, a simple direct comparison is sufficient, providing the format for the required correlatable name (identified in this subclause or the specific profile) is adhered to.

For objects that do require the use of additional properties such as NameFormat, the correlatable names of objects representing the same entity should compare positively, negatively or indicate clearly when a comparison is ambiguous

- if the two objects have the same NameFormat and Name, then they refer to the same resource,
- if the two objects have the same NameFormat and different Names, then they refer to different resources,
- if the two objects have different NameFormats (whether the Names are the same or different), then it is unknown whether they refer to the same resource.

This reduces the possibility that a match will be missed by a string comparison for equality simply because of an incompatibility of formats rather than non-equality of the data.

7.2.4.4 Standard Formats for Correlatable Names

It is important that correlatable names are used and formatted consistently. Storage volume names are more complex than other element names. The same format may be used in different namespaces. For example,

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

several common SCSI VPD names use the IEEE NAA format. A client can't correlate names from different namespaces. Profile-specific standardized name formats currently defined are shown in Table 3.

Table 3 - Standardized Name Formats

Description	Format (& Namespace)	Format of Name	Class
An IP interface's v4 IP address	'IPAddressV4'	Four decimal bytes delimited with dots ('.')	ComputerSystem
An IP interface's v6 IP address	'IPAddressV6'	'x:x:x:x:x:x:x:x', where the 'x's are the uppercase hexadecimal values of the eight 16-bit pieces of the address. Examples: 'FEDC:BA98:7654:3210:FEDC:BA98:7654:3210', '1080:0:0:0:8:800:200C:417A' Leading zeros in individual fields should not be included and there shall be at least one numeral in every field. (This format is compliant with RFC 2373.) In addition, omitting groups of zeros or using dotted decimal format for an embedded IPv4 address is prohibited.	ComputerSystem
An IP interface's MAC	'MACAddress'	Six upper case hex bytes, bytes are delimited by colons ':'	NetworkPort
The DNS Name of a TCP/IP node	'DNSName'	A legal DNS name (fully qualified) consisting of strings delimited by periods.	ComputerSystem
World-Wide Name	'WWN'	16 un-separated upper case hex digits (e.g. '21000020372D3C73')	FCPort
SCSI VPD page 83 type 3, Association 0, NAA 0101b	NAA, VPD83Type3	NAA name with first nibble of 5. Recommended format (8 bytes long) when the ID is directly associated with a hardware component	StorageVolume
VPD page 83, type 3h, Association=0, NAA 0110b	NAA VPD83Type3	NAA name with first nibble of 6. Recommended format (16 bytes long) when IDs are generated dynamically	StorageVolume
VPD page 83, type 3h, Association=0, NAA 0010b	NAA VPD83Type3	NAA name with first nibble of 2. Recommended format (8 bytes long) when IDs are generated dynamically	StorageVolume
VPD page 83, type 2h, Association=0	VPD83Type2		StorageVolume
VPD page 83, type 1h, Association=0	VPD83Type1		StorageVolume

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table 3 - Standardized Name Formats (Continued)

Description	Format (& Namespace)	Format of Name	Class
VPD page 80, serial number	VPD80	Only if serial number refers to devices rather than the enclosure	StorageVolume
Concatenation of Vendor,Product, SerialNumber	SNVMI	3 strings representing the vendor name, product name within the vendor namespace and serial number within the model namespace. Strings are delimited with a '+' and spaces are included. Note that Vendor and Product are fixed length: Vendor ID is 12 bytes, Product is 16 bytes. SerialNumber is variable length and can be up to 252 bytes in length.	StorageVolume
FC Node WWN	NodeWWN	16 un-separated upper case hex digits (e.g. '21000020372D3C73')	FibreChannel StorageVolume, if target has a single LUN

7.2.4.5 Case Sensitivity

Names and NameFormats are case sensitive and the cases provided in Table 3 should be used If not otherwise specified, uppercase should be used.

7.2.4.6 Preferred Correlatable Names

For various reasons, some Agents and Providers may not have access to the preferred correlatable name of a managed object. Because of this, each Profile defines a preferred order of alternate correlatable names, if any, to maximize the possibility of a secondary match. This also helps alias matches (when a managed object is known by multiple correlatable names) to be found more efficiently if aliases matching is supported in future versions of the standard. In cases where an Agent or Provider is unable to use any of the existing correlatable names defined here or in the Profile, a NameFormat of 'Other' should be used as shown above. In cases where no correlatable name information is known to the Agent or Provider, both the Name and NameFormat fields may take on the value of 'Unknown'.

Note that secondary name matches using an alternate name format are not guaranteed, since this specification does not provide mapping between alternate name formats. Use of alternate name formats should be done with care to avoid having two CIM object instances that represent the same underling entity.

7.2.4.7 Concatenation

Sometimes, it may be necessary to concatenate multiple formatted names to create a correlatable name. In this case, both the NameFormat and the Name should contain those strings delimited with a plus sign character '+'. If the strings being concatenated contain this delimiter character, this character should be escaped with a backslash '\'.

--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.2.5 Events – CIM indications

7.2.5.1 Background

Indications are the mechanism used to accomplish the following functional capabilities in SMI-S (see 5.2, “Capabilities of this version”):

- allow a client to receive asynchronous notification that the configuration of a SAN has changed;
- allow a client to receive asynchronous notification that the health of a SAN resource has degraded;
- allow a client to receive asynchronous notification that the performance of a SAN interconnect has degraded.

CIM Indications are described in a DMTF white paper, “Common Information Model Indications”, which can be obtained from the education subsection of the DMTF web site (www.dmtf.org).

Indications are also used in place of non-blocking methods for long-running operations. In most cases, the operation requested in a method completes quickly, the return status from the method indicates the status of the operation. When a long-running operation (such as RAID volume creation) is requested, the method return code indicates whether the operation started successfully. An indication is sent when the operation is complete. Information on the indication is included in the profile whenever long-running operations are implemented.

7.2.5.2 Using indications

Clients request indications to be sent to them by subscribing to the indications. Subscriptions are stored in CIMOM as CIM object instances. A *Subscription* is expressed by the creation of a IndicationSubscription

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

association instance that references a IndicationFilter (a filter) instance and a IndicationHandler (a handler) instance. A Filter contains the query that selects an indication class or classes.

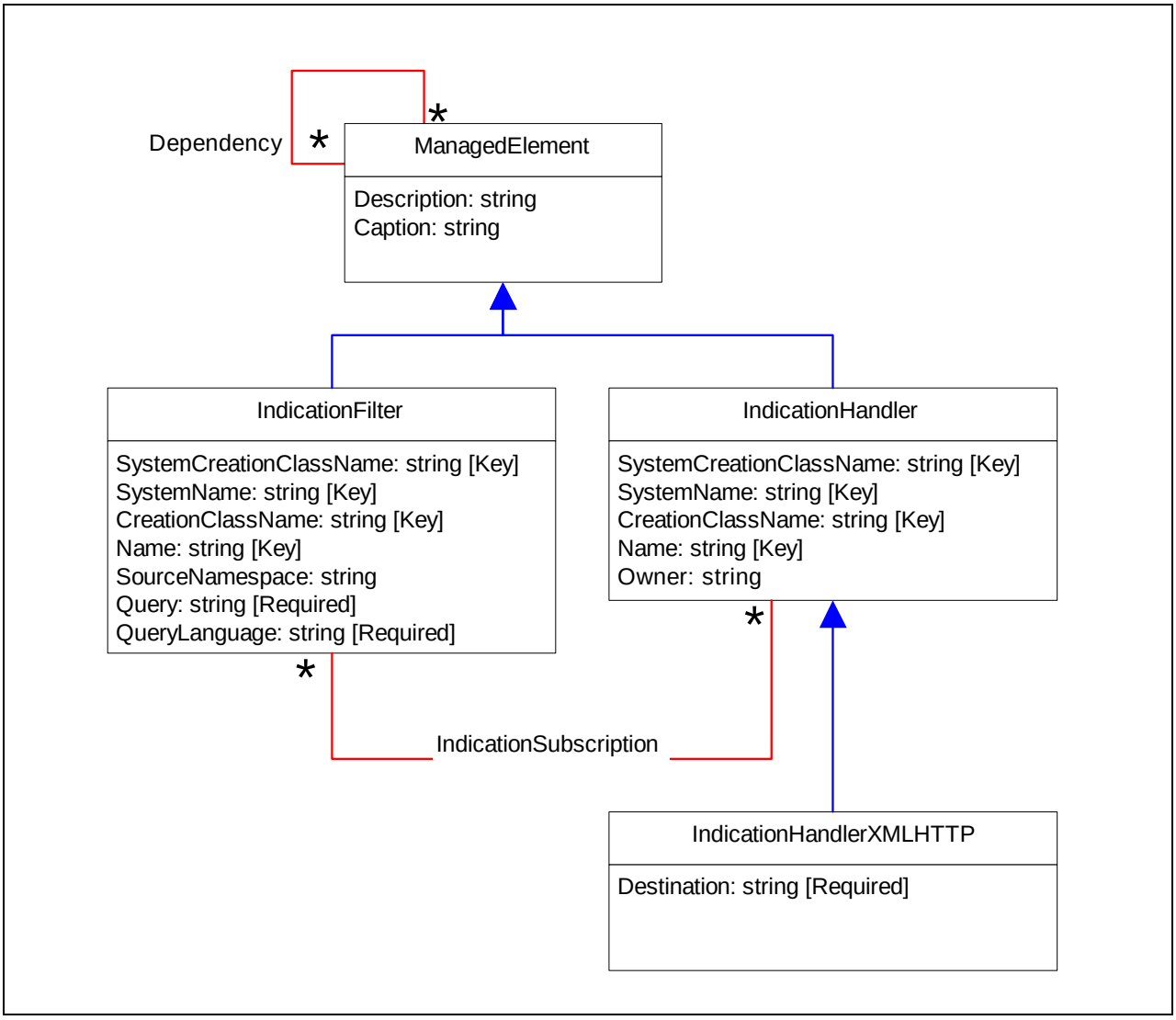


Figure 15 - Indication Filter Schema

Filters can be created by indication consumers (e.g., SMI-S Clients) or indication providers (e.g., SMI-S Agents). The client would create these using CreateInstance intrinsic method.

The query property of the IndicationFilter is a string that specifies which indications are to be delivered to the client. There is also a query language property that defines the language of the query string. Example query strings are:

```
"SELECT * FROM CIM_AlertIndication"

"SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_ComputerSystem"
```

AlertIndication and InstModification are types of indications (see 7.2.5.3). The first query says to deliver all alert type indications to the client and the second query says to deliver all instance modification indications to the client, where the instance being modified is a ComputerSystem (or any subclass thereof).

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

IndicationHandler specifies the means of delivering indications to the client. The subclass IndicationHandlerXMLHTTP provides for XML encoded indications to be sent to a specific URL, which is specified as a property of that class.

When a client receives an indication, it may receive some information with the indication and then need to do additional queries to determine all of the consequences of the event. As far as possible, all relevant pieces of information are given in the indication.

7.2.5.3 Indication hierarchy

Indications are grouped in three broad categories, ClassIndication, InstIndication and ProcessIndication (see figure 16).

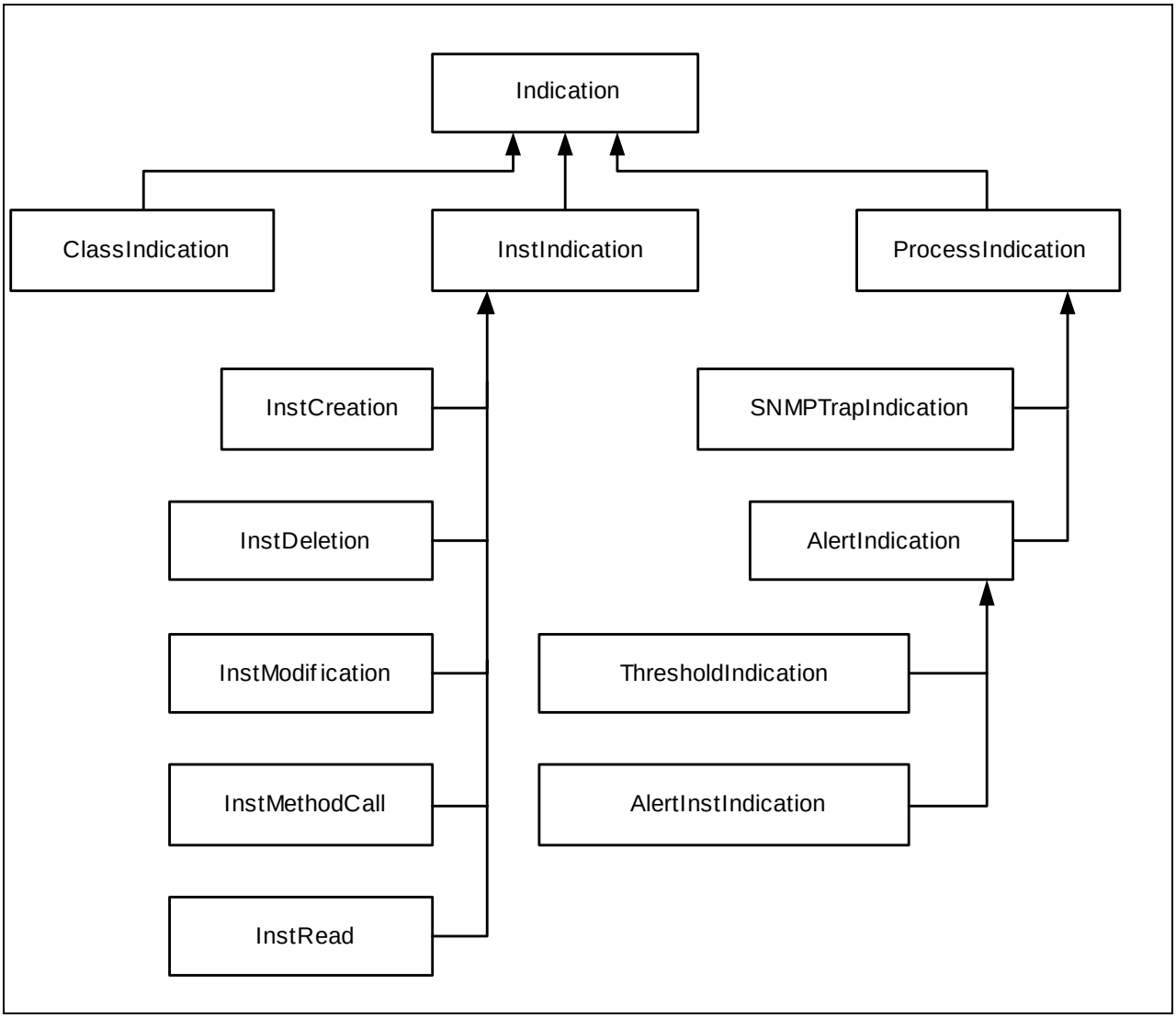


Figure 16 - Indications Schema

A ClassIndication is delivered in response to the creation, deletion or modification of a class, i.e., when there are changes to the schema. SMI-S clients should not need to subscribe to ClassIndications. An InstIndication is delivered in response to the creation, deletion, modification, etc. of an instance. For example, an InstIndication is delivered when a new volume is created or a zone is deleted. ProcessIndications allow for indications that are not associated with changes to specific instances. An event can be modelled with one of three indication subclasses.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

An InstIndication contains an embedded copy of the object that generated the indication. In the case of an InstModification, there are two copies: one with the old value and the other with the new value. These embedded copies may be full copies of the object, or they may contain only the properties that have changed.

An AlertIndication is a simpler indication that contains just strings and enumerated types. One of its properties is the path to the object generating the alert. Other properties include alert type, severity and description. An AlertIndication can be used to indicate changes in the health condition or other state of a SAN.

A SNMPTrapIndication is designed to encapsulate the information from an SNMP trap in an indication. Without a standardization process, SNMPTrapIndications are not interoperable and are generally not used in SMI-S agents.

In general, it is best to use InstIndication for all events that result in the creation, deletion or modification of instances in the SMI-S Agent.

7.2.5.4 Agent/Provider Considerations

7.2.5.4.1 Overview

As mentioned above, a SMI-S profile can be deployed as a proxy provider running in a general-purpose CIMOM or as a SMI-S agent – a combination lightweight CIMOM and provider – used when CIM is embedded on a device.

Considerations that apply to either deployment are the following.

A general purpose CIMOM (and perhaps an embedded agent) allows a client to create indications filters. The provider shall send a return code indicating a request to create an instance of a filter is unsupported. This allows the provider to inform clients which types of indications the provider supports. For example, a provider that does not support SNMPTrapAlertIndications should return unsupported for an indications filter create request.

Agents shall persist subscription information across reboots. For CIM, the subscription information is IndicationFilter and IndicationHandler classes.

An InstIndication subclass can only embed a single instance. A hardware configuration change may involve many instances of objects and associations. Ideally, Agents will detect and merge groups of related hardware events and then send a single indication for an object identifying the system using the SystemCreationClassName property. The client shall rediscover the indicating system to determine the impact of the change.

7.2.5.4.2 SMI-S Embedded Agent Considerations

A SMI-S Agent can minimize footprint by initializing “canned” IndicationFilter objects and returning “unsupported” for all requests to create filter instances. A SMI-S client can determine what indications the agent supports by enumerating these objects. A minimal embedded agent can simply support a subset of these IndicationFilter query strings:

1) “SELECT * FROM CIM_InstIndication”;

2) “SELECT * FROM CIM_AlertIndication”.

The presence of an IndicationFilter object with query string 1 indicates that the agent supports InstIndication, and similarly for the others.

The embedded agent should supply more detailed queries as described in the profile subclauses that follow.

A standard implementation of indications requires the agent to accept client requests to create indication handlers. Other aspects of SMI-S profiles do not require the agent to handle instance creation requests (the CIM operations “Basic Read” functional group). The embedded agent implementer has two options:

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

- use the Instance Manipulation functional group rather than Basic Read. The agent can treat non-indications instance creation requests as unsupported. At a minimum, the agent shall allow instance creation of IndicationHandlerXMLHTTP and IndicationSubscription instances;
- if the agent wishes to provide NO instance creation, then the agent needs to provide a backdoor for indications subscribers. For example, the agent can require customers to edit a text file describing indications subscriptions.

If the agent opts for no indications support, it shall assure that no IndicationFilter instances exist in the SMI-S Agent and to return “unsupported” to requests to create instance of IndicationHandler instances.

7.2.5.5 Client Considerations

The client needs to determine whether each target CIMOM is an embedded SMI-S agent or a general-purpose CIMOM with a SMI-S provider. The client should try to create an instance of an IndicationHandlerXMLHTTP. If the embedded agent does not allow the client to subscribe via CIM, it returns unsupported. The client can then enumerate IndicationHandlerXMLHTTP instances to determine whether they are subscribed via some non-CIM facility.

If the client can create an IndicationHandlerXMLHTTP instance, it should then try to create an IndicationFilter instance. A return of unsupported indicates the CIMOM is an embedded agent and supplies its own filters. In this case, the client enumerates the existing filters and creates IndicationSubscription associations to their IndicationHandler.

The client can minimize the number of filters by using the indications schema hierarchy. For example, subscribing to InstIndication is the same as subscribing to “InstCreation”, “InstDeletion” and “InstModification”.

Client needs to consider subscriptions that generate excessive events. Subscriptions to a general-purpose CIMOM (as determined by the tests described above) should be specific to the provider – for example “select * from CompnayCorp_InstIndication” rather than “select * from CIM_InstIndication”.

When a client receives an “InstCreation” subclass, it needs to rediscover the indicating system to determine the associations and other classes impacted by the configuration change. Providers will sometimes opt to consolidate complex configuration changes into a single indication.

The general algorithm for client subscription is:

```
Look for an existing IndicationHandlerXMLHTTPs
If one exists targeting your indication listener,
                          Then you are already subscribed from agent
                          persistence, exit
                          Else Create an IndicationHandlerXMLHTTP
                          instance
If the response is “unsupported”,
                          Then quit (this provider does not support
                          indications)
Enumerate IndicationFilters
If the desired filter instances do not exist,
                          Then try to create them.
If filter instance creation requests fail,
                          Then back off to an existing filter.
Create instances of IndicationSubscription associating the desired
                          filters and your handler.
```

The client should look for status changes represented as either “AlertIndication” or as “InstModification” with a status change. With “InstModification”, the current and previous statuses can be compared, for example:

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
"select * from CompanyCorp_InstModification
                            where PreviousInstance.Status <>
                           SourceInstance.Status"
```

where CompanyCorp would be replaced with an appropriate, vendor-specific prefix. The DMTF events white paper has other examples of filter queries.

The client can use indications to get information about the general health of the SAN. For example, the class "FCPortStatistics" includes among its properties various error counts. A query like this:

```
Select * FROM CIM_FCPortStatistics
                           WHERE PreviousInstance.ErrorFrames <
                           SourceInstance.ErrorFrames
```

generates an indication whenever the ErrorFrames count increases.

If the client is unable to create this query (i.e. if the agent doesn't support this filter), then the client can periodically read the "FCPortStatistics" of all the "FCPort"s in the model. This method, however, is much more expensive in terms of communications bandwidth and load on both the client and the server.

7.2.5.6 Requirements

SMI-S Clients shall use the subclass IndicationHandlerXMLHTTP when creating subscriptions.

If indications are supported, then the DMTF query language shall be supported (this is a DMTF requirement).

7.2.5.7 Implementation Considerations

The encoding of indications is specified in WQL, a variant of the Structure Query Language (see http://msdn2.microsoft.com/en-us/library/aa394606.aspx).

The specification for the EmbeddedObject qualifier is defined in the CIM Specification Errata (version 2.2.2),

http://www.dmtf.org/standards/documents/WBEM/CIM_Errata/CIM_Spec_Addenda222.pdf

7.2.6 Device Credentials

The device credentials are modelled using the CIM classes "SharedSecretService" and "SharedSecret". The "ComputerSystem" class represents the device, and the "SharedSecret" object contains the credentials in its properties.

A SMI-S client or application can pass the device credentials to the agent or object manager by instantiating the "SharedSecret" object, using the CIM intrinsic method NewInstance(). The SMI-S agent or provider uses the information from this object to talk to the device.

For more information, see 7.3.3.12.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.2.7 Recipe Conventions

7.2.7.1 Recipe Definition

Recipe: A set of instructions for making something from mixing various ingredients in a particular sequence. The set of ingredients used by a particular recipe is scoped by the particular profile, subprofile or some other well-defined context in which that recipe is defined.

A recipe shall specify an interoperable means for accomplishing a particular task across all conformant implementations. However, a recipe does not necessarily specify the only set of instructions for accomplishing that task. Nor are all tasks that may be accomplished necessarily specified by the set of recipes defined for a particular profile or subprofile.

In order to compress the document, some recipes are implied or assumed. This would include, for instance, that the set of available, interoperable properties are those explicitly defined by a particular profile or subprofile. In general, any CIM intrinsic read methods on profile or subprofile models are implied. However, CIM intrinsic write methods (Create/Delete/Modify) should not be assumed unless explicitly listed in the profile or subprofile definition with a well defined semantic.

For a profile or subprofile, the set of all defined and implied recipes defines the range of interoperable behavior across all conformant implementations. Unless specifically defined in a recipe, other sequences of actions (even simple Create/Delete instance requests) are not guaranteed to have the same results across multiple implementations.

Each recipe defines an interoperable series of interactions (between a SMI-S Client and a SMI-S Server) required to manage storage devices or applications. Another goal is to list the operations required for the CIM Client realize functionality. It is not a goal to comprehensively express the programming logic required to implement the recipe in any particular language. In fact, recipes are limited to the expression of CIM or SLP operations, and they may simply reference or describe any of the implementation that may be required beyond that.

7.2.7.2 Recipe Pseudo Code Conventions

7.2.7.2.1 Overview

A recipe's instructions are written using the pseudo code language defined in this subclause.

All recipes are prefixed with a summary narrative of the functionality being implemented. This summary may be included explicitly as part of the recipe or reference to the appropriate narrative that can be found elsewhere in the standard.

NOTE The use of optional features (profiles or subprofiles) in recipes shall be clearly identified.

CIM Operations and their parameters are taken directly from the *CIM Operations Over HTTP* specification. It is assumed that these methods are being called on the CIM Client API. Arrays grow in size automatically.

7.2.7.2.2 General Syntax

<condition>	logical statement that evaluates to true (Boolean)
!<condition>	tests for false (Boolean)
<action>	unspecified list of programming logic that is not important to the understanding of the reader for a particular recipe.
@{recipe}	logic flow is contained within the specification of the recipe elsewhere in the specification
<variable>	some variable

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.2.7.2.3 CIM related variable and methods

7.2.7.2.3.1 CIM Instances and Object Names

$name represents a single instance (CIMInstance) with a given variable name

$name.property represents a property in a single instance (CIMInstance)

$name.getObjectPath()
method returns a object name, REF, to the CIM Instance

$name.getNameSpace()
method returns the namespace name for the CIM Instance or Object Name

{value1, value2 ...}
an anonymous array, comprised of selected values of a given type; an anonymous array is an array that is not referable by a variable

Example:

{“Joe”, “Fred”, “Bob”, “Selma”}

$name[ ] represents an array of instances (CIMInstances) with a given variable name; array are initialized by constructing an anonymous array.

Example:

Names = {“Joe”, “Fred”, “Bob”, “Selma”}

$name-> represents an object path name (CIMObjectPath)

$name->[ ] represents an array of object names of a given name

$name->property
represents a property of object $name

$name[ ].size() returns the number of CIM instances in the array

$name->[ ].length returns the number of CIM object names in the array

#name[ ].length returns the number of variable elements in the array

%name[ ].length returns the number of method arguments elements in the array

7.2.7.2.3.2 Extrinsic method arguments

%name represents a CIM Argument that can contain any CIM or other variable

%name[ ] represents an array of CIM Arguments

7.2.7.2.3.3 Other Variables

#name neither CIM Instance nor Object Name variable. The type may be a string, number or some other special type. Types are defined in the CIM Specification 2.2.

#name[ ] a non-CIM variable array

"literal” some string literal

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.2.7.2.4 Data Structure

Variables can be collected by an array. The array can be indexed by another variable (see 7.2.7.3.3).

Arguments are always indexed by strings. In other words, the arguments are retrieved from the array by name.

7.2.7.2.5 Operations

=	assigns right value to left value
==	test for equivalency
!=	test for not equivalency
<	true if the left argument is numerically less than the right argument
>	true if the left argument is numerically greater than the right argument
<=	true if the left argument is numerically less than, or equal to, the right argument
>=	true if the left argument is numerically greater than or equal to the right argument
&&	condition A AND condition B
||	condition A OR condition B
+, -, *, /	addition, subtraction, multiplication and division, respectively
++, --	increment and decrement a variable, respectively. Placement of the operator relative to the variable determines whether the operation is completed before or after evaluation.

Example:

```
#i = 1
#names[] = {"A", "B, "C"}
“B” == #names[++#i]  is true
2 == #i is true
```

Example:

```
#i = 2
#names[] = {"A", "B, "C"}
“B” == #names[#i++]  is true
3 == #i is true
```

//	comments
nameof	returns an Object Name given a CIM Instance. This unitary operator does nothing in other usages.
ISA	tests for the name of the CIM Instance or object name

Example:

```
if ($SomeName-> ISA CIM_StorageVolume) {
<The Object Name is a reference to a CIM_StorageVolume >
}
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.2.7.2.6 Control Operations

The pseudocode used in this standard relies on control operators common to most high-level languages. For example:

- **for**

Example:

```
for #x in <variable array> {
<actions>
}
```

- **if**

Example:

```
if (<condition>) {
<actions>
} ;
if (<condition>) {
<actions>
} else {
<alternate actions>
}
```

- **do/while**

Example:

```
do {
<actions>
} while (<condition>)
```

- continue
  Within a **for** loop: initialize loop variable to next available value and restart loop body. Terminate loop if no more loop variable values available. Within a **do/while** loop: transfer control immediately to **while** test.

Example:

```
for #i in <array> {
     if (<some condition>)
         continue;    // process next loop variable
    <alternative>
}
```

- **break**: interrupts the sequence of statement execution within a loop block and exits the loop block altogether. The looping condition is not re-evaluated Statement execution starts at the next statement outside of the loop block.

- **exit**
  Terminate recipe instantly, including termination of any callers.

Example:

```
if (<unexpected condition>)
   exit
```

7.2.7.2.7 Functions

7.2.7.2.7.1 Function Declaration

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

A function definition is of the form *sub functionName(),* followed by the body of the function enclosed in braces. If parameters are to be passed to a function, then they are expressed as a comma-separated list of arguments within the parentheses following the function name. Each argument is comprised of a data type and an accompanying argument name.

Functions shall be declared at the beginning of a recipe.

```
sub functionName(integer nArg1, Class &cArg2) {
<actions>
}
```

7.2.7.2.7.2 Function invocation

A function invocation is of the form *&functionName()*. If parameters are to be passed to a function, then they are expressed as a comma-separated list within the parentheses following the function name.

```
&functionName(5, pClass)
```

7.2.7.2.8 Exception handling

All operations may produce exceptions or errors. The following construct is used to test for particular errors. Once a particular error is caught, then special exception handling logic is processed. Only CIM Errors can be caught.

```
try {
     <actions>
}
     catch (CIM Exception) {
 <recovery actions>
 }
 The error received may also be thrown
 throw CIM Exception
```

7.2.7.2.9 Built-in functions

a) boolean = compare(<variable>, <variable>)
   1) Used to determine if two variables of the same type are equivalent.
   2) The variables shall not be CIM instances or object names nor other complex data types or structures.
   3) The variables shall be of the same type.

b) $instance = newInstance("CIM Classname")
   1) Creates a CIM instance, which does not exist in the CIMOM (yet), that can be later filled in with properties and passed to CreateInstance. The namespace is assumed to be the same that the CIM client is connected to.

c) $instance - newInstance("CIM Namespace", "CIM Classname”)
   1) Variable of the above method that has the namespace name as an argument.

d) boolean = contains(<test value>, <variable array>)
   1) Used to test if the variable array contains a value equivalent to the test variable.
   2) The array shall be of variables of the same types as the test variable.
   3) If the equivalency is found with at least one value then the function returns true, else false is returned.
   4) If the array is not a simple, or non-CIM, data type then the test value shall be a CIM property, $SomeInstance.SomeProperty or $SomeObjectname->SomeProperty.

e) %Argument = newArgument("Argument Name", <variable>)

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

1) Creates a CIM Argument of a given name containing a value, CIM or non-CIM.

f) $objectPath-> = newObjectPath("Class name", "NameSpace name")
1) Returns a new ObjectPath, built from the supplied arguments.
2) Required to perform the EnumerateInstances and EnumerateInstanceNames operations.

7.2.7.2.10 Extrinsic method calls

```
<variable> = InvokeMethod ($someobjectname->, "Method Name",
 %InArguments[], %OutArguments[])
```

7.2.7.3 Common Recipes

7.2.7.3.1 Overview

This subclause defines recipes that can be used as common functions and utilities for all recipes in SMI-S.

7.2.7.3.2 Determine what the health of the storage device given the health of the components. (Operational status on managed elements)// DESCRIPTION

```
// Iterate every system device associated with the system. If any of the
// system devices return an  operational status indicating degraded
// health or failure, then the status of the entire system is degraded.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
// 1.The object name for the device, CIM_ComputerSystem, of
//   interested has been previously identified and defined in the
//   $Device-> variable

sub boolean systemDeviceOK ($Device->)
{
     #bReturn = false
     #OK = 2 // from OperationalStatus ValueMap
     $SystemDevices[] = Associators(
        $Device->,
        “CIM_SystemDevice”,
        null,
        “PartComponent”,
        false,
        false,
        {“OperationalStatus”})
     for #i in $SystemDevices[]
     {
        if (!contains(#OK, $SystemDevices[#i].OperationStatus))
        {
           #bReturn = false
        }
     }
     return #bResult
}
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3 Profiles

7.3.1 Profile content

7.3.1.1 Profile and subprofile definition

A profile is a named standard for CIM Server based management of a particular set of subsystems for a defined set of uses. The name of the profile is scoped by its authorizing organization. All profiles defined in this standard, except the Server Profile, are scoped by SNIA. The Server profile is scoped by DMTF. The CIM Server is expected to advertise supported profiles (using SLP). All parts of a profile shall be implemented to conform to the SMI-S standard. If a profile is implemented, then all constituent parts, except those defined in subprofiles, shall be implemented.

A subprofile is a named subset of a profile. The CIM Server shall advertise supported subprofiles (using SLP). The name of the subprofile is scoped by its parent profile. A subprofile may (or may not) be implemented. That is, a subprofile represents an optional part of the SMI-S standard. But if it is implemented, the subprofile prescribes its implementation and behavior.

A profile can refer to one or more subprofiles. In addition, a profile can also refer to other profiles. That is, a profile can act as a subprofile to another profile.

Profiles and subprofiles provide a context for implementation and implementation behavior for a subset of the CIM model. That is, classes, properties, methods and indications can be defined as required in the context of a profile or subprofile. A profile or subprofile can add restrictions to usage and behavior, but cannot change CIM defined characteristics. For example, if a property is required in the CIM model, then it is required in a profile (or subprofile). On the other hand, a profile or subprofile may define that a property is required (profile required) even if it is not required by the general CIM model.

7.3.1.2 Format for Profile Specifications

For each profile there is a set of information that is provided to specify the characteristics and requirements of the profile.

NOTE 1 Subprofiles are also defined using this format, but they are clearly identified as subprofiles.

Each profile is defined in terms of the subclauses that are described in Table 4.

--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

NOTE 2 Schema diagrams are logically part of a profile description. However, they can be rather involved and cannot be easily depicted in a single diagram. As a result, the reader is advised to refer to DMTF characterizations of schema diagrams

Table 4 - Profile components

Profile Element	Goal
Description	A textual introduction to the CIM Subset (e.g., SAN entity) being profiled. It provides a high-level foundation for the more detailed descriptions to follow.
Standard Dependencies	The list of standards on which this profile or subprofile depends. For subprofiles, the subprofile inherits the standards listed in the parent profile and shall not be listed here. Only unique standards added by the subprofile shall be listed.
Profile Dependencies	The list of profiles that this profile (or subprofile) requires. NOTE Profiles that provide optional features of this profile shall not be listed here. They are listed in the “Optional Subprofiles and Profiles” subclause for this profile
CIM Server Requirements	A list of requirements on the CIM Server that shall be supported in order to support the profile or subprofile. NOTE A subprofile may simply declare that the support required is the same as the parent profile. If, however, a subprofile defines requirements, the requirements shall add to those of the parent profile. That is, a subprofile cannot remove requirements of its parent profile.
Instance Diagrams	One or more instance diagrams to highlight common implementations that employ the components of this profile or subprofile. Instance diagrams also contain classes and associations but represent a particular configuration. Multiple instances of an object may be depicted in an instance diagram. Instance diagrams may include references to dependent or optional profiles or subprofiles, but it shall be obvious which profile or subprofile these belong to. If a profile (or subprofile) refers to optional material, the optional material should be enclosed in a **dashed line** box to indicate that it is optional.
Durable Names and Correlatable IDs	The Durable Names and Correlatable IDs for resources exported by the profile AND identifies the Durable Names and Correlatable IDs used by other profiles. For the Durable Names exported by the profile, the subclause identifies the valid name formats and the specific encoding of names for each name format. For Correlatable IDs exported by the profile, the source for the ID is identified and the conditions that would cause the ID to be reset are identified.
Methods	A list of the extrinsic and intrinsic methods whose semantics are standardized. NOTE Any given implementation can implement more methods, but the methods and semantics listed in this subclause are those upon which this standard is based.
Client Considerations	A summary of the implementation concerns that are likely to be encountered by products and services that rely on the SAN entity being described. The client considerations also identify how items in the funcitionality ladder (see 5.2) are accomplished. This subclause also identifies how to find any “subprofiles” required for this profile.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table 4 - Profile components (Continued)

Profile Element	Goal
Recipes	A set of "recipes" that sequence the CIM operations and other steps required to accomplish particular tasks. These recipes do not define the upper bound of what a CIM Server may support. However, they define a lower bound. That is, a CIM implementation shall support these recipes as prescribed to be SMI-S compliant. NOTE A recipe that is defined as part of a subprofile is only required if the subprofile is implemented.
Instrumentation Requirements	A summary of the implementation concerns that shall be accounted for by agent implementations (either embedded or proxy) that provide information from one or more of the SAN entities to SMI-S clients.
Required CIM Elements	A table listing the classes, associations, subprofile, packages and indications that this profile (or subprofile) shall support. Everything listed in this subclause is required by the profile or subprofile. The subclause shall not list optional elements.
Required Properties for CIM Elements	A table listing the properties and methods that this profile (or subprofile) may support. Some properties are required, are labelled REQUIRED and shall be implemented. Others are optional. All listed properties are REQUIRED unless the description of the property either explicitly states that the property is "Optional", describes that the property is used when another property is set in a given way, or otherwise, provides special instructions. All properties that CIM defines explicitly (e.g., with a Required qualifier) or implicitly (e.g., identified as a Key) as REQUIRED are also REQUIRED by SMI-S. If a property cannot be produced despite best efforts to do so, has a value that is unconstrained by either the property's description or by a ValueMap qualifier, and is not referenced in any recipe and thereby required to have meaningful value, then the value of this property shall be NULL. Note that there is a distinction in CIM between NULL and an empty string or zero value. Implementers should assure that their CIM toolkits provide the capability to work with NULL values.
Optional Subprofiles and Profiles	A list of the profiles and subprofiles that are optional for this profile. A Profile may also list other profiles or "common subprofiles" as optional features. NOTE 1 Classes, associations, methods and indications are listed by subprofile. A class may be REQUIRED under one subprofile and optional under another. NOTE 2 A class, association, methods or indication can be listed as REQUIRED even when it supports an Optional subprofile. For example, a Disk Drive subprofile may be optional, but if an implementation chooses to model Disk Drives, then the “PhysicalPackage” class is REQUIRED.

7.3.1.3 Registry of Profiles and Subprofiles

Each profile and subprofile within the Storage Management Initiative is identified by a unique name, selected and maintained by the SNIA, to assure that SMI implementers do not encounter any namespace

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

collisions. The registry of these names and a reference to their definition within this standard are summarized in Table 5.

Table 5 - Registry of profiles and subprofiles

Area	Registered Profile Name	Registered Subprofile Names
Fabric	Fabric	Zone Control Enhanced Zoning and Enhanced Zoning Control FDMI
	Switch	Blades
	Router	Software Backend Ports LUN Masking and Mapping
Hosts	FC HBA	
	Host Discovered Resources	Initiator Target
Storage	Array	Cluster Software Access Points Location Pool Manipulation Capabilities and Settings Extent Mapping Disk Drive Backend Ports LUN Creation LUN Masking and Mapping Copy Services Job Control Device Credentials
	In-band Virtualization System	Cluster Software Access Points Location Pool Manipulation Capabilities and Settings Extent Mapping LUN Creation LUN Masking and Mapping Copy Services Job Control Device Credentials
	Storage Library	Software Access Points Location Limited Access Port Elements
Server	Server	ObjectManagerAdapter

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.3.2 Common CIM Packages

7.3.2.1 Description

There are packages of classes and associations that are used in multiple profiles or subprofiles. Rather than repeat the material in each of the Profile models, the information is documented once (here) and referenced in the appropriate Profile and Subprofile models.

A “Package” is just a set subset of CIM constructs that go together to effect an aspect of the model (e.g., physical packaging). They are not subprofiles, in that they are not formally recognized as profiles or subprofiles in the CIM Server model. They are named here for reference in this specification. The names are not recognized in the CIM Server model for profiles and subprofiles.

7.3.2.2 Physical Package “package”

7.3.2.2.1 Description

CIM has a strong separation between the Physical and Logical sides of the model. A System is 'realized' using a SystemPackaging association to a PhysicalPackage (or one of its subclasses such as Chassis). The physical containment model can then be built up using Container associations and subclasses (such as PackageInChassis).

The physical elements can be described as products using Product and ProductPhysicalComponent associations. The Product instances can be built up into a hierarchy using the ProductParentChild association.

Figure 17 shows an example of the use of the physical classes.

The Physical Package “package” is used in most SNIA Profiles. Specifically, it is used in the profiles for the Switch, Routers, Extenders, HBAs, Management Appliance, JBOD, Array, Out-of-band Virtualization, In-band Virtualization and Tape Libraries. In the context of SMI-S, it shall be used to hold “Product” identification information (Vendor, serial number and version) for profiles model specific products. The Physical Package “package” is not required in Profiles that don’t correspond to an actual vendor product (e.g., CIM Server, Fabric or Host Discovered Resources).

In addition to defining Physical Package at the “System” level, Physical Package may also be defined at a lower, subcomponent level. For SMI-S, Physical Package is used in the “Disk Drive” and Logical Devices supported by Tape libraries (e.g., PhysicalTape, Tape Drive and Changer Device). If the subcomponents are supported by the Profile, they shall model their physical packaging. When subcomponents are modelled, there shall be a container relationship between their physical package and the containing package (e.g., the System level physical package). In addition, there shall be a ProductParentChild association between the subcomponent Product and the parent Product.

The Physical Package constructs may also be used to model other aspects of the environment. However, this is not required. Note that each controller is realized by a card. The cards are contained in a controller chassis. Each JBOD chassis is a separately orderable sub-product.

7.3.2.2.2 Standard dependencies

The Physical Package “package” described here is at the CIM Schema 2.7 final level. It does not require that Profiles be on a later schema. It operates within profiles that are at the CIM schema 2.7 final or later. The subprofile operates correctly with CIM Specification 2.2 (or later) and CIM Operations over HTTP 1.1 (or later).

7.3.2.2.3 Profile dependencies

The Physical Package part of the model introduces no Profile dependencies.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.2.2.4 CIM server requirements

For the SMI-S uses of Physical Package, support for Basic Read and Association Traversal functional profiles shall be supported (by the base Profile CIM server).

The Physical Package does may include support for extrinsic methods.

The Physical Package, as a SMI-S package, is not advertised.

7.3.2.2.5 Instance diagram

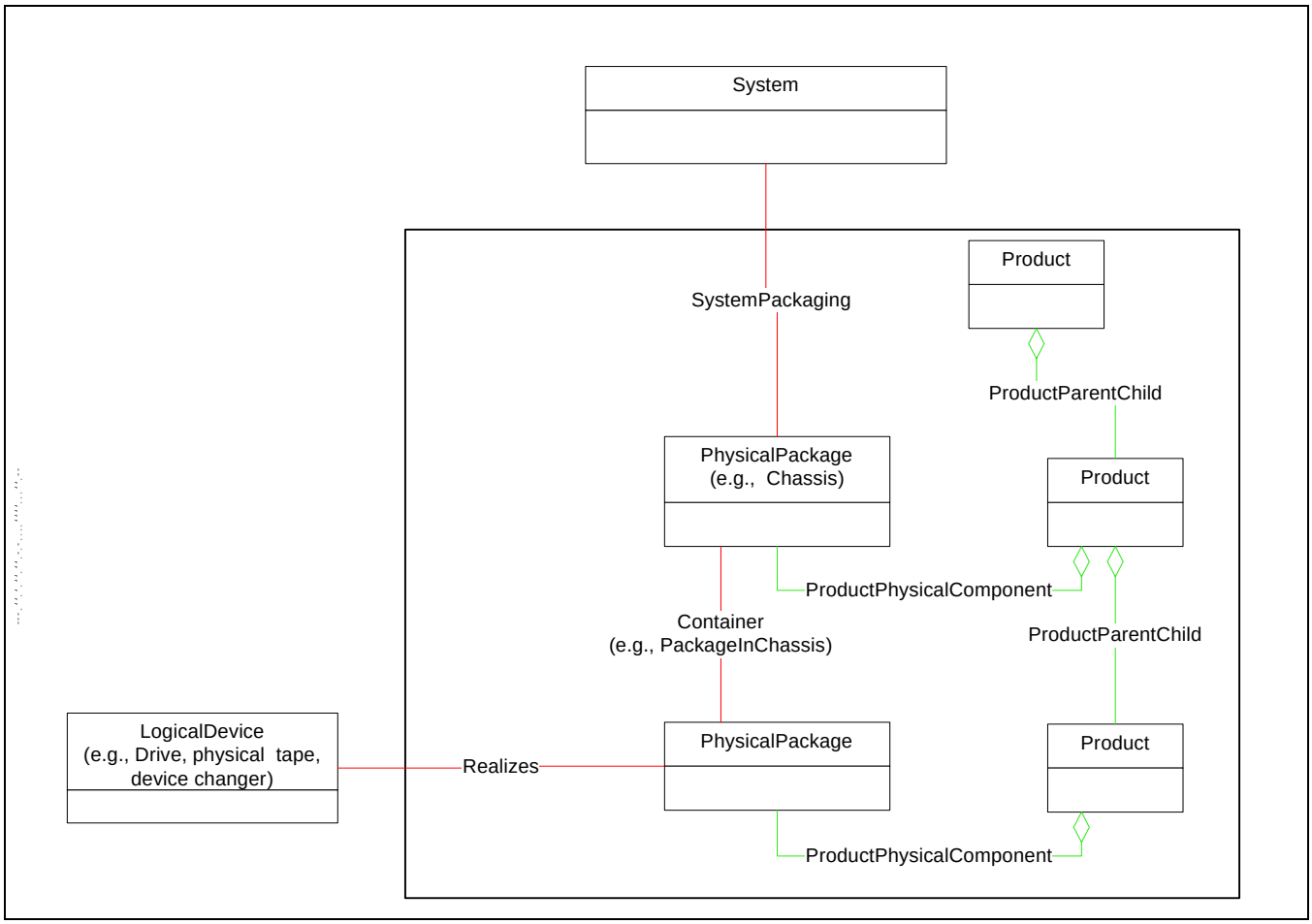


Figure 17 - Physical Package Instance

7.3.2.2.6 Durable names and correlatable IDs

The Physical Package "package" does not add any durable names or correlatable ids to the profiles in which it is used.

7.3.2.2.7 Methods

The Physical Package is populated by providers and is accessible to clients using basic read and association traversal.

No extrinsics are specified on the physical package, even though the CIM Schema identifies an extrinsic method on the PhysicalPackage class. This extrinsic may be implemented by any given implementation, but its behavior is not specified by SMI-S.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.3.2.2.8 Client considerations

7.3.2.2.8.1 Find asset information

Information about a system is modelled in PhysicalPackage. PhysicalPackage may be subclassed to Chassis; the more general PhysicalPackage is used here to accommodate device implementations that are deployed in multiple chassis. PhysicalPackage has an associated Product with physical asset information such as Vendor and Version.

7.3.2.2.8.2 Finding product information for a Profile

To locate product information (Vendor, Serial number and product versions) information about a device that conforms to the profile, you start with the “top-level” computer system and traverse the SystemPackaging to the PhysicalPackage (e.g., a Chassis). From the PhysicalPackage, the client would then traverse the ProductPhysicalComponent association to locate the Product instance. The Vendor, Serial number and version for the device is in the Product instance.

7.3.2.2.8.3 Finding Asset information within a Profile

There are certain subcomponents of a device that a client may be interested in locating. For example, disk drives in an array or changer devices in a library. To locate the asset information of these subcomponents, the client would follow the ProductParentChild association from the system Product to lower level Products.

Alternatively, if the client is starting from a LogicalDevice, it can locate the PhysicalPackage by following the Realizes association from the LogicalDevice. From the PhysicalPackage, the client can find the Product information by traversing the ProductPhysicalComponent association.

7.3.2.2.9 Recipes

No recipes have been defined for this Package.

7.3.2.2.10 Instrumentation Requirements

7.3.2.2.10.1 Well Defined Subcomponents

When establishing physical packages for subcomponents (e.g., disk drives, changers, etc.) the provider shall populate both a Container and Realizes associations. Similarly, when establishing the Product instances for the packages the provider shall populate the ProductParentChild association to the parent product.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.2.2.11 Required CIM Elements

Table 6 - Required CIM Elements

Profile Classes & Associations	Notes
SystemPackaging	This associates a PhysicalPackage to the System it supports. For Tape Libraries, this association is actually subclassed to LibaryPackage. For other profiles, this would be subclassed to ComputerSystemPackage.
PhysicalPackage (7.3.2.2.12.2)	Or a subclass of this (e.g., Chassis or Card). When subclassed, only the PhysicalPackage properties are required. This can be the “System” package or the package for a subcomponent (e.g., drive).
Product (7.3.2.2.12.3)	This class holds vendor, model and serial number information for the product in question. This can be the “system” product or a subcomponent product (e.g., drive).
ProductPhysicalComponent (7.3.2.2.12.4)	This associates a PhysicalPackage to the Product.
Container	This associates a PhysicalPackage to its component physical packages (e.g., Drives in a Storage System). This may be subclassed (e.g., PackageInChassis), but only the Container properties are required. NOTE This is only required if component parts are modelled.
ProductParentChild	This association aggregates subcomponent products under higher level products. NOTE This is only required if multiple Product instances are modelled.
Realizes	This associates a logical device (e.g., Drive) to its physical package. NOTE This is only required if component parts are modelled.
Packages	
None	
Methods	
None	
Package Indications	
None	

7.3.2.2.12 Required Properties for CIM Elements

7.3.2.2.12.1 SystemPackaging

Similar to the way that LogicalDevices are 'Realized' by PhysicalElements, Systems may be associated with specific packaging/PhysicalElements.This association explicitly defines the relationship between a System and its packaging.

This association is used in SMI-S to associate a System with its PhysicalPackages.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

SystemPackaging is subclassed from Dependency.

Table 7 - Required Properties for SystemPackaging

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Antecedent	ref	override	The PhysicalPackage(s) that realize a System.
Dependent	ref	override	The System.

7.3.2.2.12.2 PhysicalPackage

The PhysicalPackage class represents PhysicalElements that contain or host other components. Examples are a Rack enclosure or an adapter Card. In the context of SMI-S, PhysicalPackage is used to model the physical aspects of the “System” and may be used to model Logical Devices that are contained in the System.

PhysicalPackage is subclassed from PhysicalElement.

Table 8 - Required Properties for PhysicalPackage

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
ElementName	string		User Friendly name. This property is OPTIONAL.
Name	string	maxlen(1024)	This property is OPTIONAL
Tag	string	maxlen(256), key	An arbitrary string that uniquely identifies the Physical Element
CreationClassName	string	maxlen(256). key	The name of the concrete subclass
Manufacturer	string	maxlen(256)	
Model	string	maxlen(64)	
SerialNumber	string	maxlen(64)	This property is OPTIONAL
Version	string	maxlen(64)	This property is OPTIONAL
Partnumber	string	maxlen(256)	This property is OPTIONAL

7.3.2.2.12.3 Product

Product is a concrete class that aggregates PhysicalElements, software (SoftwareIdentity and SoftwareFeatures), Services and/or other Products and is acquired as a unit. Acquisition implies an agreement between supplier and consumer that may have implications to Product licensing, support and warranty. Non-commercial (e.g., in-house developed Products) should also be identified as an instance of Product.

In the context of SMI-S, Product is used to convey vendor and serial number for the objects being modelled. These can be at the “system” level (e.g., array) or at the logical device (component) level.

--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Product is subclassed from ManagedElement.

Table 9 - Required Properties for Product

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
ElementName	string		User Friendly name. Suggested use is Vendor, Version and product name.
Name	string	key, maxlen(256)	Commonly used Product name.
IdentifyingNumber	string	key, maxlen(64)	Product identification such as a serial number on software, a die number on a hardware chip, or (for non-commercial Products), a project number.
Vendor	string	key, maxlen(256)	The name of the Product's supplier or entity selling the Product (the manufacturer, reseller, OEM, etc.). Corresponds to the Vendor property in the Product object in the DMTF Solution Exchange Standard.
Version	string	key, maxlen(64)	Product version information.

7.3.2.2.12.4 ProductPhysicalComponent

Indicates that the referenced PhysicalElement is acquired as part of a Product.

This association is used in SMI-S to associate a Product with its PhysicalPackage.

ProductPhysicalComponent is subclassed from Component

Table 10 - Required Properties for ProductPhysicalComponent

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
GroupComponent	ref	override	The Product.
PartComponent	ref	override	The PhysicalElement that is a part of the Product.

7.3.2.2.12.5 Container

The Container association represents the relationship between a contained and a containing PhysicalElement. A containing object shall be a PhysicalPackage.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Container is subclassed from Component.

Table 11 - Required Properties for Container

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
GroupComponent	ref	max(1), override	The PhysicalPackage that contains other PhysicalElements, including other Packages.
PartComponent	ref	override	The PhysicalElement that is contained in the Package.

7.3.2.2.12.6 ProductParentChild

The ProductParentChild association defines a parent child hierarchy among Products. For example, a Product may come bundled with other Products.

ProductParentChild is not subclassed from anything.

Table 12 - Required Properties for ProductParentChild

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Parent	ref	key	The parent Product in the association.
Child	ref	key	The child Product in the association.

7.3.2.2.12.7 Realizes

Realizes is the association that defines the mapping between LogicalDevices and the PhysicalElements that implement them.

In SMI-S, this class shall be used if physical packaging is modelled for system components (Logical Devices).

Realizes is subclassed from Dependency.

Table 13 - Required Properties for Realizes

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Antecedent	ref	override	The physical component that implements the Device.
Dependent	ref	override	The LogicalDevice.

7.3.2.2.13 Optional Subprofiles

A package shall not have subprofiles.

7.3.2.3 Software Package

7.3.2.3.1 Description

The Software Package shall be implemented as part of the Switch Profile and the FDMI Subprofile.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

The Software Package shall be implemented when it is part of the optional Software Subprofile for the Extenders, Routers, Management Appliance, Array, Out-of-band Virtualization System and In-band Virtualization System Profiles. The package is described here. The Software Subprofile is defined later under Common Subprofiles.

Information on the installed software is given using the SoftwareIdentity class. This is linked to the system using a SoftwareInstalledOnSystem association.

Software information can be associated with the “top” level ComputerSystem (if all components are using the same software) or a component ComputerSystem if the software loaded varies according to the processor used.

7.3.2.3.2 Standard dependencies

The Software package is defined using the CIM Schema 2.8 final. As such it can be used in profiles at 2.8 and later. It does not require that Profiles be on a later schema. It operates within profiles that are at the CIM schema 2.8 final or later. The package operates correctly with CIM Specification 2.2 (or later) and CIM Operations over HTTP 1.1 (or later).

7.3.2.3.3 Profile dependencies

The Software package introduces no Profile dependencies.

7.3.2.3.4 CIM server requirements

For the SMI-S uses of the Software package, support for Basic Read and Association Traversal functional profiles shall be supported (by the base Profile CIM server).

The Software package does not require support for extrinsic methods.

As a package, the Software subprofile is not advertised.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.2.3.5 Instance Diagram

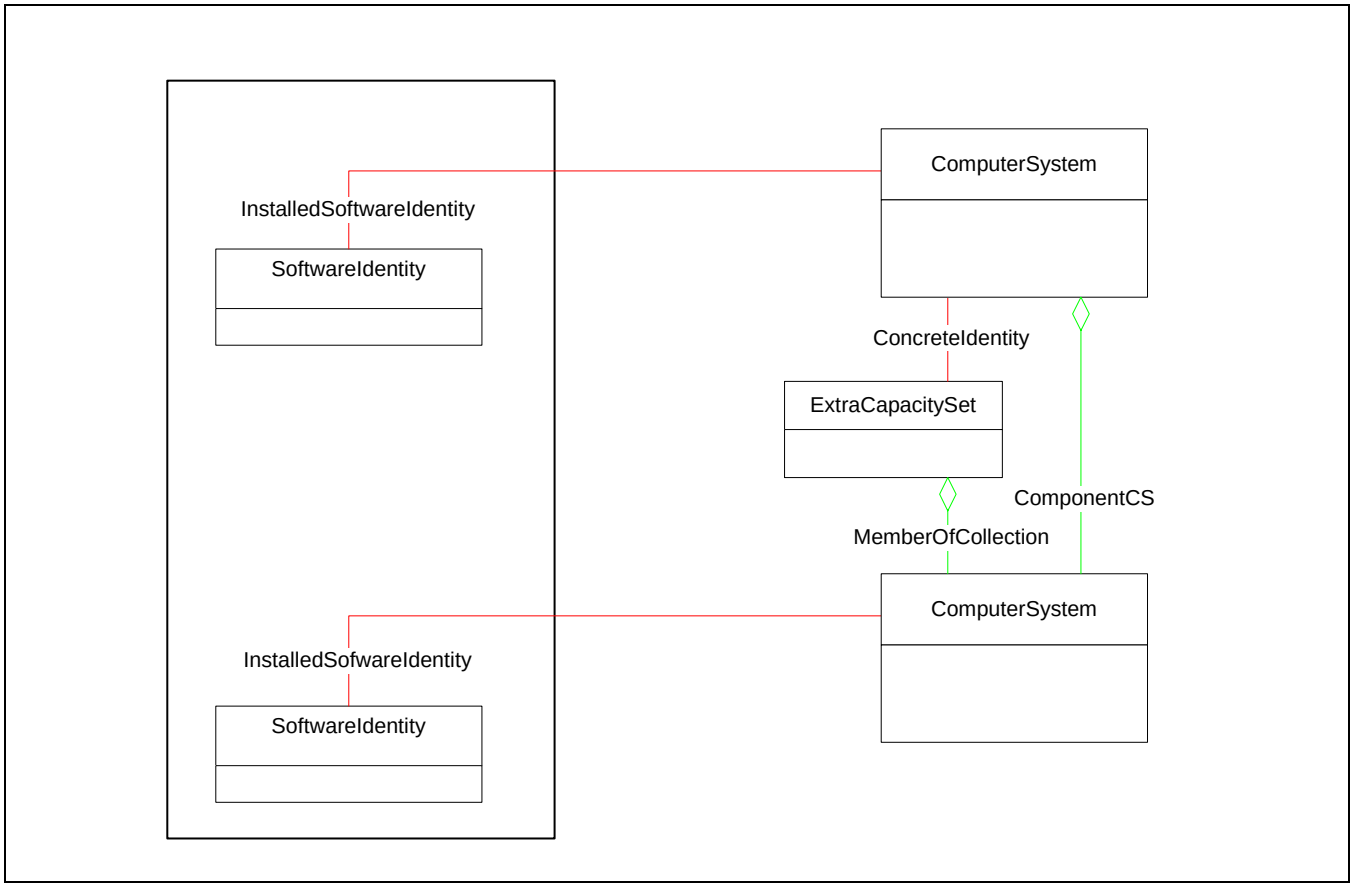


Figure 18 - Software Instance Diagram

7.3.2.3.6 Durable names and correlatable IDs

The Software Package does not add any durable names or correlatable IDs to the profiles (or subprofiles) in which it is used.

7.3.2.3.7 Methods

The Software package is populated by providers and is accessible to clients using basic read and association traversal.

No extrinsics are specified on the Software Package.

7.3.2.3.8 Client Considerations

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

7.3.2.3.9 Recipes

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

7.3.2.3.10 Instrumentation requirements

Firmware

Firmware is modelled as SoftwareIdentity. SoftwareInstalledOnSystem is used for firmware associated with a System.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.3.2.3.11 Required CIM Elements

Table 14 - Required CIM Elements

Profile Classes & Associations	Notes
InstalledSoftwareIdentity (7.3.2.3.12.1)	
SoftwareIdentity (7.3.2.3.12.2)	
Packages	
None	
Associated Indications	
None	

7.3.2.3.12 Required Properties for CIM Elements

7.3.2.3.12.1 InstalledSoftwareIdentity

The InstalledSoftwareIdentity association allows the identification of the ComputerSystem on which a particular SoftwareIdentity is installed.

InstalledSoftwareIdentity is not subclassed from anything.

Table 15 - Required Properties for InstalledSoftwareIdentity

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
System	ref	key	The system the software is installed on.
InstalledSoftware	ref	key	The SoftwareIdentity that is installed.

7.3.2.3.12.2 SoftwareIdentity

The SoftwareIdentity is used to model either software or firmware.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

SoftwareIdentity is subclassed from LogicalElement.

Table 16 - Required Properties for SoftwareIdentity

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
InstanceID	string	key	The name used to identify this SoftwareIdentity.
VersionString	string		Software Version should be in the form <Major>.<Minor>.<Revision> or <Major>.<Minor><letter><revision>.
Manufacturer	string		Manufacturer of this software.
BuildNumber	uint16		OPTIONAL. The internal identifier for this compilation of software, if available.
RevisionNumber	uint16		OPTIONAL. This is the numeric representation of the revision number in the VersionString.
MajorVersion	uint16		OPTIONAL. This is the numeric representation of the Major number in the VersionString.
MinorVersion	uint16		OPTIONAL. This is the numeric representation of the Minor number in the VersionString.

7.3.2.3.13 Optional subprofiles

A package shall not have subprofiles.

7.3.3 Common subprofiles

7.3.3.1 Overview

There are several subprofiles that are used in multiple profiles and deserve specific descriptive information. The detailed descriptions of these subprofiles are described in this subclause to avoid redundant descriptions in the profile clauses.

7.3.3.2 Access Points subprofile

7.3.3.2.1 Description

The Access Points subprofile is used in the Array, Out-of-Band Virtualization and In-band Virtualization Profiles to provide addresses of remote access points for management tools.

Most devices now have a web GUI to allow device specific configuration. This is modelled using a RemoteServiceAccessPoint. This is linked to the managed element using a HostedAccessPoint association. If several access points are provided (say one for each chassis), then multiple instances of RemoteServiceAccessPoint / HostedAccessPoint would be instantiated and linked to the specific Element by SAPAvailableForElement.

7.3.3.2.2 Standard dependencies

The Access Point subprofile is defined using the CIM Schema 2.7 final. As such it can be used in profiles at 2.7 and later. It does not require that Profiles be on a later schema. It operates within profiles that are at the

--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

CIM schema 2.7 final or later. The subprofile operates correctly with CIM Specification 2.2 (or later) and CIM Operations over HTTP 1.1 (or later).

7.3.3.2.3 Profile dependencies

The Access Point subprofile introduces no Profile dependencies.

7.3.3.2.4 CIM server requirements

For the SMI-S uses of the Access Point subprofile, support for Basic Read and Association Traversal functional profiles shall be supported by the base Profile CIM server.

The Access Point subprofile does not require support for extrinsic methods.

The Access Point subprofile is not advertised.

7.3.3.2.5 Instance diagrams

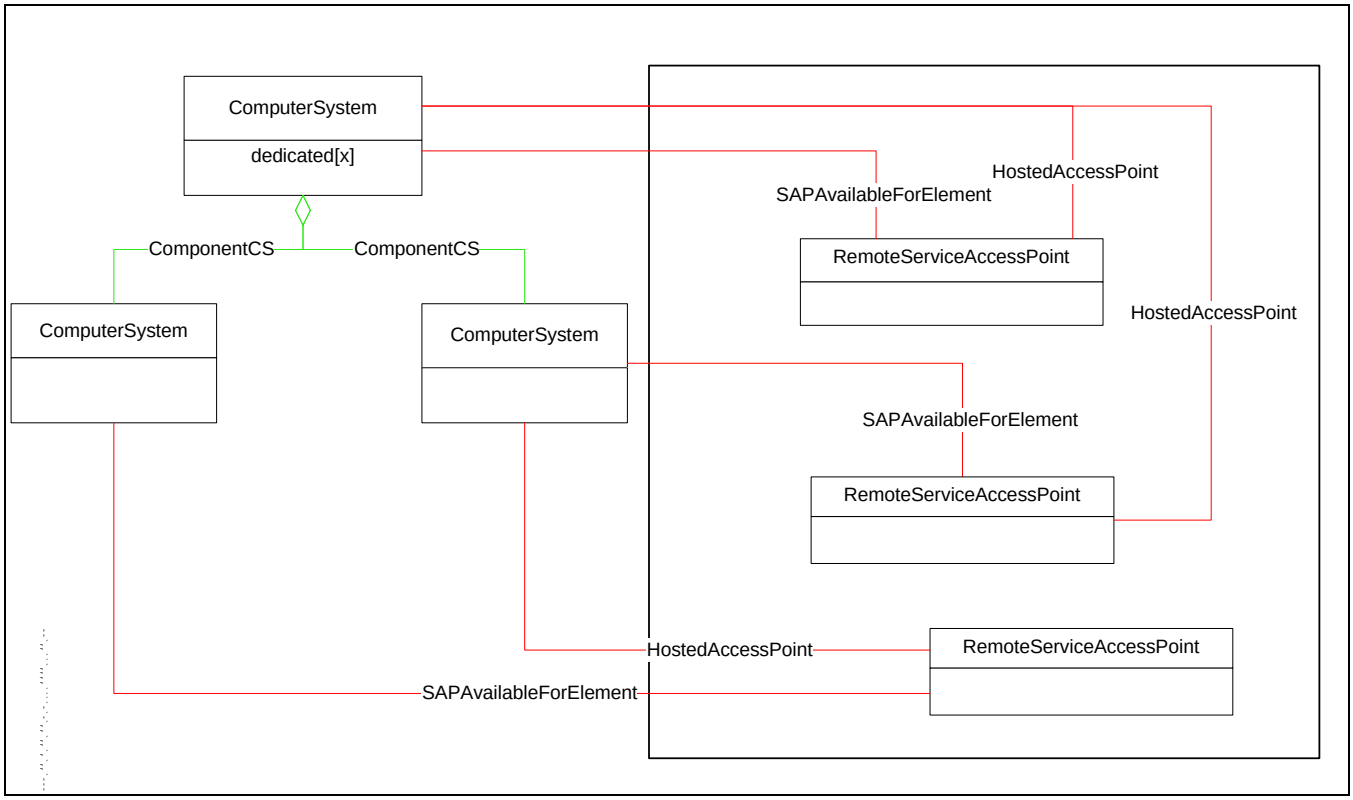


Figure 19 - Access Point Instance Diagram

Figure 19 depicts a configuration with two RemoteServiceAccessPoint instances. One represents a system-wide service and the other represents a service that applies just to certain devices.

The exposed management services may represent a web UI that can be launched by a web browser, a telnet interface, or some vendor-specific interface. The RemoteServiceAccessPoint.InfoFormat property describes the format of the AccessInfo property. Valid options include “URL” and “FQDN”. In a URL, the text before the “://” is referred to as the “scheme”. A URL with an http or https scheme is often a web/HTML page, but HTTP

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

can be used for other purposes. xxx specifies the requirements for InfoFormat, AccessInfo and the scheme subset for a URL AccessInfo.

Table 17 - RemoteAccessPoint.InfoFormat and AccessInfo Properties

InfoFormat	AccessInfo Scheme	Description
"URL"	"http" or "https"	The referenced URL shall be a valid web page. It should provide element management for the system or elements referenced by the associated HostedAccessPoint association.
"Other" with OtherInfoFormatDescription = "Non-UI URL"	"http" or "https"	Used for HTTP URLs that do not reference a valid web UI.
"URL"	anything other than "http" or "https"	May be used. No standard behavior is specified.
others from the MOF	N/A	May be used. No standard behavior is specified.

There are two associations that need to be instantiated. The "HostedAccessPoint" associates the service to the System on which it is hosted. The "ServiceAvailableToElement" associates the service to the system (or element) that is affected by the service.

7.3.3.2.6 Durable names and correlatable IDs

The Access Point subprofile does not add any durable names or correlatable IDs to the profiles (or subprofiles) in which it is used.

7.3.3.2.7 Methods

The Access Point subprofile is populated by providers and is accessible to clients using basic read and association traversal.

No extrinsics are specified on the Access Point subprofile.

7.3.3.2.8 Client considerations

See details in related profile clauses.

7.3.3.2.9 Recipes

See details in related profile clauses.

7.3.3.2.10 Instrumentation requirements

See details in related profile clauses.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.3.2.11 Required CIM Elements

Table 18 - Required CIM Elements

Profile Classes & Associations	Notes
HostedAccessPoint (7.3.3.2.12.1)	Associate the RemoteServiceAccessPoint to the System on which it is hosted.
RemoteServiceAccessPoint (7.3.3.2.12.3)	A ServiceAccessPoint for management tools.
SAPAvailableForElement (7.3.3.2.12.2)	This association identifies the element that is serviced by the RemoteServiceAccessPoint.
Packages	
None	
Methods	
None	
Subprofile Indications	
None	

7.3.3.2.12 Required properties for CIM elements

7.3.3.2.12.1 HostedAccessPoint

HostedAccessPoint is an association between a ServiceAccessPoint and the System on which it is provided. The cardinality of this association is 1-to-many and is weak with respect to the System. Each System may host many ServiceAccessPoints.

HostedAccessPoint is subclassed from Dependency.

Table 19 - Required Properties for HostedAccessPoint

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Antecedent	ref	override	The hosting System.
Dependent	ref	override	The SAP(s) that are hosted on this System.

7.3.3.2.12.2 SAPAvailableForElement

SAPAvailableForElement conveys the semantics of a Service being available for the 'use' of a ManagedElement. When use of this service is restricted or has limited availability/applicability, the SAPAvailableForElement association is instantiated between the Service and specific Processors, Chassis or other ManagedElements that can use the Service.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

SAPAvailableForElement is not subclassed from anything.

Table 20 - Required Properties for SAPAvailableForElement

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
AvailableSAP	ref	override	The Service that is available.
ManagedElement	ref	override	The ManagedElement that may use the Service.

7.3.3.2.12.3 RemoteServiceAccessPoint

RemoteServiceAccessPoint describes access and/or addressing information for a remote connection, that is known to a 'local' network element. This information is scoped/contained by the 'local' network element, since this is the context in which it is 'remote'.

Why the remote access point is relevant and information on its use are described by subclassing RemoteServiceAccessPoint or by associating to it.

RemoteServiceAccessPoint is subclassed from ServiceAccessPoint.

Table 21 - Required Properties for RemoteServiceAccessPoint

Property/Method	Type	Qualifier/ Parameter	Description/Notes
ElementName	string		User Friendly name
SystemName	string	key, maxlen (256)	
SystemCreationClassName	string	key, maxlen (256)	
CreationClassName	string	key, maxlen (256)	
Name	string	key, maxlen (256)	
AccessInfo	string		Management Address. For interoperability, this should be a URL.
InfoFormat	uint16		The format of the Management Address. For interoperability, this shall be “URL”.

7.3.3.2.13 Optional subprofiles

Table 22 - Optional Profiles or Subprofiles

Name	Notes
None	

7.3.3.3 Cluster Subprofile

7.3.3.3.1 Description

The Cluster Subprofile is an optional subprofile for the JBOD, Array, Out-of-Band Virtualization and In-band Virtualization Profiles.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

It is not defined for use in the Fabric, Switch, Routers, Extenders, HBA, Host Discovered Resources, Management Appliance, Tape Library or Server profiles.

Many profiles define a ComputerSystem as the base representation of the system being modelled. However, the device being modelled may, in fact, be made up of multiple processing elements that act as a cluster. This may be modelled using the Cluster Subprofile. Each of the elements of the cluster would be its own ComputerSystem, but they would be collected into a ComputerSystem that represents the system image of the collection of processors.

If the storage system consists of more than one controller then there is an instance of a ComputerSystem representing the overall system, and each controller is linked to that instance of ComputerSystem using ComponentCS associations.

The ‘top’ computer system would be the required part of the profile and would be the anchor point for key associations to other parts of the profile. It is not part of this subprofile, but is the point where the cluster of computer systems tie into the profile.

When a Cluster subprofile is implemented, care should be taken in where “SystemDevice” associations are attached. If the system device (e.g., FCPort) goes away if one of the component computer systems goes away, then the device shall be connected (via SystemDevice) to the component ComputerSystem. On the other hand, if the system device (e.g., StorageVolume) remains available no matter which component ComputerSystem fails or goes away, then the SystemDevice connection should be to the “top” level ComputerSystem.

7.3.3.3.2 Standard dependencies

The Cluster subprofile is defined using the CIM Schema 2.7 final. As such it can be used in profiles at 2.7 and later. It does not require that Profiles be on a later schema. It will operate within profiles that are at the CIM schema 2.7 final or later. The subprofile will operate correctly with CIM Specification 2.2 (or later) and CIM Operations over HTTP 1.1 (or later).

7.3.3.3.3 Profile dependencies

The Cluster subprofile introduces no Profile dependencies.

7.3.3.3.4 CIM server requirements

For the SMI-S uses of the Cluster subprofile, support for Basic Read, Indications and Association Traversal functional profiles shall be supported by the CIM Server.

The Cluster subprofile does not require support for extrinsic methods.

The Cluster subprofile is not advertised.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.3.3.5 Instance diagrams

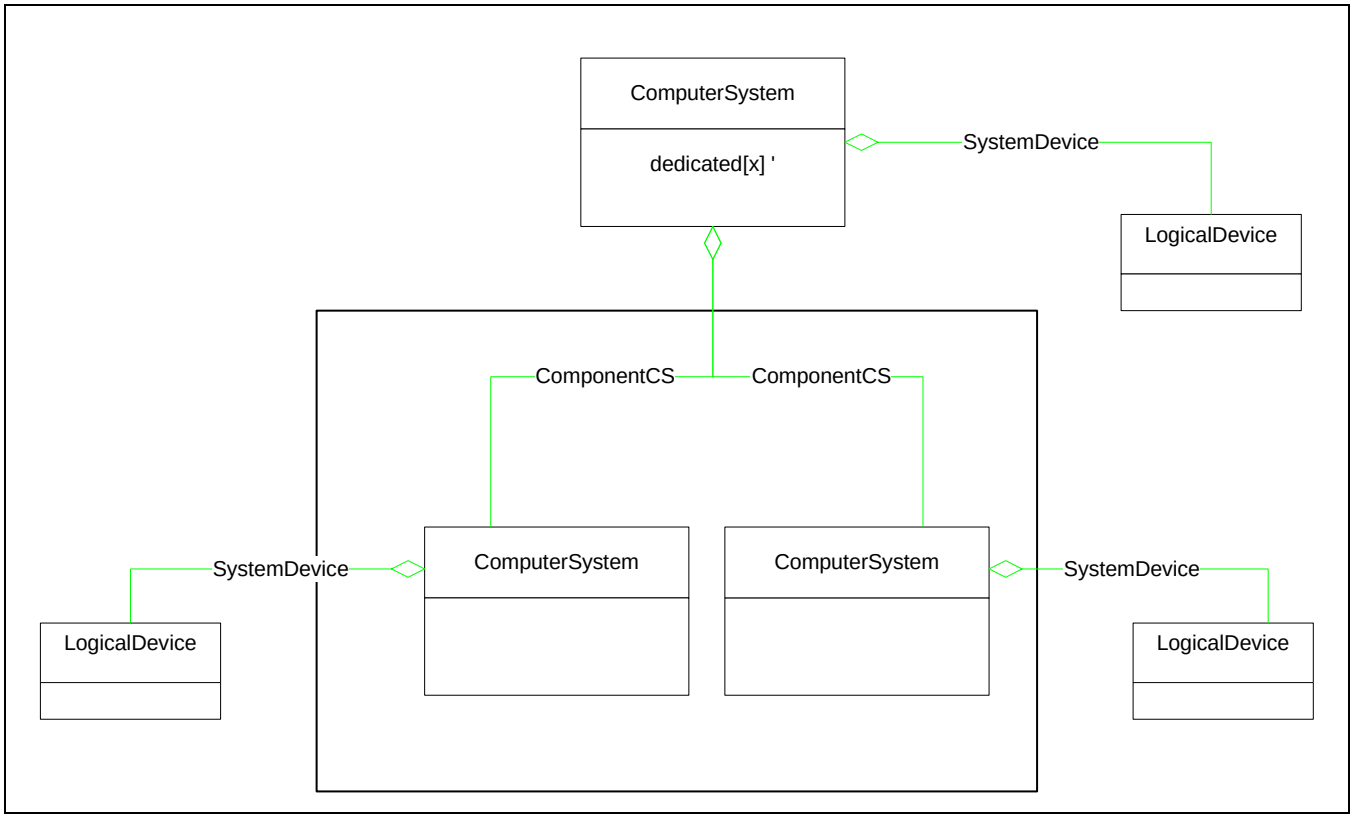


Figure 20 - Cluster Instance

7.3.3.3.6 Durable names and correlatable IDs

The Cluster subprofile does not add any durable names or correlatable IDs to the profiles (or subprofiles) in which it is used. While a Durable Name is defined for the top-level ComputerSystem, for the component computer systems they are not considered Durable. The name property of a component computer system is scoped to the top-level computer system.

7.3.3.3.7 Methods

The Cluster subprofile is populated by providers and is accessible to clients using basic read and association traversal.
No extrinsics are specified on the Cluster subprofile.

7.3.3.3.8 Client Considerations

7.3.3.3.8.1 Finding the Top-level Computer System

```
// DESCRIPTION
// A client can find the top computer systems relatively easy using an
//Associators call
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
// None

Associators(ObjectName=CIMObjectPath(CIM_ComputerSystem)
     AssocClass=CIM_ComponentCS
     ResultClass=CIM_ComputerSystem
```

--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
ResultRole=GroupComponent)
```

7.3.3.3.8.2 Find System Status of a Component Computer System

The ‘OperationStatus’ property is available on most objects in the model and is used to indicate its status. For component computer systems, the ComputerSystem instance shall have one of the following Main Operational Statuses and possibly one of the Subsidiary statuses.

Table 23 - OperationStatus for Component ComputerSystem

Main Operational Status	Possible Subsidiary Operational Status	Description
OK		The computer system has a good status.
OK	Stressed	The computer system is stressed, for example the temperature is over limit or there is too much IO in progress.
OK	Predictive Failure	The computer system will probably fail sometime soon.
Degraded		The computer system is operational but not at 100% redundancy. A component has suffered a failure or something is running slow.
Error		An error has occurred causing the computer system to stop. This error may be recoverable with operator intervention.
Error	Non-recoverable error	A severe error has occurred. Operator intervention is unlikely to fix it.
Error	Supporting entity in error	A modelled element has failed.
No contact		The provider knows about the computer system but has not talked to it since last reboot.
Lost communication		The provider used to be able to communicate with the computer system, but has now lost contact.
Starting		The computer system is starting up.
Stopping		The computer system is shutting down.
Stopped		The computer system is OK, but shut down; the management channel is still working.

A client may subscribe for Asynchronous notification of changes in status through InstModification Indications. More details on indications are in “Events - CIM Indications”.

7.3.3.3.9 Recipes

None defined in this specification.

7.3.3.3.10 Instrumentation requirements

None defined in this specification.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.3.3.11 Required CIM Elements

Table 24 - Required CIM Elements

Profile Classes & Associations	Notes
ComponentCS (7.3.3.3.12.1)	Associates the processors of the computer system to the system.
ComputerSystem (7.3.3.3.12.2)	For the processors that are clustered to make up the system.
Packages	
None	
Methods	
None	
SubProfile Indications	
To indicate the creation of a Component ComputerSystem	SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_ComputerSystem.
To indicate the deletion of a Component ComputerSystem	SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_ComputerSystem.
Change in operational status of a Component ComputerSystem.	SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_ComputerSystem AND SourceInstance.OperationalStatus <> PreviousInstance.OperationalStatus

7.3.3.3.12 Required properties for CIM Elements

7.3.3.3.12.1 ComponentCS

A ComputerSystem can aggregate another ComputerSystem. This association is used to model multiple processors that act as a single system image for a storage system. ComponentCS represents that unique and distinct ComputerSystems are aggregated by a higher level CS object. However, each of the component CSs are still distinguishable entities and are only viewed as such.

ComponentCS is subclassed from SystemComponent.

Table 25 - Required Properties for ComponentCS

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
GroupComponent	ref	override	The ComputerSystem that contains and/or aggregates other Systems.
PartComponent	ref	override	The contained (Sub)ComputerSystem.

7.3.3.3.12.2 ComputerSystem

In the context of the Cluster Subprofile, instances of this class represent processors that make up the cluster that is the storage system.

NOTE The ‘top level’ ComputerSystem is required as part of the base profiles.

--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

ComputerSystem is subclassed from System.

Table 26 - Required Properties for ComputerSystem

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
ElementName	string		User Friendly name.
OperationalStatus[ ]	uint16		Status of the component computer system (same encodings as the top-level computer system).
CreationClassName	string	key, maxlen(256)	Name of Class.
Name	string	key, maxlen (256), override	For component computer systems, the provider shall provide a unique name using one of the NameFormats.
NameFormat	string	override	For component computer systems, this should be coded as Other.

7.3.3.3.13 Optional Subprofiles

Table 27 - Optional Profiles or Subprofiles

Name	Notes
None	

7.3.3.4 Disk Drive subprofile

7.3.3.4.1 Description

The Disk Drive subprofile is used in the JBOD and Array Profiles.

A disk drive is modeled as a MediaAccessDevice (DiskDrive) containing (MediaPresent) some logical media (StorageExtent) that is realized (RealizesExtent) by some physical media. Other classes can further refine the modeling (e.g. Product or SoftwareIdentity).

The Disk Drive subprofile ties into the rest of the Array (or JBOD) profile via a number of key associations.

- ConcreteComponent - To associate an extent exported by the Disk Drive to a StoragePool
- BasedOn - To associate an extent exported by the Disk Drive to another (higher level) extent (or a Volume)
- Container - To associate the physical package of the disk drive to the physical package of the system
- ProductParentChild - to associate the product of the disk drive to a higher level product (e.g., the system product)

7.3.3.4.2 Standard Dependencies

The Disk Drive subprofile is defined using the CIM Schema 2.8 Preliminary. As such it can be used in profiles at 2.8 and later. It does not require that Profiles be on a later schema. It will operate within profiles that are at the CIM schema 2.8 final or later. The subprofile will operate correctly with CIM Specification 2.2 (or later) and CIM Operations over HTTP 1.1 (or later).

7.3.3.4.3 Profile dependencies

The Disk Drive subprofile introduces no Profile dependencies.

--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.3.4.4 CIM Server requirements

7.3.3.4.4.1 Functional profiles

Table 28 - Required Functional Profiles

Profile Required	Functional Group	Dependency
YES	Basic Read	None
NO	Basic Write	Basic Read
NO	Instance Manipulation	Basic Write
NO	Schema Manipulation	Instance Manipulation
YES	Association Traversal	Basic Read
NO	Query Execution	Basic Read
NO	Qualifier Declaration	Schema Manipulation
YES	Indication	None

7.3.3.4.4.2 Extrinsic methods

The Disk Drive subprofile does not require support for extrinsic methods.

7.3.3.4.4.3 Discovery

The Disk Drive subprofile is not advertised.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.3.4.5 Instance diagrams

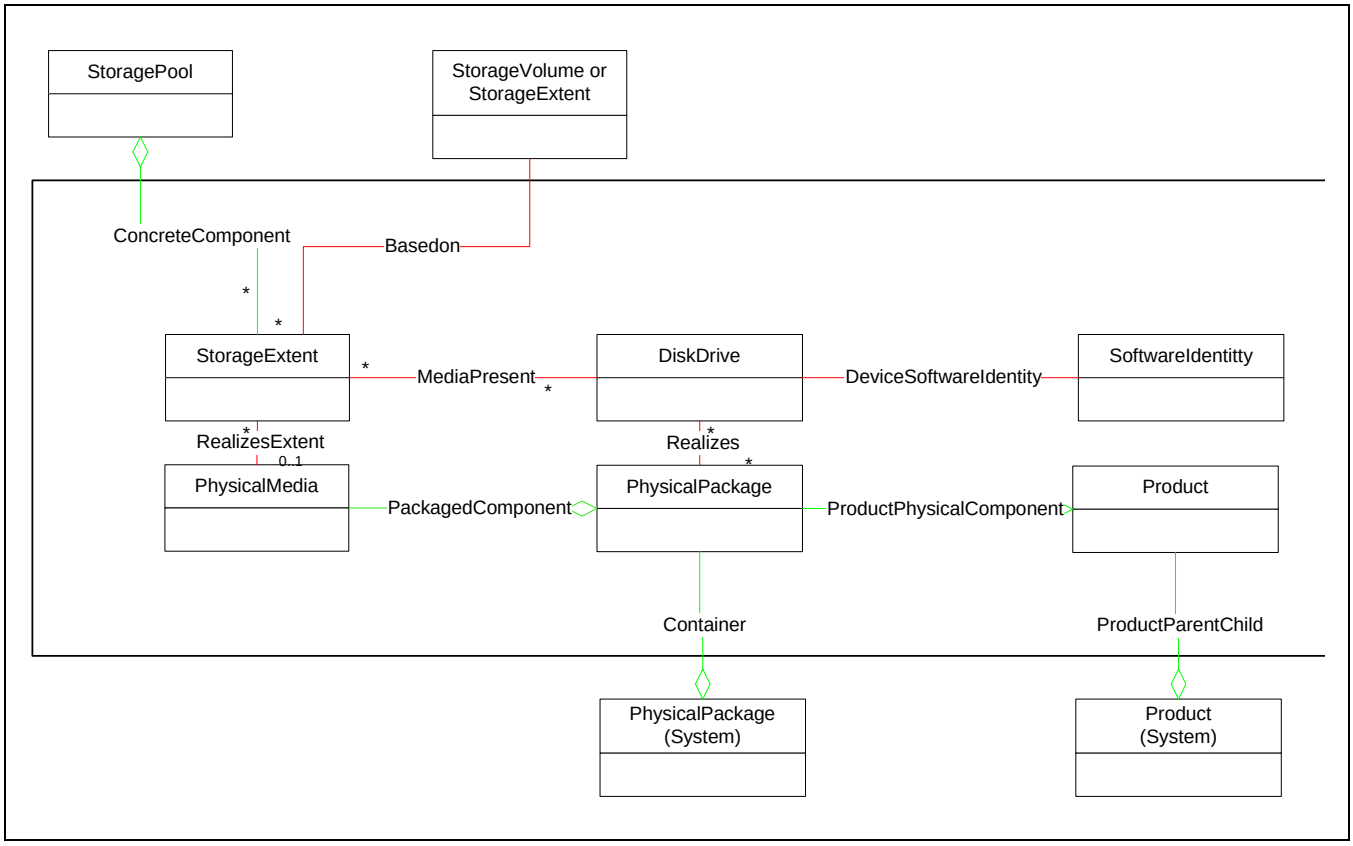


Figure 21 - Disk Drive Instance Model

7.3.3.4.6 Durable names and correlatable IDs

The Disk Drive subprofile does not add any durable names or correlatable IDs to the profiles (or subprofiles) in which it is used.

7.3.3.4.7 Methods

The Disk Drive subprofile is populated by providers and is accessible to clients using basic read and association traversal.

No extrinsics are specified on the Disk Drive subprofile.

7.3.3.4.8 Client considerations

7.3.3.4.8.1 Find Disk Drive Status

The status of a Disk Drive may be determined by the value of the OperationalStatus property. Table 29, “DiskDrive Status,” shows the allowed values for this property and their meanings. Table 29 defines the

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

possible states that shall be supported for DiskDrive.OperationalStatus. The main OperationalStatus shall be the first element in the array.

Table 29 - DiskDrive Status

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

7.3.3.4.9 Recipes

```
// DESCRIPTION
// A client can find the ‘top’ computer systems relatively easy using an
//‘Associators’ call
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
// 1.a CIM_ObjectPath to a CIM_ComputerSystem with dedicated[] =
//   [ “Array” , “Block Server”]

// Step 1. Get the Primordial Pool for the Array from the
// HostedStoragePool association.  (This represents the unallocated
// storage on the array.) To tell which pool is the PrimordialPool, look
// for a Pool with no AllocatedFromStoragePool associations where it is
//the dependent.
$StoragePools->[] = Associators(
     $StorageArray->,
     “CIM_HostedStoragePool”,
     “CIM_StoragePool”,
     null,
     null,
     false,
     false,
     null)
for $StoragePool-> in $StoragePools->[]
{
     if ($StoragePool->Primordial == TRUE)
     {
         $PrimordialPool-> = $StoragePool->
         break;
     }
}

// Step 2. Get an Enumeration of the StorageExtents that make up the
// StoragePool using the ConcreteComponent association.
$StorageExtents->[] = Associators(
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
     $PrimordialPool->,
     “CIM_ConcreteComponent”,
     “CIM_StorageExtent”,
     “GroupComponent”,
     “PartComponent”,
     false,
     false,
     null)

// Step 3. For each StorageExtent in the enumeration, follow the
// RealizesExtent association to the PhysicalMedia object.
for $StorageExtent-> in $StorageExtents->[]
{
     $PhysicalMedia->[] = Associators(
        $StorageExtent->,
        “CIM_RealizesExtent”,
        “CIM_PhysicalMedia”,
        “StorageExtent”,
        “PhysicalComponent”,
        false,
        false,
        null)
     if ($PhysicalMedia->[].length != 0)
     {
        // According to the schema, there should be zero or one.
        $PhysicalMedium-> = $StoragePools->[0]

// Step 4. Read the ElementName property and the Capacity property on
                          the
// PhysicalMedia.
        < Disk Drive $PhysicalMedium->Tag is unused and >
        < has a capacity of $PhysicalMedium->Capacity bytes.  >

     } // if.
} // for.
```

7.3.3.4.10 Instrumentation requirements

7.3.3.4.10.1 Required external associations

When implementing the Disk Drive subprofile, the ConcreteComponent association to StoragePools shall be implemented (because LogicalStorage is required in the Profile).

The Container association to a higher level physical package shall be implemented (because the PhysicalPackage for the System is required). However, in the case of the Container association, it is possible that the Disk Drive PhysicalPackage is not directly contained in the System PhysicalPackage. It shall be possible for a client to traverse the container associations from the System PhysicalPackage to the Disk Drive PhysicalPackage, even if the client is required to go through intermediate steps (that is, intermediate physical packages).

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

The ProductParentChild association from the disk drive product to the higher level product shall be implemented. It is not necessary for the System Product to be the next level up the ProductParentChild association, unless this is the implementer's preference.

7.3.3.4.10.2 Optional external associations

The BasedOn association that ties a Disk Drive extent to a higher level extent (or volume) is only required if the ExtentMapping subprofile is also implemented.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.3.4.11 Required CIM Elements

Table 30 - Required CIM Elements

Profile Classes & Associations	Notes
BasedOn	This maps the storage extent of the disk drive to other (higher level) extents that are based on the disk drive.
ConcreteComponent	This maps an extent exported by the disk drive to a StoragePool.
Container	This association aggregates the disk drive physical package to the next higher level physical package.
ProductParentChild	This association aggregates the disk drive product to the next higher level product.
ElementSoftwareIdentity (7.3.3.4.12.5)	This associates the disk drive to the firmware for the drive.
DiskDrive (7.3.3.4.12.6)	
MediaPresent (7.3.3.4.12.7)	
PhysicalMedia (7.3.3.4.12.8)	
Realizes (7.3.3.4.12.9)	
RealizesExtent (7.3.3.4.12.10)	
SoftwareIdentity (7.3.3.4.12.11)	The firmware for the disk drive.
StorageExtent	The storage extent that is exported by the disk drive.
Packages	
Physical Package “package” (7.3.2.2)	
Methods	
None	
SubProfile Indications	
Creation of a disk drive	SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_DiskDrive
Deletion of a disk drive	SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_DiskDrive

7.3.3.4.12 Required properties for CIM Elements

7.3.3.4.12.1 BasedOn

BasedOn is an association describing how StorageExtents can be assembled from lower level Extents. In the context of the Disk Drive subprofile, the BasedOn association is used to associate the extents exported by the Disk Drive to higher level storage extents (or StorageVolumes). As such, the based on association is only required if the Profile is also implementing the Extent Mapping subprofile.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

BasedOn is subclassed from Dependency.

Table 31 - Required Properties for BasedOn

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Antecedent	ref	override, key	The lower level StorageExtent.
Dependent	ref	override, key	The higher level StorageExtent.
StartingAddress	uint64		StartingAddress indicates where in lower level storage, the higher level Extent begins.
EndingAddress	uint64		EndingAddress indicates where in lower level storage, the higher level Extent ends. This property is useful when mapping non-contiguous Extents into a higher level grouping.
OrderIndex	uint16		If there is an order to the BasedOn associations that describe how a higher level StorageExtent is assembled, the OrderIndex property indicates this.

7.3.3.4.12.2 ConcreteComponent

ConcreteComponent is a generic association used to establish 'part of' relationships between ManagedElements. It is defined as a concrete subclass of the abstract Component class, to be used in place of many specific subclasses of Component that add no semantics - that is, that do not clarify the type of composition, update cardinalities or add/remove qualifiers. Note that when defining additional semantics for Component that this class shall not be subclassed.

In the context of the Disk Drive subprofile, this association ties extents exported by the disk drive to StoragePools.

ConcreteComponent is subclassed from Component.

Table 32 - Required Properties for ConcreteComponent

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
GroupComponent	ref	key, override	The parent element in the association (e.g., the StoragePool).
PartComponent	ref	key, override	The child element in the association (e.g., the StorageExtent exported by the disk drive).

7.3.3.4.12.3 Container

The Container association represents the relationship between a contained and a containing PhysicalElement. A containing object shall be a PhysicalPackage.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Container is subclassed from Component.

Table 33 - Required Properties for Container

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
GroupComponent	ref	max(1), override, key	The PhysicalPackage that contains other PhysicalElements, including other Packages.
PartComponent	ref	override, key	The PhysicalElement that is contained in the Package.

7.3.3.4.12.4 ProductParentChild

The ProductParentChild association defines a parent child hierarchy among Products. For example, a Product may come bundled with other Products.

ProductParentChild is not subclassed from anything.

Table 34 - Required Properties for ProductParentChild

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Parent	ref	key	The parent Product in the association.
Child	ref	key	The child Product in the association.

7.3.3.4.12.5 ElementSoftwareIdentity

The ElementSoftwareIdentity relationship identifies any software that is associated with a Device - such as drivers, configuration or application software, or firmware.

ElementSoftwareIdentity is subclassed from Dependency.

Table 35 - Required Properties for ElementSoftwareIdentity

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Antecedent	ref	key, override	A LogicalDevice's Software Asset.
Dependent	ref	key, override	The LogicalDevice (Disk Drive) that requires or uses the software.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

7.3.3.4.12.6 DiskDrive

Capabilities and management of a DiskDrive, a subtype of MediaAccessDevice.

Table 36 - Required Properties for DiskDrive

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Name	string	maxlen (256)	The Name property defines the label by which the object is known. When subclassed, the Name property can be overridden to be a Key property.
OperationalStatus[ ]	uint16		Indicates the current status(es) of the element. Various health and operational statuses are defined.
SystemCreationClassName	string	key, maxlen(256)	The scoping System's CreationClassName.
SystemName	string	key, maxlen(256)	The scoping System's Name.
CreationClassName	string	key, maxlen(256)	CreationClassName indicates the name of the class or the subclass used in the creation of an instance. When used with the other key properties of this class, this property allows all instances of this class and its subclasses to be uniquely identified.
DeviceID	string	key, maxlen(64)	An address or other identifying information to uniquely name the LogicalDevice.

7.3.3.4.12.7 MediaPresent

Where a StorageExtent is accessed through a MediaAccessDevice, this relationship is described by the MediaPresent association.

MediaPresent is subclassed from Dependency.

Table 37 - Required Properties for MediaPresent

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Antecedent	ref	override, key	Reference to MediaAccessDevice.
Dependent	ref	override, key	Reference to the StorageExtent accessed using the MediaAccessDevice.

7.3.3.4.12.8 PhysicalMedia

The PhysicalMedia class represents 'sealed' Media, so that the Media can then be associated with the PhysicalPackage using the PackagedComponent relationship.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

PhysicalMedia is subclassed from PhysicalComponent.

Table 38 - Required Properties for PhysicalMedia

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Tag	string	maxlen(256), key	An arbitrary string that uniquely identifies the Physical Element.
CreationClassName	string	maxlen(256). key	The name of the concrete subclass.
Manufacturer	string	maxlen(256)	
Model	string	maxlen(64)	
SKU	string	maxlen(64)	This property is OPTIONAL.
SerialNumber	string	maxlen(256)	This property is OPTIONAL.
Version	string	maxlen(64)	This property is OPTIONAL.
PartNumber	string	maxlen(256)	This property is OPTIONAL.
VendorEquipmentType	string		
Capacity	uint64		The number of bytes that can be read from or written to a Media. Data compression should not be assumed, as it would increase the value in this property. Units ("Bytes").

7.3.3.4.12.9 Realizes

Realizes is the association that defines the mapping between LogicalDevices and the PhysicalElements that implement them.

Realizes is subclassed from Dependency.

Table 39 - Required Properties for Realizes

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Antecedent	ref	override, key	The physical component that implements the Device.
Dependent	ref	override, key	The LogicalDevice.

7.3.3.4.12.10 RealizesExtent

StorageExtents can be realized by PhysicalComponents. Disks are realized by PhysicalMedia. This relationship of Extents to PhysicalComponents is made explicit by the RealizesExtent association. In addition, the StartingAddress of the StorageExtent on the Component is specified here.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

RealizesExtent is subclassed from Realizes.

Table 40 - Required Properties for RealizesExtent

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Antecedent	ref	override, key	The PhysicalComponent on which the Extent is realized.
Dependent	ref	override, key	The StorageExtent that is located on the Component.
StartingAddress	uint64		The starting address on the PhysicalComponent where the StorageExtent begins. Ending address of the StorageExtent is determined using the NumberOfBlocks and Block Size properties of the StorageExtent object. This property is OPTIONAL.

7.3.3.4.12.11 SoftwareIdentity

SoftwareIdentity class is used to model the firmware on a Disk Drive. A SoftwareIdentity object captures the management details of a part or component.

SoftwareIdentity is subclassed from LogicalElement.

Table 41 - Required Properties for SoftwareIdentity

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
InstanceID	string	key	The name used to identify this SoftwareIdentity.
VersionString	string		Software Version should be in the form <Major>.<Minor>.<Revision> or <Major>.<Minor><letter><revision>.
Manufacturer	string		Manufacturer of this Software.
BuildNumber	uint16		OPTIONAL. The internal identifier for this compilation of software, if available.
RevisionNumber	uint16		OPTIONAL. This is the numeric representation of the revision number in the VersionString.
MajorVersion	uint16		OPTIONAL. This is the numeric representation of the Major number in the VersionString.
MinorVersion	uint16		OPTIONAL. This is the numeric representation of the Minor number in the VersionString.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.3.3.4.12.12 StorageExtent

Table 42 - Required properties for StorageExtent

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
SystemCreationClassName	string	maxlen(256), key	The scoping System's CreationClassName.
SystemName	string	maxlen(256), key	The scoping System's Name.
CreationClassName	string	maxlen(256), key	The name of the concrete subclass.
DeviceID	string	maxlen(64), key	unique identifying information.
BlockSize	uint64		
NumberOfBlocks	uint64		
ExtentStatus[ ]	uint16		
OperationalStatus[ ]	uint16		

7.3.3.4.12.13 Optional subprofiles

Table 43 - Optional profiles or subprofiles

Name	Notes
None	

7.3.3.5 Extent Mapping subprofile

7.3.3.5.1 Description

The Extent Mapping subprofile is used in the Array, In-band Virtualization and Out-of-band Virtualization Profiles.

In the StoragePool mechanism storage allocation is focused on a logical usage of storage in a device. There is little information exposed to clients about how data is laid out on underlying extents. Storage administrators may want to know specifically which disk (or underlying extent) a LUN is on so they can avoid locating frequently used LUNs on the same extents. The extent mapping subprofile allows an agent to describe this information to the level of a single disk drive or underlying extent.

7.3.3.5.2 Standard dependencies

The Extent Mapping subprofile is defined using the CIM Schema 2.8 Preliminary. As such it can be used in profiles at 2.8 and later. It does not require that Profiles be on a later schema. It will operate within profiles that are at the CIM schema 2.8 final or later. The subprofile will operate correctly with CIM Specification 2.2 (or later) and CIM Operations over HTTP 1.1 (or later).

7.3.3.5.3 Profile dependencies

The Extent Mapping subprofile introduces no Profile dependencies.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.3.5.4 CIM Server requirements

7.3.3.5.4.1 Functional profiles

Table 44 - Required Functional Profiles

Profile Required	Functional Group	Dependency
YES	Basic Read	None
NO	Basic Write	Basic Read
NO	Instance Manipulation	Basic Write
NO	Schema Manipulation	Instance Manipulation
YES	Association Traversal	Basic Read
NO	Query Execution	Basic Read
NO	Qualifier Declaration	Schema Manipulation
NO	Indication	None

7.3.3.5.4.2 Extrinsic methods

The Extent Mapping subprofile does not require support for extrinsic methods.

7.3.3.5.4.3 Discovery

The Extent Mapping subprofile is not advertised.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.3.5.5 Instance diagrams

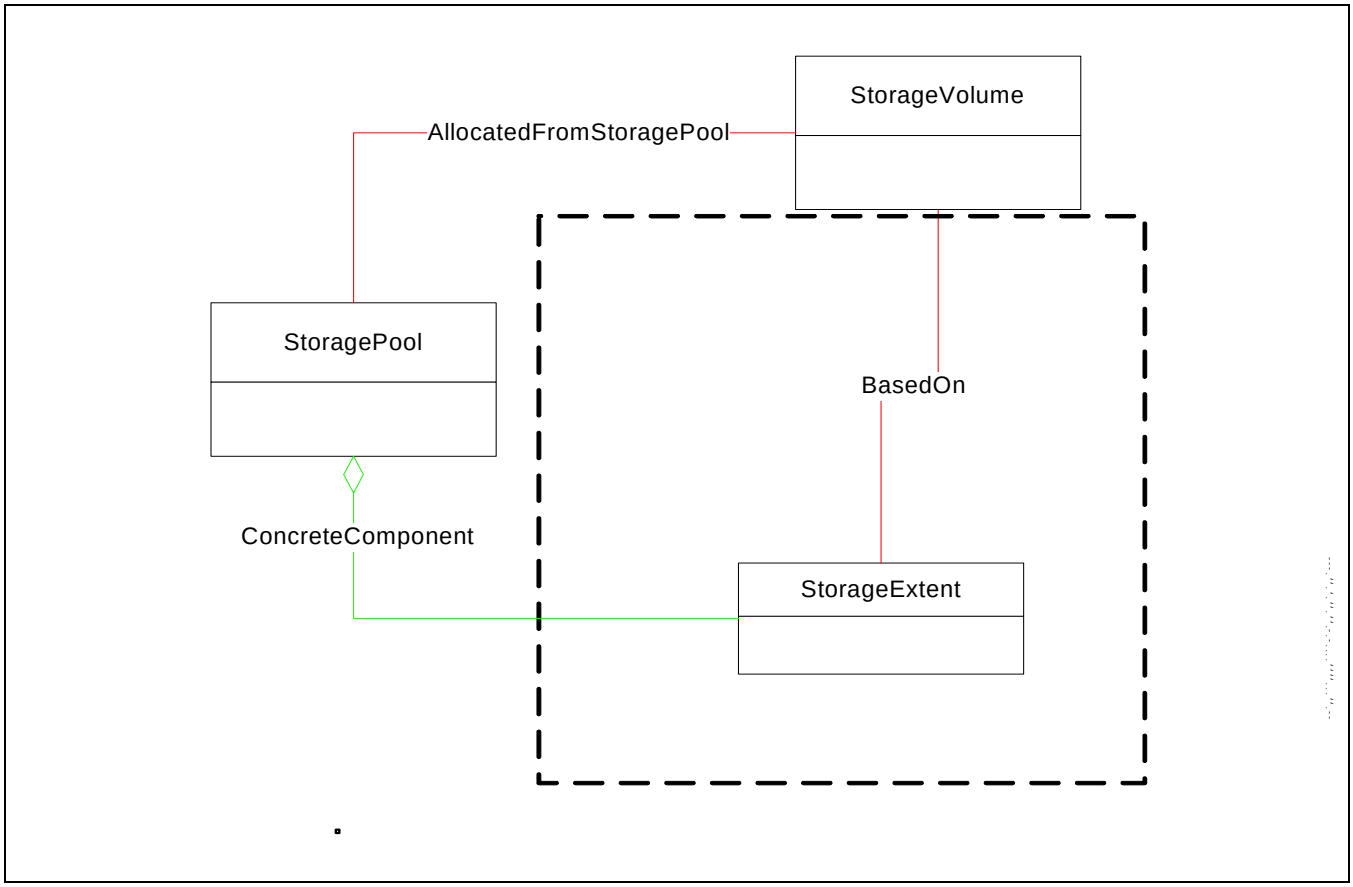


Figure 22 - Extent Mapping Instance

The instance diagram shows the extents included in the appropriate Storage Pools using the ConcreteComponent relationship. StorageVolumes created from the pool are linked back to the source extent using the BasedOn relationship.

The StartingAddress and EndingAddress may be used to locate of the data on the StorageExtent. The OrderIndex may be used to order the different BasedOn relationships composing the StorageVolume.

7.3.3.5.6 Durable names and correlatable IDs

No new durable identifiers are defined in this subprofile.

7.3.3.5.7 Methods

No method are defined in this subprofile.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.3.3.5.8 Required CIM Elements

Table 45 - Required CIM Elements

Profile Classes & Associations	Notes
BasedOn (7.3.3.5.9.1)	
ConcreteComponent (7.3.3.5.9.2)	
StorageExtent (7.3.3.5.9.3)	
Packages	
None	
Associated Indications	
None	

7.3.3.5.9 Required Properties for CIM Elements

7.3.3.5.9.1 BasedOn

Table 46 - Required Properties for BasedOn

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Antecedent	ref	override, key	StorageExtent Reference
Dependent	ref	override, key	StorageExtent Reference
StartingAddress	unit64		Where in lower level storage, the higher level Extent begins (optional).
EndingAddress	unit64		Where in lower level storage, the higher level Extent ends (optional).
OrderIndex	unit16		Indicates the order to the BasedOn associations that describes how a higher level StorageExtent is assembled (optional).

7.3.3.5.9.2 ConcreteComponent

Table 47 - Required Properties for ConcreteComponent

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
PartComponent	ref	override, key	The component StorageExtent.
GroupComponent	ref	aggregate, override, key	The Storage Pool.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.3.5.9.3 StorageExtent

Table 48 - Required Properties for StorageExtent

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Name	string		For virtualization systems, this should be the Name of the StorageVolume that is imported.
SystemCreationClassName	string	maxlen(256), key	The scoping System's CreationClassName.
SystemName	string	maxlen(256), key	The scoping System's Name.
CreationClassName	string	maxlen(256), key	The name of the concrete subclass.
DeviceID	string	maxlen(64), key	Unique identifying information.

7.3.3.5.10 Optional subprofiles

Table 49 - Optional Profiles or Subprofiles

Name	Notes
None	

7.3.3.6 Location subprofile

7.3.3.6.1 Description

Associated with product information, a PhysicalPackage may also have a location. This is indicated using an instance of a Location class and the PhysicalElementLocation association.

The Location Subprofile is an optional subprofile of the Management Appliance, JBOD, Array, Out-of-band Virtualization System and In-band Virtualization System.

7.3.3.6.2 Standard dependencies

The Location subprofile is defined using the CIM Schema 2.7 final. As such it can be used in profiles at 2.7 and later. It does not require that Profiles be on a later schema. It will operate within profiles that are at the CIM schema 2.7 final or later. The subprofile will operate correctly with CIM Specification 2.2 (or later) and CIM Operations over HTTP 1.1 (or later).

7.3.3.6.3 Profile dependencies

The Location subprofile introduces no Profile dependencies.

7.3.3.6.4 CIM Server Requirements

For the SMI-S uses of the Location subprofile, support for Basic Read and Association Traversal functional profiles shall be supported by the CIM server of the parent profile.

The Location subprofile does not require support for extrinsic methods.

The Location subprofile is not advertised.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.3.6.5 Instance diagram

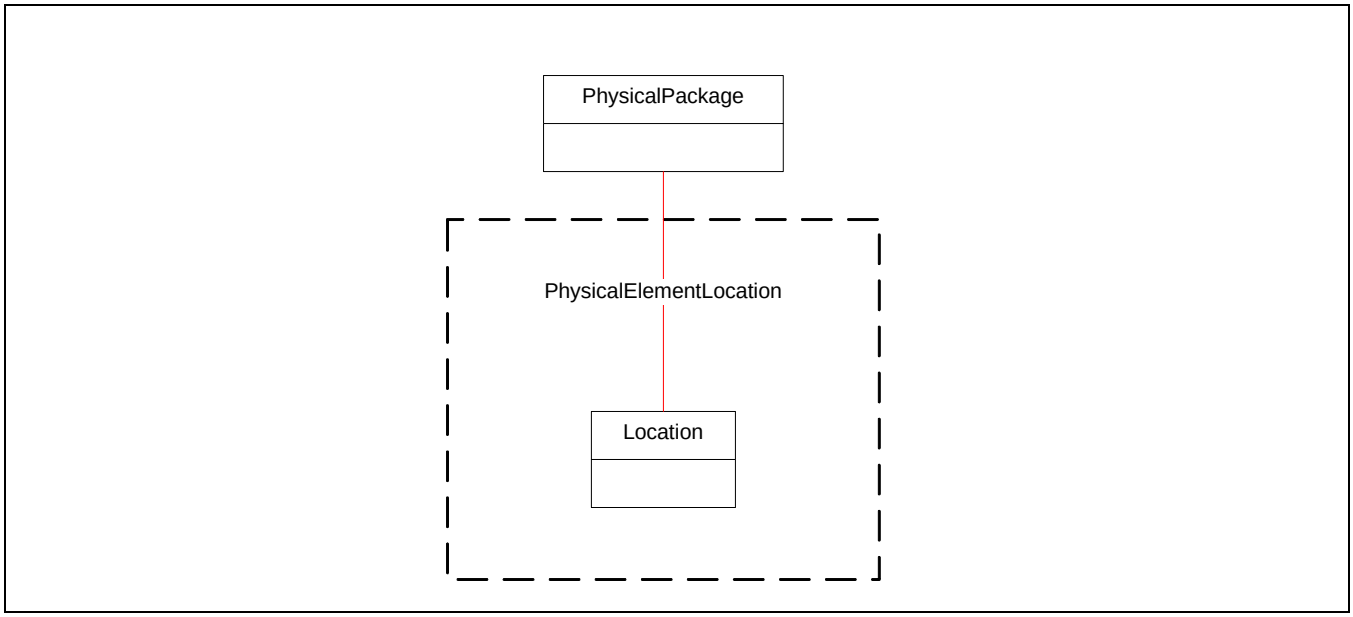


Figure 23 - Location Instance

7.3.3.6.6 Durable names and correlatable IDs

The Location subprofile does not add any durable names or correlatable IDs to the profiles (or subprofiles) in which it is used.

7.3.3.6.7 Methods

The Location subprofile is populated by providers and is accessible to clients using basic read and association traversal.

No extrinsics are specified on the Location subprofile.

7.3.3.6.8 Client considerations

None for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

7.3.3.6.9 Recipes

None for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

7.3.3.6.10 Instrumentation requirements

None for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.3.3.6.11 Required CIM Elements

Table 50 - Required CIM Elements

Profile Classes & Associations	Notes
Location (7.3.3.6.12.1)	
PhysicalElementLocation (7.3.3.6.12.2)	Associates the location to product
Packages	
None	
Methods	
None	
SubProfile Indications	
None	

7.3.3.6.12 Required Properties for CIM Elements

7.3.3.6.12.1 Location

The Location class specifies the position and address of a PhysicalElement.

Location is subclassed from ManagedElement.

Table 51 - Required Properties of Location

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
ElementName	string		User Friendly name. This property is OPTIONAL.
Name	string	key, maxlen (256)	A free-form string defining a label for the Location.
PhysicalPosition	string	key, maxlen (256)	A free-form string indicating the placement of a PhysicalElement.
Address		maxlen (1024)	A free-form string indicating a street, building or other type of address for the PhysicalElement's Location. This property is OPTIONAL.

7.3.3.6.12.2 PhysicalElementLocation

PhysicalElementLocation associates a PhysicalElement with a Location object for inventory or replacement purposes.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

PhysicalElementLocation is subclassed from ElementLocation.

Table 52 - Required Properties for PhysicalElementLocation

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Element	ref	key	The PhysicalElement whose Location is specified.
PhysicalLocation	ref	key	The PhysicalElement's Location.

7.3.3.6.12.3 Optional subprofiles

Table 53 - Optional Profiles or Subprofiles

Name	Notes
None	

7.3.3.7 Software Subprofile

7.3.3.7.1 Description

The Software Subprofile is used in the Extender, Router, Management Appliance, Array, Out-of-band Virtualization System and In-band Virtualization System Profiles.

Information on the installed controller software is given using the SoftwareIdentity class. This is linked to the controller using an InstalledSoftwareIdentity association.

7.3.3.7.2 Standard dependencies

The Software package is defined using the CIM Schema 2.8 final. As such it can be used in profiles at 2.8 and later. It does not require that Profiles be on a later schema. It will operate within profiles that are at the CIM schema 2.8 final or later. The package will operate correctly with CIM Specification 2.2 (or later) and CIM Operations over HTTP 1.1 (or later).

7.3.3.7.3 Profile dependencies

The Software subprofile introduces no Profile dependencies.

7.3.3.7.4 CIM Server requirements

For the SMI-S uses of the Software subprofile, support for Basic Read and Association Traversal functional profiles shall be supported by the base Profile CIM server.

The Software subprofile does not require support for extrinsic methods.

The Software subprofile is not advertised.

7.3.3.7.5 Instance Diagram

See the Software Package Instance Diagram.

7.3.3.7.6 Durable names and correlatable IDs

The Software Subprofile does not add any durable names or correlatable IDs to the profiles (or subprofiles) in which it is used.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.3.7.7 Methods

The Software Subprofile is populated by providers and is accessible to clients using basic read and association traversal.

No extrinsics are specified on the Software Subprofile.

7.3.3.7.8 Client considerations

None for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

7.3.3.7.9 Recipes

None for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

7.3.3.7.10 Instrumentation requirements

None for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

7.3.3.7.11 Required CIM Elements

Table 54 - Required CIM Elements

Profile Classes & Associations	Notes
InstalledSoftwareIdentity (7.3.2.3.12.1)	See under Software Package
SoftwareIdentity (7.3.2.3.12.2)	See under Software Package
Packages	
Software Package	
Associated Indications	
None	

7.3.3.7.12 Required Properties for CIM Elements

See the Required Properties for CIM Elements for the Software Package.

7.3.3.7.13 Optional subprofiles

Table 55 - Optional Profiles or Subprofiles

Name	Notes
None	

7.3.3.8 Copy Services Subprofile

7.3.3.8.1 Description

The Copy Service Subprofile is an optional subprofile for the Array, Out-of-Band Virtualization and In-band Virtualization Profiles.

The copy services profile within the SMI-S object model allows vendors to express management functionality to support clones and point-in-time snapshots of non-volatile storage. This profile also provides support for remote replication services (either asynchronous or synchronous) for non-volatile storage. In this release of

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

this standard, copy services applies to volumes. While copy services functionality is broad in scope and represents vital functionality in any enterprise storage environment and the time of this standard publication, copy services design has principally only been validated for use in support of volume snapshots.

Copy services are addressed by the StorageSynchronized association between two storage elements. The model addresses the use of StorageSynchronized in the context of StorageVolumes (Disk arrays and virtualization systems). In addition, the copy services are designed to support synchronization of file system elements (A.11, “Synchronization of file system elements through copy services”). However, the specifics of this support are not addressed in this standard.

The design for copy services is based the StorageSynchronized construct that defines the relationship between volumes.

In addition to this construct, the design identifies the methods to support the copy services.

7.3.3.8.2 Standard dependencies

The Copy Services subprofile is based on the following standards:

Table 56 - Copy Services Standard Dependencies

Standard	Version	Organization
CIM Specification	2.2	DMTF
CIM Operations over HTTP	1.1	DMTF
CIM Schema	2.8 Preliminary	DMTF

7.3.3.8.3 Profile dependencies

The Copy Services subprofile does not require any other Profiles.

7.3.3.8.4 CIM Server requirements

7.3.3.8.4.1 Functional Profiles

Table 57 - Required Functional Profiles

Profile Required	Functional Group	Dependency
YES	Basic Read	None
YES	Basic Write	Basic Read
YES	Instance Manipulation	Basic Write
NO	Schema Manipulation	Instance Manipulation
YES	Association Traversal	Basic Read
NO	Query Execution	Basic Read
NO	Qualifier Declaration	Schema Manipulation
YES	Indication	None

7.3.3.8.4.2 Extrinsic methods

The CIM Server shall support extrinsic methods for the Copy Services subprofile.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.3.8.4.3 Discovery

The Copy Services subprofile, as currently defined, is not an advertised subprofile.

7.3.3.8.5 Instance diagrams for Copy Services

7.3.3.8.5.1 Overview

The following diagram shows the basic model for copy services

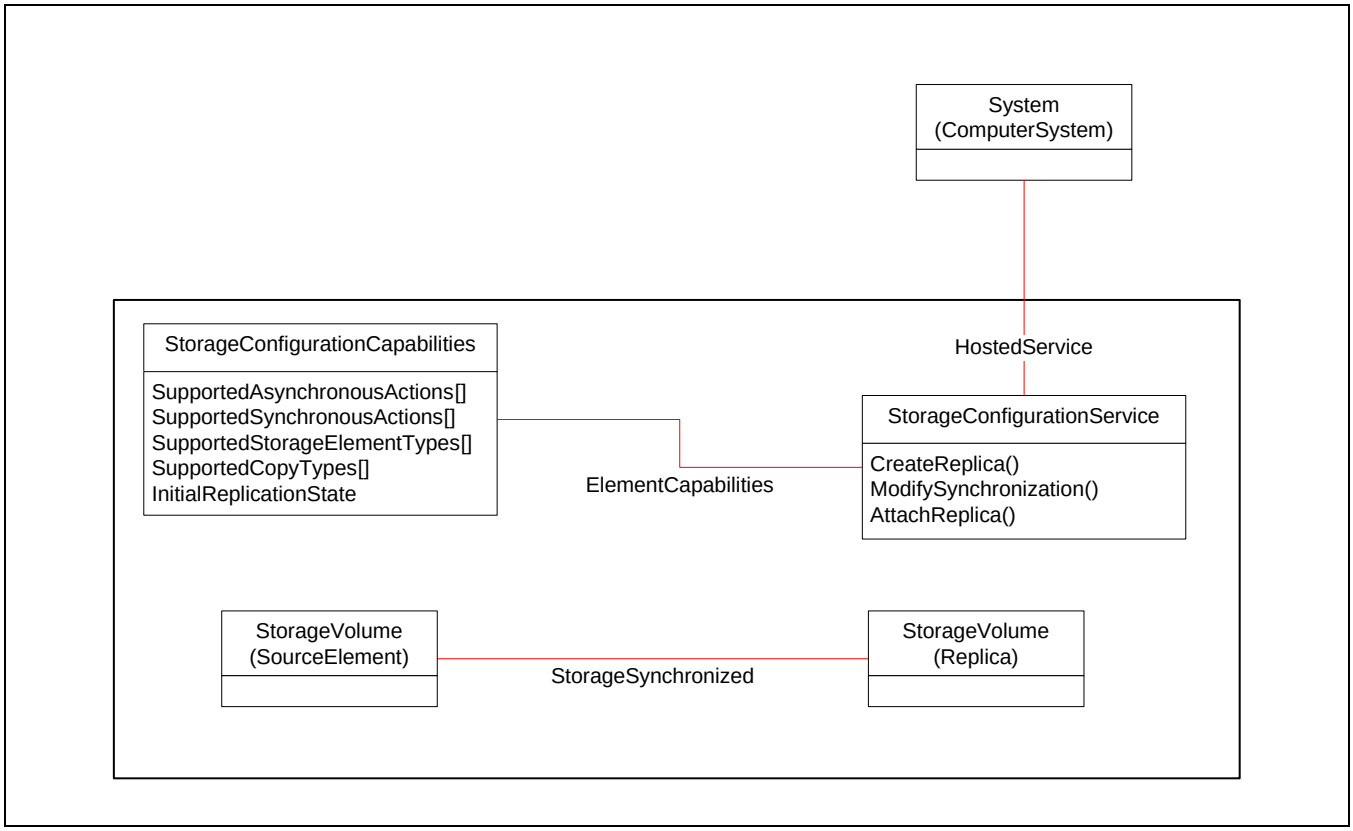


Figure 24 - Instance Diagram for Copy Services

NOTE For simplicity, the StorageCapabilities, StorageSetting and ElementSettingData classes are not shown.

The copy services subprofile depends upon the CIM elements shown in the dashed box in Figure 24. The StorageSynchronized is the basic construct for establishing the replication relationship between storage elements (e.g., StorageVolumes). The methods used are supported in the StorageConfigurationService, which is hosted (HostedService) on the system that represents the storage device that supports the service. The specific Copy Service capabilities supported are specified in the StorageConfigurationCapabilities instance (using the properties shown in Figure 24).

The the model constructs and their defined properties are detailed in 7.3.3.8.11 and its subclauses. Manipulation of model elements is covered in 7.3.3.8.7, “Methods for Copy Services”.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.3.3.8.5.2 StorageSynchronized

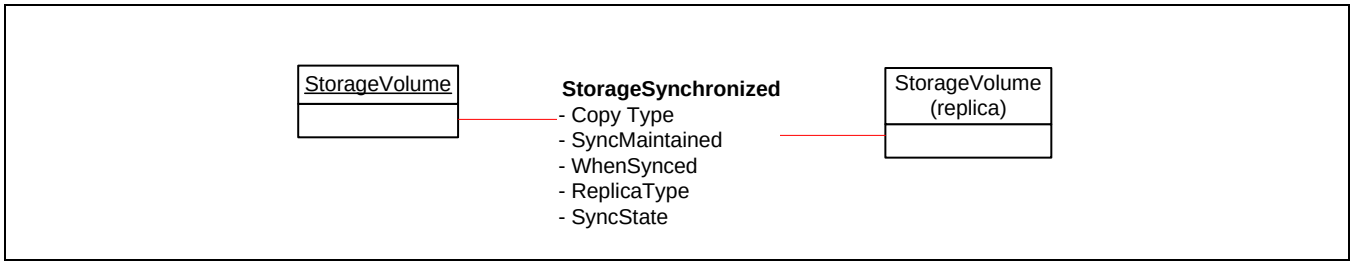


Figure 25 - StorageSynchronized Association

The primary model construct for replication services is the StorageSynchronized association (illustrated in Figure 29). This association is subclassed from Synchronized and is used to represent snapshots and mirror copies of storage elements. In the context of this release of SMI-S, the focus is on the use of StorageSynchronized to model copies of volumes. However, the actual definition of the StorageSynchronized also accommodates a broader interpretation of the storage elements that can be synchronized (e.g., file systems).

The StorageSynchronized association is defined as an association that “Indicates that two storage objects were replicated at the specified point in time. If the CopyType property is set to ‘Sync’ (=3), then synchronization of the storage object is preserved.”

The properties of the StorageSynchronized association are:

SystemElement (Key): The SystemElement represents the storage that is the source of the replication.

SyncedElement (key): The SyncedElement represents the storage that is the target of the replication.

CopyType: The CopyType describes the replication policy. CopyType has four values:

> **Async:** Create and maintain an asynchronous copy of the source. Updates to the source volume are not immediately available on the copy. The copy is done in the background. Typically, this is used to maintain a geographically remote copy of the source volume where the latency overhead of write synchronization would be too expensive or too slow.
>
> **Sync:** Create and maintain a synchronous copy of the source. Writes done to the source volume are reflected on the replica volume before control is returned to the host that issued the write to the source volume.
>
> **UnSyncAssoc:** Create an unsynchronized copy and maintain an association to the source. This type of copy is generally referred to as a persistent “Snapshot” or “point in time” copy.
>
> **UnSyncUnAssoc:** Create an unsynchronized copy, but do not maintain the association with the source. This is a simple copy of a volume, where there is NO StorageSynchronized instantiated between the source and the replica.

SyncMaintained: This property indicates whether (or not) the synchronization is being maintained on an ongoing basis. SyncMaintained can be true for either synchronous or asynchronous copies. SyncMaintained changes from true to false when a “fracture” service is issued or when the relationship is “broken.” The SyncMaintained changes from false to true when a “resync” service is invoked. The SyncMaintained is always false if the CopyType is UnSyncAssoc (e.g., a point in time copy).

WhenSynced: This property is only meaningful if SyncMaintained is false. In this case, it indicates the time/ date of the last synchronization. WhenSynced is not maintained while the SyncMaintained is true. That is, it is not updated on every IO for an Async or Sync relationship that has not been broken (fractured). It represents the time of the last fracture or resync issued on the association.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

NOTE 1 If the synchronization has not been established, then the value of WhenSynced is null.

NOTE 2 The properties “SyncMaintained” and “WhenSynced” are inherited from the Synchronized class, from which StorageSynchronized is subclassed.

The following are meaningful combinations of these properties, with their common nomenclature.

Table 58 - Name Formats

CopyType	Sync Maintained	When Synced	Notes
Sync	True	Null or Datetime	An ongoing mirror relationship exists and the mirror is identical to the source.
	False	Datetime	A synchronous copy relationship has been fractured (split or broken mirror). The “whensynced” value is the time of the last fracture.
	False	Null	This means the relationship has been established, but the copy has not been initiated.
Async	True	Null or Datetime	An ongoing, but delayed mirror relationship exists. The replica may be slightly out of date.
	False	Date/Time	An asynchronous copy relationship has been fractured (split or broken mirror). The “whensynced” value indicates the time of the last fracture.
	False	Null	This means the relationship has been established, but the copy has not been initiated.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table 58 - Name Formats (Continued)

CopyType	Sync Maintained	When Synced	Notes
UnSyncAssoc	False	Null	This means the relationship has been established, but the copy has not been initiated. A snapshot relationship after the replica has been modified. NOTE Some implementations may not allow this state.
	False	Date/Time	A snapshot relationship where the "whensynced" value is the time of the last snapshot.
UnSyncUnAssoc	False	Date/Time	This state does not actually exist. It corresponds to the service where a snapshot is taken but the relationship is not maintained (e.g., logically deleted as soon as the snapshot is done).

In addition there are other properties that add information about the relationship:

ReplicaType: ReplicaType is an optional property for describing how the replica is maintained. The types are Copy, before delta, after delta, log or Not Specified. This is an informational property that backup tools might want to exploit. This is particularly useful when used in association with Snapshots (UnSyncAssoc).

SyncState: The status property addresses the state of the copy operation represented by the association. The state values are:

Fracture In Progress – A fracture has been requested and is in progress.

ReSync In Progress – A resync has been requested and is in progress (the replica not in sync yet).

Restore In Progress – A restore has been requested and is in progress.

Prepare In Progress – A Prepare has been requested and is in progress.

Quiesce In Progress – A quiesce has been requested and is in progress.

Fractured – The relationship has been fractured and is ready for a resync or restore request. This status also appears for UnSyncAssoc, indicating that the point in time copy (as of the WhenSynced date) has been completed.

Synchronized – The Async or Sync relationship is active and copying is going on.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Prepared – The association is in a Prepared state and ready to be Resync'd.

Quiesced – The association is in a quiesced state and ready to be Fractured.

Broken – The source element and the replica have gotten out of sync. Repair actions are required to re-establish the relationship. The repair action would be a Resync or a RestoreFromReplica.

Initialized – The relationship has been established, the copy has not been initiated.

Idle – The relationship has been established, the copy (Resync) has been initiated and completed (for UnSyncAssoc associations).

A source object may have multiple replicas (each with their own StorageSynchronized association and each of the same or different copy types). That is, one source can have many targets (replicas), but one replica can have only one source.

NOTE 3 Any implementation may place a restriction on the number of replicas that can be made from a single source. In the extreme case, the maximum number of replicas would be zero (that is, none). This would indicate the replication is not supported for the source storage object.

It should also be noted that the replica may, itself, be the source of another replica. That is, the architecture allows a copy to be copied.

7.3.3.8.6 Durable names and correlatable IDs

Copy services uses and exports the following durable names:

- StorageVolumes (StorageVolume.name)

For StorageVolumes the Durable Name is the StorageVolume.name and the format for the name is defined by the NameFormat property. See Table 3 for the valid formats for StorageVolumes.

7.3.3.8.7 Methods for Copy Services

The following extrinsic methods are part of storage configuration service that are defined to support replication (and the StorageSynchronized association).

Create Replica: This service creates a new storage object that is a replica of the source storage object. Based on the CopyType property, the service can be used to instantiate the replica and to create StorageSynchronized association between the source and the replica. When creating a replica, a target StoragePool and a target setting can be specified. If the target setting is not specified, a default setting is used. If the target StoragePool is not specified, the default assumption is the same pool as the source storage object.

This service takes as input the reference to the source storage object, the StorageSetting to be maintained for the target storage object (the replica), the target StoragePool (optional) and the CopyType (see the CopyType property of StorageSynchronized). The output of this service is a reference to the target storage object.

CreateReplica():

[In, Description ("A end user relevant name for the element being created. If NULL, then a system supplied default name can be used. The value will be stored in the 'ElementName' property for the created element.")]

string **ElementName**,

[Out, IN(false), Description("Reference to the job (may be null if job completed).")]

CIM_ConcreteJob REF **Job**,

[In, Required, Description("The source storage object.")]

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

CIM_LogicalElement REF **SourceElement**,

[Out, IN(false), Description("Reference to the created target storage element (i.e., the replica).")]

CIM_LogicalElement REF **TargetElement**,

[In, Description("The definition for the StorageSetting to be maintained by the target storage object (the replica).")]

CIM_StorageSetting REF **TargetSettingGoal**,

[In, Description("The underlying storage for the target element (the replica) will be drawn from TargetPool if specified, otherwise the allocation is implementation specific.")]

CIM_StoragePool REF **TargetPool**,

[In, Description("CopyType describes the type of copy that will be made. Values are:

Async: Create and maintain an asynchronous copy of the source.

Sync: Create and maintain a synchronized copy of the source.

UnSyncAssoc: Create an unsynchronized copy and maintain an association to the source.

UnSyncUnAssoc: Create unassociated copy of the source element."),

ValueMap {"2", "3", "4", "5", "..", "0x8000.."},

Values {"Async", "Sync", "UnSyncAssoc", "UnSyncUnAssoc", "DMTF Reserved", "Vendor Specific"}]

Uint16 **CopyType**

ModifySynchronization: This service specifies a modification to the StorageSynchronized association. It takes as input a reference to the association and the type of modification desired. The types of modification defined are: Detach, Fracture, ResyncReplica RestoreFromReplica, Prepare, Unprepare, Quiesce, Unquiesce, ResetToSync and ResetToAsync. These types of modifications are defined as:

Detach: This deletes a StorageSynchronized association and effectively breaks the relationship between a replica and its source. It does not delete the replica, but makes it a "normal" storage object. So, detaching the replica of a StorageVolume would turn the replica into a "normal" storage volume.

Fracture: This suspends the synchronization between the two storage objects (changes SyncMaintained to false and sets the time/date for the time of last synchronization). The association is not deleted (assumes there may be a future "resync"). This function is sometimes called "breaking" or "splitting" a mirror.

NOTE FractureReplica only applies to CopyTypes of Async or Sync. It has no meaning to an UnSyncAssoc (or UnSyncUnAssoc) type.

Resync Replica: This modification re-establishes the synchronization between the source and the replica. If the CopyType of the association was either Sync or Async, it returns the relationship to Sync or Async (respectively). If the copy type is UnSyncAssoc, the resync re-derives the snapshot (e.g., take another snapshot and reset the WhenSynced value).

Restore From Replica: This modification asks that the source storage object be restored from the replica. In effect, this reverses the copy. That is, the replica is copied back to the source object. This has the effect of restoring the source volume to the state in the replica.

Prepare: This modification indicates that the agent should get prepared for a Resync action. Some implementations require this to be invoked before the ResyncReplica.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Unprepare: This modification is required clear a quiesced state if a Prepare is not followed by a ResyncReplica.

Quiesce: This modification indicates that the agent should get prepared for a Fracture action. Some implementations require this to be invoked before the Fracture.

Unquiesce: This modification is required clear a quiesced state if a Quiesce is not followed by a Fracture.

Reset To Sync: This modification changes the association to the “Sync” CopyType (e.g., from the “Async” CopyType).

Reset To Async: This modification changes the association to the “Async” CopyType (e.g., from the “Sync” CopyType).

ModifySynchronization():

[In, Description(“Operation describes the type of modification to be made to the replica. Values are:

Detach: 'Forget' the synchronization between two storage objects. Start to treat the objects as independent.

Fracture: Suspend the synchronization between two storage objects. The association and (typically) changes are remembered to allow a fast re-synchronization. This may be used during a backup cycle to allow one of the objects to be copied while the other remains in production.

Resync Replica: Re-establish the synchronization of a replica. If CopyJob is Sync or Async, this negates the action of a previous Fracture operation.

Restore from Replica: Renew the contents of the original storage object from a replica.

Prepare: Place the association in a quiesced state when a prepared state is required for a Resync.

Unprepare: Take the association out of the prepared state without issuing a Resync.

Quiesce: Place the association in a quiesced state when a quiesced state is required for a Fracture.

Unquiesce: Take the association out of the quiesced state without issuing a Fracture.

Reset To Sync: Change the CopyType to “Sync”.

Reset To Async: Change the CopyType to “Async”.

ValueMap {“0”,”1”,”2”, “3”, “4”, “5”, “6”, “7”, “8”, “9”, “10”,”11”, “..”, “0x8000..”},

Values {“DMTF Reserved”, “DMTF Reserved”, “Detach”, “Fracture”, “Resync Replica”, “Restore from Replica”, “Prepare”, “Unprepare”, “Quiesce”, “Unquiesce”, “Reset To Sync”, “Reset To Async”, “DMTF Reserved”, “Vendor Specific”}]

Uint16 **Operation**,

[Out, IN(false), Description(“Reference to the job (may be null if job completed).”)]

CIM_ConcreteJobREF **Job**,

[In, Description(“The referenced to the StorageSynchronized association describing the storage source/ replica relationship.”)]

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

CIM_StorageSynchronized REF **Synchronization**

AttachReplica – This function creates a StorageSynchronized relationship between two (existing) storage volumes. Once the association is created the SyncState is set to “initialized”, “Prepared” or “Synchronized” as defined in the StorageConfigurationCapabilities associated with the StorageConfigurationService. There is no ConcreteJob created or returned on this method call (since the only action effected is the creation of the association).

AttachReplica():

[In, Description (“A end user relevant name for the element being created. If NULL, then a system supplied default name can be used. The value will be stored in the

'ElementName' property for the created element”)]

string **ElementName**,

[In, Required, Description(“The source storage object.”)]

CIM_LogicalElement REF **SourceElement**,

[In, Required, Description(“Reference to the target storage element (i.e., the replica).”)]

CIM_LogicalElement REF **TargetElement,**

[In, Required, Description(“CopyType describes the type of copy that will be made. Values are:

Async: Create and maintain an asynchronous copy of the source.

Sync: Create and maintain a synchronized copy of the source.

UnSyncAssoc: Create an unsynchronized copy and maintain an association to the source.

UnSyncUnAssoc: Create unassociated copy of the source element.”),

ValueMap {“2”, “3”, “4”, “5”, “.”, “0x8000..”},

Values {“Async”, “Sync”, “UnSyncAssoc”, “UnSyncUnAssoc”, “DMTF Reserved”, “Vendor Specific”}]

Uint16 **CopyType**

[Out, IN(false), Description(“Reference to the job (may be null if job completed).”)]

CIM_ConcreteJob REF **Job**,

7.3.3.8.8 Client Considerations for Copy Services

7.3.3.8.8.1 Determining the Type of Copy Services Supported

Copy Services come in multiple types and variations. To support a client’s ability to recognize the functions and capabilities provided there is a StorageConfigurationCapabilities instance that is associated with the StorageConfigurationService. A client can determine the exact support provided by the profile in question by inspecting the following properties of the StorageConfigurationCapabilities:

a) SupportedCopyTypes - identifies the types of copies (CopyType input to CreateReplica or AttachReplica) that are supported.

-`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

- Async - means Asynchronous mirroring is supported
- Sync - means Synchronous mirroring is supported
- UnSyncAssoc - means Snapshots are supported and the association persists after a snapshot is taken
- UnSyncUnAssoc - means Snapshots are supported and the association is automatically detached after a Resync

b) SupportedSynchronousActions - This is an array that indicates which methods are supported without the use of jobs. For Copy Services, the actions to look for are “Replica Creation”, “Replica Attach”, “Replica Synchronization”.

- Replica Creation - means jobs may not be returned on CreateReplica
- Replica Attachment - means jobs may not be returned on AttachReplica
- Replica Modification - means jobs may not be returned on ModifySynchronization

c) SupportedAsynchronousActions - This is an array that indicates whether or not jobs may be created based on the actions listed. For Copy Services, the actions to look for are “Replica Creation”, “Replica Attach”, “Replica Synchronization”.

- Replica Creation - means jobs may be returned on CreateReplica
- Replica Attachment - means jobs may be returned on AttachReplica
- Replica Modification - means jobs may be returned on ModifySynchronization

d) SupportedStorageElementTypes - This is an array that indicates the types of storage that are supported. For the Copy Services subprofile “StorageVolume” shall be present.

e) InitialReplicationState - indicates the initial state that results from a CreateReplica or AttachReplica

- Initialized - means the SyncState is initialized as a result of creating a StorageSynchronized
- Prepared - means the SyncState is Prepared as a result of creating a StorageSynchronized
- Synchronized - means the SyncState is Synchronized as a result of creating a StorageSynchronized

By inspecting these StorageConfigurationCapabilities, the client can determine the types and variations of copy services supported.

7.3.3.8.8.2 Creating a Copy Relationship between two existing Volumes

A client can establish a copy relationship between two existing volumes by simply issuing a AttachReplica to create the StorageSynchronized association. Once the association is created the SyncState is set to “initialized”, “Prepared” or “Synchronized” as defined in the StorageConfigurationCapabilities associated with the StorageConfigurationService.

If the relationship is left in the “initialized” state, then it will be necessary to Prepare the relationship before issuing a “Resync Replica” to effect the copy action. If the relationship is left in the “Prepared” state, then it will be necessary to issue a “Resync Replica” to effect the copy action. If the relationship is left in the “Synchronized” state, no action is required to effect the copy action. It has been performed.

7.3.3.8.8.3 Creating a Point-in-Time Copy of a Volume

If the replica volume has not been created, a client could create a point in time copy by issuing the method CreateReplica of the StorageConfigurationService passing the Source volume, a CopyType of UnSyncAssoc

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

or UnSyncUnAssoc. Either of these result in a point in time copy of the source volume. In addition, the client may optionally specify the StorageSetting to be supported by the replica.

The client gets the target volume reference as an output of the method call. In addition, if the point in time copy is not done at the time when the response is returned, the client may get a ConcreteJob reference to monitor completion of the copy.

If the CopyType specified was UnSyncAssoc, then a StorageSynchronized relationship is established between the source volume and the replica. The CopyType is set to “UnSyncAssoc”, the WhenSync value is the effective time of the copy and SyncMaintained is set to False. In addition, the ReplicaType (if supported) identifies the type of point-in-time copy that was taken. The SyncState is set to “initialized”, “Prepared” or “Synchronized” as defined in the StorageConfigurationCapabilities associated with the StorageConfigurationService.

If the relationship is left in the “initialized” state, then it will be necessary to Prepare the relationship before issuing a “Resync Replica” to effect the copy action. If the relationship is left in the “Prepared” state, then it will be necessary to issue a “Resync Replica” to effect the copy action. If the relationship is left in the “Synchronized” state, no action is required to effect the copy action. It has been performed.

7.3.3.8.8.4 Creating a Synchronous Copy of a Volume

If the replica volume has not been created, a client could create a synchronous copy on the volume by issuing the method CreateReplica of the StorageConfiguarionService passing the Source Volume and a CopyType of Sync. This results in a synchronous copy relationship of the source volume. In addition, the client may optionally specify the StorageSetting to be supported by the replicas.

The client gets the target StorageVolume reference as an output of the method call. In addition, if the synchronous copy is not done at the time when the response is returned, the client gets a ConcreteJob reference to monitor completion of the copy.

A StorageSynchronized relationship is established between the source volume and the replica. The CopyType is set to “Sync”, the WhenSync value is null and SyncMaintained is set to True. In addition, the ReplicaType (if supported) identifies the type of synchronous copy that was taken. SyncState is set to “initialized”, “Prepared” or “Synchronized” as defined in the StorageConfigurationCapabilities associated with the StorageConfigurationService.

If the relationship is left in the “initialized” state, then it will be necessary to Prepare the relationship before issuing a “Resync Replica” to effect the copy action. If the relationship is left in the “Prepared” state, then it will be necessary to issue a “Resync Replica” to effect the copy action. If the relationship is left in the “Synchronized” state, no action is required to effect the copy action. It has been performed.

7.3.3.8.8.5 Creating an Asynchronous Copy of a Volume

If the replica volume has not been created, a client could create an asynchronous copy on the volume by issuing the method CreateReplica of the StorageConfiguarionService passing the Source Volume and a CopyType of Async. This results in an asynchronous copy relationship of the source volume. In addition, the client may optionally specify the StorageSetting to be supported by the replicas.

The client gets the target StorageVolume reference as an output of the method call. In addition, if the asynchronous copy is not done at the time when the response is returned, the client gets a ConcreteJob reference to monitor completion of the copy.

A StorageSynchronized relationship is established between the source volume and the replica. The CopyType is set to “Async”, the WhenSync value is null and SyncMaintained is set to True. In addition, the ReplicaType (if supported) identifies the type of asynchronous copy that was taken. The SyncState is set to “initialized”, “Prepared” or “Synchronized” as defined in the StorageConfigurationCapabilities associated with the StorageConfigurationService.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

If the relationship is left in the “initialized” state, then it will be necessary to Prepare the relationship before issuing a “Resync Replica” to effect the copy action. If the relationship is left in the “Prepared” state, then it will be necessary to issue a “Resync Replica” to effect the copy action. If the relationship is left in the “Synchronized” state, no action is required to effect the copy action. It has been performed.

7.3.3.8.8.6 Splitting Mirrored Volumes

A client can split a pair of mirrored volumes by issuing the Quiesce and Fracture options of the ModifySynchronization method call. This method call only works when CopyType is “Sync” or “Async.” In the case of an “Async” copy type, a “Prepare and Resync Replica” is implied by the Quiesce. That is, the copies are synchronized before the fracture is effected.

Once mirrors have been split, the client has a copy with the effective time (WhenSynced) that is the time of the Fracture.

7.3.3.8.8.7 Re-establishing the Mirrored relationship between Volumes

After a fracture has been done on mirrored volumes, the mirrored relationship can be re-established by issuing the Resync Replica option of the ModifySynchronization method.

7.3.3.8.8.8 Getting notification of Completion of Copy Actions Initiated

A client can monitor the progress of copy methods that it has initiated by monitoring the PercentComplete property of the ConcreteJob returned on the original copy action. The client would do this by subscribing to the PercentComplete InstModificaiton indication.

The client can also subscribe to JobStatus, just looking for completion of the job (rather than tracking the percent complete property). By monitoring the JobStatus, the client would get the JobStatus (normal or abnormal end) when the indication is raised.

NOTE GUIs would typically monitor PercentComplete so they can report the progress to their end users. However, programs that are not directly interacting with end users would typically just subscribe to the JobStatus changes.

7.3.3.8.8.9 Breaking Copy Relationships between Volumes

A client can delete the copy relationship between volumes by issuing the Detach option of the ModifySynchronization method.

7.3.3.8.9 Recipes for Copy Services

None.

7.3.3.8.10 Instrumentation Requirements for Copy Services

7.3.3.8.10.1 Implementation Restrictions

The intent of the architecture for copy services is not to dictate implementations, but rather to establish a consistent way of specifying the services that are implemented. For example, some implementations may not support creation of Replicas as defined for CreateReplica. That is, they may only operate on volumes that are created by the using client (or application). This would be a valid implementation. The StorageSynchronized is supported by the AttachReplica method. All implementations shall supply the “Replica Modification” in the SupportedSynchronousActions or SupportedAsynchronousActions arrays of StorageConfigurationCapabilities. An implementation shall also supply at least one of “Replica Creation” or “Replica Attachment” in at least one of the same arrays. An implementation may supply only one of the two.

Similarly, not all implementations are expected to support all CopyTypes. This too is allowed. An implementation shall identify the CopyTypes supported in the StorageConfigurationCapabilities.SupportedCopyTypes array.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

For the Copy Services subprofile, implementations shall support one and only one value in the StorageConfigurationCapabilities.InitialReplicationState. This value shall be one of “Initialized”, “Prepared” or “Synchronized”.

For this version of the specification any implementation of the Copy Services subprofile shall set “StorageVolume” in the StorageConfigurationCapabilities.SupportedStorageElementTypes array.

In addition, the implementation shall supply the Methods for which it supports job control. This is done in the StorageConfigurationCapabilities.SupportedAsynchronousActions array.

7.3.3.8.10.2 Mapping of SyncState information

It is expected that various implementations of the copy services will have state information that does not exactly match that of this architecture. Where possible and practical, implementations will map to the standard architected states to enable applications that utilize those states. In cases where using the architected state would be misleading, implementations are encouraged to use the “Vendor Specific” state to avoid misleading the applications.

7.3.3.8.10.3 Resync after Fracture Considerations

When a StorageSynchronized association is modified with a “fracture” request, the agent may want to consider “remembering” changes to the source volume. Typically, a “fracture” request will be followed by a “resync” request. This resync will go a lot faster if the device was maintaining a change log between the fracture and the resync.

7.3.3.8.10.4 State Diagrams for Snapshot (UnSyncAssoc)

The Figure 26 illustrates the state changes that would be supported for Snapshots. The solid arrows between states are application provoked state changes. The dashed arrows are automatic state changes as a result of completion of the previous state.

There are 3 possible ways on entering this state diagram. In any case, entry is done through CreateReplica or AttachReplica. This is determined based on the StorageConfigurationCapabilities.InitialReplicationState value. The Create or Attach creates a StorageSynchronized association and puts it in either the “initialized”, “Prepared” or “Synchronized” state. If it is put in the initialized state, a Prepare shall be issued before issuing a Resync.

Once a Prepare has finished, the state moves to Prepared. At this point, the client may issue the Resync to drive the Snapshot. If the client decides against doing the Resync, the client will issue an Unprepare.

After a Resync, the association is put in the ResyncInProgress state until the copy is complete. Once the copy is complete, the association is put in the idle state.

--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

At any point in time, except the “in progress” states, a detach may be issued to delete the StorageSynchronized association.

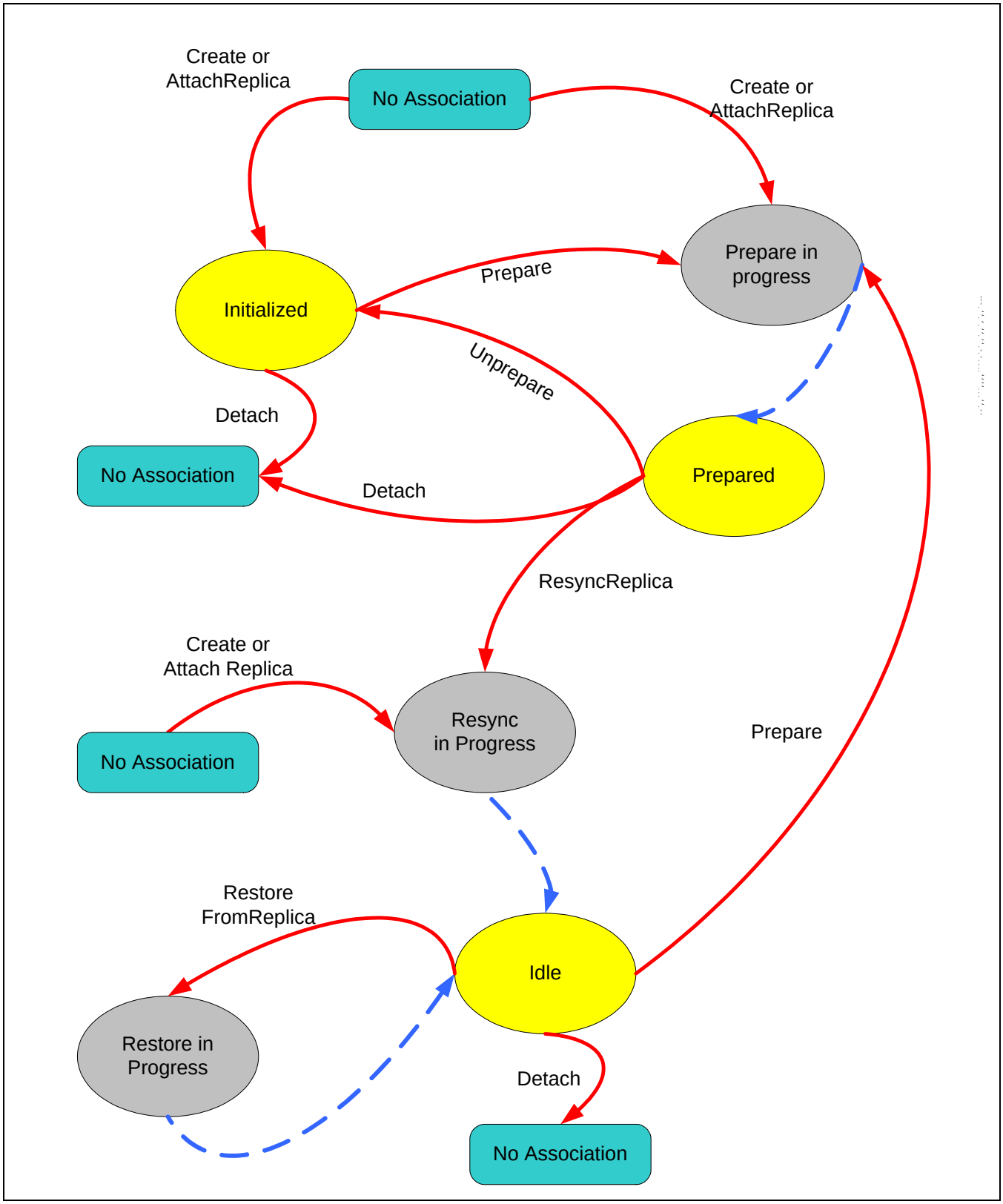


Figure 26 - State Diagram for Snapshot

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.3.3.8.10.5 State Diagrams for Mirrors (Sync or Async)

The Figure 27 illustrates the state changes that would be supported for Mirrors. The solid arrows between states are application provoked state changes. The dashed arrows are automatic state changes as a result of completion of the previous stateservices.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

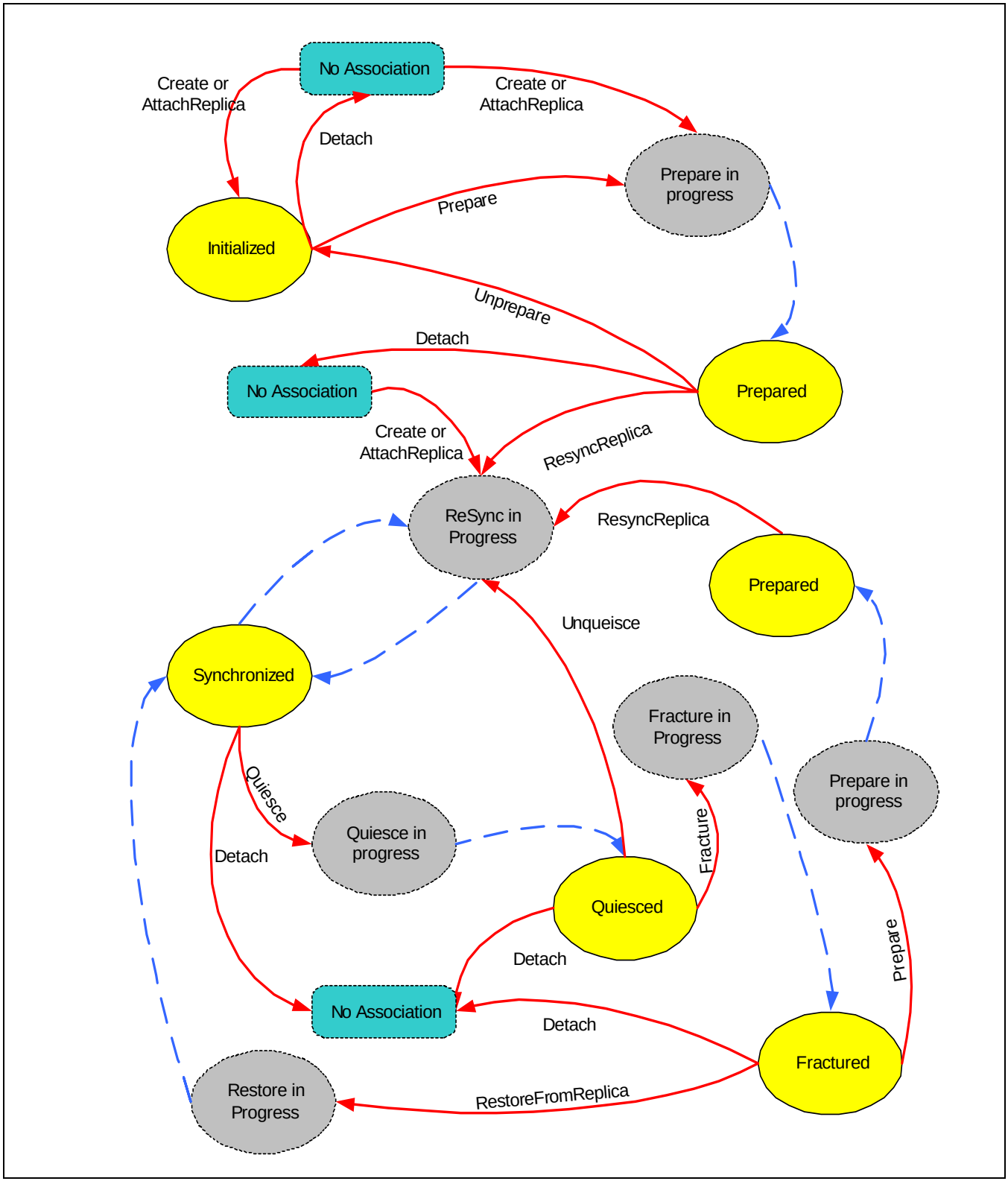


Figure 27 - State Diagram for Mirrors

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

There are 3 possible ways on entering this state diagram. In any case, entry is done through CreateReplica or AttachReplica. This is determined based on the StorageConfigurationCapabilities.InitialReplicationState value. The Create or Attach creates a StorageSynchronized association and puts it in either the “initialized”, “Prepared” or “Synchronized” state. If it is put in the initialized state, a Prepare shall be issued before issuing a Resync.

Once a Prepare has finished, the state moves to Prepared. At this point, the client may issue the Resync to drive the actual copy functions. If the client decides against doing the Resync, the client will issue an Unprepare.

After a Resync, the association is in the ResyncInProgress state until the copy is in the “synchronized” state. For Sync copies this may take a while. Even for Async, it may take awhile. Once the copy is Synchronized, the association is put in the “Synchronized” state.

In the case of mirrors, there is an additional state of “Quiesced”. A Quiesce is required before a Fracture. This basically gets the mirrors in sync for the fracture. For Sync mirrors, this may be trivial. For Async mirrors, this may be more involved.

At any point in time, except the “in progress” states, a detach may be issued to delete the StorageSynchronized association.

--`,,```,,,,````-`-,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.3.8.11 Required CIM Elements

Table 59 - Subprofile Required Classes, Associations, Methods and Indications

Subprofile Class & Associations	Notes
ElementCapabilities	Associates the StorageConfigurationCapabilities to the StorageConfigurationService and associates StorageCapabilities to the target StoragePool of a CreateReplica
HostedService	
StorageConfigurationService	
StorageConfigurationCapabilities	This identifies the specific capabilities supported in the StorageConfigurationService
StorageVolume	
StorageSynchronized	
StorageCapabilities (7.3.3.8.12.7)	
ElementSettingData (7.3.3.8.12.8)	
StorageSetting (7.3.3.8.12.9)	
Profile Methods	**Notes**
CreateReplica()	Creates a replica (volume), establishes the StorageSynchronized relationship to source and initiates the copy operation
AttachReplica()	Establishes the StorageSynchronized relationship between source and an existing replica
ModifySynchronization()	Used to modify the state of a StorageSynchroinized relationship (e.g., Fracture, Resync, Restore, ...)
Profile Indications	**Notes**
Creation/Deletion of StorageSynchronized	SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_StorageSyncrhonized SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_StorageSynchronized
Change in status for StorageSynchronized	SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_StorageSynchronized AND SourceInstance.SyncState <> PreviousInstance.SyncState

7.3.3.8.12 Required Properties for CIM Elements

7.3.3.8.12.1 ElementCapabilities

ElementCapabilities represents the association between ManagedElements and their Capabilities. In the Copy Services Subprofile, the ManagedElement is the StorageConfigurationService. ElementCapabilities describes the existence requirements and context for the referenced instance of ManagedElement

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

(StorageConfigurationService). Specifically, the ManagedElement shall exist and provides the context for the Capabilities.

ElementCapabilities is not subclassed from anything.

Table 60 - Required Properties for ElementCapabilities

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
ManagedElement	ref	key, min(1), max(1)	The managed element (StorageConfigurationService).
Capabilities	ref	key	The Capabilities object associated with the element (service).

7.3.3.8.12.2 HostedService

HostedService is an association between a Service and the System on which the functionality resides. The cardinality of this association is 1-to-many. A System may host many Services. Services are weak with respect to their hosting System. Heuristic: A Service is hosted on the System where the LogicalDevices or SoftwareFeatures that implement the Service are located. The model does not represent Services hosted across multiple systems. This is modelled as an ApplicationSystem that acts as an aggregation point for Services, that are each located on a single host.

HostedService is subclassed from Dependency.

Table 61 - Required Properties for HostedService

Class Properties	Type	Qualifier/ Parameter	Notes
Antecedent	ref	override, max(1), min(1)	The hosting System.
Dependent	ref	override, weak	The Service hosted on the System.

7.3.3.8.12.3 StorageConfigurationService

The StorageConfigurationService is required for the extrinsic methods. It supports (CreateReplica, AttachReplica and ModifySynchronization).

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

StorageConfigurationService is subclassed from Service.

Table 62 - Required Properties for StorageConfigurationService

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
ElementName	string		User Friendly name.
SystemCreationClassName	string	maxlen(256), key, propagated	The scoping System's CreationClassName.
SystemName	string	maxlen(256), key, propagated	The scoping System's Name.
CreationClassName	string	maxlen(256), key	The name of the concrete subclass.
Name	string	maxlen(256), key, override	
CreateReplica()	uint32		
ModifySynchronization()	uint32		
AttachReplica()	uint32		

7.3.3.8.12.4 StorageConfigurationCapabilities

A subclass of Capabilities that defines the Capabilities of a StorageConfigurationService. An instance of StorageConfigurationCapabilities is associated with a StorageConfigurationService using ElementCapabilities.

StorageConfigurationCapabilities is subclassed from Capabilities.

Table 63 - Required Properties for StorageConfigurationCapabilities

Class Properties	Type	Qualifier/ Parameter	Notes
InstanceID	uint16	key	
ElementName	string	req	
SupportedSynchronousActions[ ]	uint16	valuemap	Values {"Replica Creation", "Replica Attachment", "Replica Modification"
SupportedAsynchronousActions[ ]	uint16	valuemap	Values {"Replica Creation", "Replica Attachment", "Replica Modification"
SupportedStorageElementTypes[ ]	uint16	valuemap	Values {"StorageVolume"}
SupportedCopyTypes[ ]	uint16	valuemap	Values {"Async", "Sync", "UnSyncAssoc", "UnSyncUnAssoc", "DMTF Reserved", "Vendor Specific"}
InitialReplicationState	uint16	valuemap	Values {"Initialized", "Prepared", "Synchronized"}

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.3.8.12.5 StorageSynchronized

Indicates that two Storage objects were replicated at the specified point in time. If the CopyType property is set to 'Sync' (=3), then synchronization of the Storage objects is preserved.

For block servers, there are specific uses of StorageSynchronized. The SystemElement and the SyncedElement are defined as StorageExtents. Specifically, these are StorageVolumes or LogicalVolumes.

StorageSynchronized is subclassed from Synchronized.

Table 64 - Required Properties for StorageSynchronized

Class Properties	Type	Qualifier/ Parameter	Notes
WhenSynced	datetime		The point in time that the Elements were synchronized.
SyncMaintained	boolean		Boolean indicating whether synchronization is maintained.
SystemElement	ref		StorageExtent Reference. It identifies the original storage element.
SyncedElement	ref		StorageExtent Reference. Identifies the replica.
CopyType	uint16		CopyType describes the Replication Policy. Values are: Async: create and maintain an Asynchronous copy of the source. Sync: create and maintain a synchronized copy of the source. UnSyncAssoc: create an unsynchronized copy and maintain an association to the source. Values {"Async", "Sync", "UnSyncAssoc", "DMTF Reserved", "Vendor Specific"}
ReplicaType	uint16		This is an informational property that indicates how the replica is being achieved. The values are: FullCopy, BeforeDelta, AfterDelta, Log or NotSpecified. Values {“FullCopy”, “BeforeDelta”, “AfterDelta”, “Log”, “DMTF Reserved”, “NotSpecified”}
SyncState	uint16		This is the current state of the StorageSynchronized association. Values {“Fracture In Progress”, “Sync In Progress”, “Restore In Progress”, “Prepare In Progress”, “Prepared”, “Quiesce In Progress”, “Quiesced”, “Fractured”, “Synchronized”, “Initialized”, “Idle”, “DMTF Reserved”, “Vendor Specific”}

7.3.3.8.12.6 StorageVolume

The Copy Services does not alter the properties of Storage Volumes as supported by the parent profile (Array, Out-of-band Virtualization and In-band Virtualization).

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.3.8.12.7 StorageCapabilities

Table 65 - Required Properties for StorageCapabilities

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
InstanceID	string	key	InstanceID opaquely identifies a unique instance of Capabilities. The InstanceID shall be unique within a namespace.
ElementName	string	override, required	The user friendly name for this instance of Capabilities. In addition, the user friendly name can be used as a index property for a search or query. (Note that ElementName does not have to be unique within a namespace.) If the capabilities are fixed, then this property should be used as a means for the client application to correlate between capabilities and device documentation.
ElementType	uint16		Enumeration indicating the type of instance to which this StorageCapabilities applies. Only '6', StorageConfigurationService and '5' StoragePool are valid.
NoSinglePointOfFailure	boolean		Indicates whether or not the associated instance supports no single point of failure. Values are: FALSE = does not support "no single point of failure" and TRUE = supports "no single point of failure".
NoSinglePointOfFailureDefault	boolean		Indicates the default value for the NoSinglePointOfFailure property.
DataRedundancyMax	uint16	minvalue(1)	DataRedundancyMax describes the maximum number of complete copies of data that can be maintained. Examples would be RAID 5 where 1 copy is maintained and RAID 1 where 2 or more copies are maintained. Possible values are 1 to *n*.
DataRedundancyMin	uint16	minvalue(1)	DataRedundancyMin describes the minimum number of complete copies of data that can be maintained. Examples would be RAID 5 where 1 copy is maintained and RAID 1 where 2 or more copies are maintained. Possible values are 1 to *n*.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table 65 - Required Properties for StorageCapabilities (Continued)

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
DataRedundancyDefault	uint16	minvalue(1)	DataRedundancyDefault describes the default number of complete copies of data that can be maintained. Examples would be RAID 5 where 1 copy is maintained and RAID 1 where 2 or more copies are maintained. Possible values are 1 to *n*.
PackageRedundancyMax	uint16	write(true)	PackageRedundancyMax describes the maximum number of spindles that can be used. Package redundancy describes how many disk spindles can fail without data loss including, at most, one spare. Examples would be RAID5 with a Package Redundancy of 1, RAID6 with 2. Possible values are 0 to *n*.
PackageRedundancyMin	uint16	write(true)	PackageRedundancyMin describes the minimum number of spindles that can be used. Package redundancy describes how many disk spindles can fail without data loss including, at most, one spare. Examples would be RAID5 with a Package Redundancy of 1, RAID6 with 2. Possible values are 0 to *n*.
PackageRedundancyDefault	uint16	write(true)	PackageRedundancyDefault describes the default number of spindles that can be used. Package redundancy describes how many disk spindles can fail without data loss including, at most, one spare. Examples would be RAID5 with a Package Redundancy of 1, RAID6 with 2. Possible values are 0 to *n*.
DeltaReservationMax	uint16	minvalue(1) maxvalue(100)	Delta reservation is a number between 1 (1 %) and a 100 (100 %) that specifies how much space should be reserved in a replica for caching changes. For a complete copy this would be 100 %, but it can be lower in some implementations. This parameter sets the upper limit.
DeltaReservationMin	uint16	minvalue(1) maxvalue(100)	Delta reservation is a number between 1 (1 %) and a 100 (100 %) that specifies how much space should be reserved in a replica for caching changes. For a complete copy this would be 100 %, but it can be lower in some implementations. This parameter sets the lower limit.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table 65 - Required Properties for StorageCapabilities (Continued)

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
DeltaReservationDefault	uint16	minvalue(1) maxvalue(100)	Delta reservation is a number between 1 (1 %) and a 100 (100 %) that specifies how much space should be reserved in a replica for caching changes. For a complete copy this would be 100 %, but it can be lower in some implementations. This parameter sets the default value.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.3.3.8.12.8 ElementSettingData

Table 66 - Required Properties for ElementSettingData

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
ManagedElement	ref	key	The ManagedElement.
SettingData	ref	key	The Setting Data object associated with the ManagedElement.
IsDefault	uint16		An enumerated integer indicating that the referenced setting is a default setting for the element or that this information is unknown."),||ValueMap {"0", "1", "2"}, ||Values {"Unknown", "Is Default", "Is Not Default"}
IsCurrent	uint16		An enumerated integer indicating that the referenced setting is currently being used in the operation of the element or that this information is unknown."),||ValueMap {"0", "1", "2"}, ||Values {"Unknown", "Is Current", "Is Not Current"}

7.3.3.8.12.9 StorageSetting

Table 67 - Required Properties for StorageSetting

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
InstanceID	string	key	InstanceID opaquely identifies a unique instance of SettingData. The InstanceID shall be unique within a namespace.
ElementName	string	override, required	The user friendly name for this instance of SettingData. In addition, the user friendly name can be used as a index property for a search of query. (Note that Name does not have to be unique within a namespace.)
NoSinglePointOfFailure	boolean	write(true)	Indicates the desired value for No Single Point of Failure. Possible values are false = single point of failure and true = no single point of failure.
DataRedundancyMax	uint16	minvalue(1) write(true)	DataRedundancyMax describes the maximum number of complete copies of data to be maintained. Examples would be RAID 5 where 1 copy is maintained and RAID 1 where 2 or more copies are maintained. Possible values are 1 to *n*.

--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table 67 - Required Properties for StorageSetting (Continued)

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
DataRedundancyMin	uint16	minvalue(1) write(true)	DataRedundancyMin describes the minimum number of complete copies of data to be maintained. Examples would be RAID 5 where 1 copy is maintained and RAID 1 where 2 or more copies are maintained. Possible values are 1 to n.
PackageRedundancyMax	uint16	write(true)	PackageRedundancyMax describes the maximum number of spindles to be used. Package redundancy describes how many disk spindles can fail without data loss including, at most, one spare. Examples would be RAID5 with a Package Redundancy of 1, RAID6 with 2. Possible values are 0 to n.
PackageRedundancyMin	uint16	write(true)	PackageRedundancyMin describes the minimum number of spindles to be used. Package redundancy describes how many disk spindles can fail without data loss including, at most, one spare. Examples would be RAID5 with a Package Redundancy of 1, RAID6 with 2. Possible values are 0 to n.
DeltaReservationGoal	uint16	minvalue(1), maxvalue(100)	Delta reservation is a number between 0 (0 %) and a 100 (100 %) that specifies how much space should reserved in a replica for caching changes. For a complete copy this would be 100 %, but it can be lower in some implementations. Use 0 if copy service Subprofile is not supported.

7.3.3.8.13 Optional Subprofiles and Profiles

Table 68 - Copy Services Optional Subprofiles and Profiles

Optional Subprofiles & Profiles	Notes
Job Control	This subprofile is used to support copy services that run for a long time. The extrinsic methods support the “ConcreteJob” output. If job control is not supported this output is null.

7.3.3.9 Job Control Subprofile

7.3.3.9.1 Description

In some profiles, some or all of the methods described may take some time to execute (longer than a HTTP time-out). In this case, a mechanism is needed to allow asynchronous execution of the method as a 'job'.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

This subprofile defines the constructs and behavior for job control for SNIA profiles that make use of the subprofile.

NOTE The subprofile describes a specific use of the constructs and properties involved. The actual CIM capability may be more, but this standard clearly states what clients may depend on in SNIA profiles that implement the Job Control subprofile.

7.3.3.9.2 Standard dependencies

The Job Control subprofile is based on the following standards:

Table 69 - Job Control Services Standard Dependencies

Standard	Version	Organization
CIM Specification	2.2	DMTF
CIM Operations over HTTP	1.1	DMTF
CIM Schema	2.7	DMTF

7.3.3.9.3 Profile dependencies

The Job Control subprofile does not require any other Profiles.

7.3.3.9.4 CIM Server Requirements

7.3.3.9.4.1 Functional Profiles

Table 70 - Required Functional Profiles

Profile Required	Functional Group	Dependency
YES	Basic Read	None
YES	Basic Write	Basic Read
YES	Instance Manipulation	Basic Write
NO	Schema Manipulation	Instance Manipulation
YES	Association Traversal	Basic Read
NO	Query Execution	Basic Read
NO	Qualifier Declaration	Schema Manipulation
YES	Indication	None

7.3.3.9.4.2 Extrinsic methods

The CIM Server shall support extrinsic methods for the Job Control subprofile.

7.3.3.9.4.3 Discovery

The Job Control subprofile, as currently defined, is not an advertised subprofile.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.3.9.5 Instance diagrams

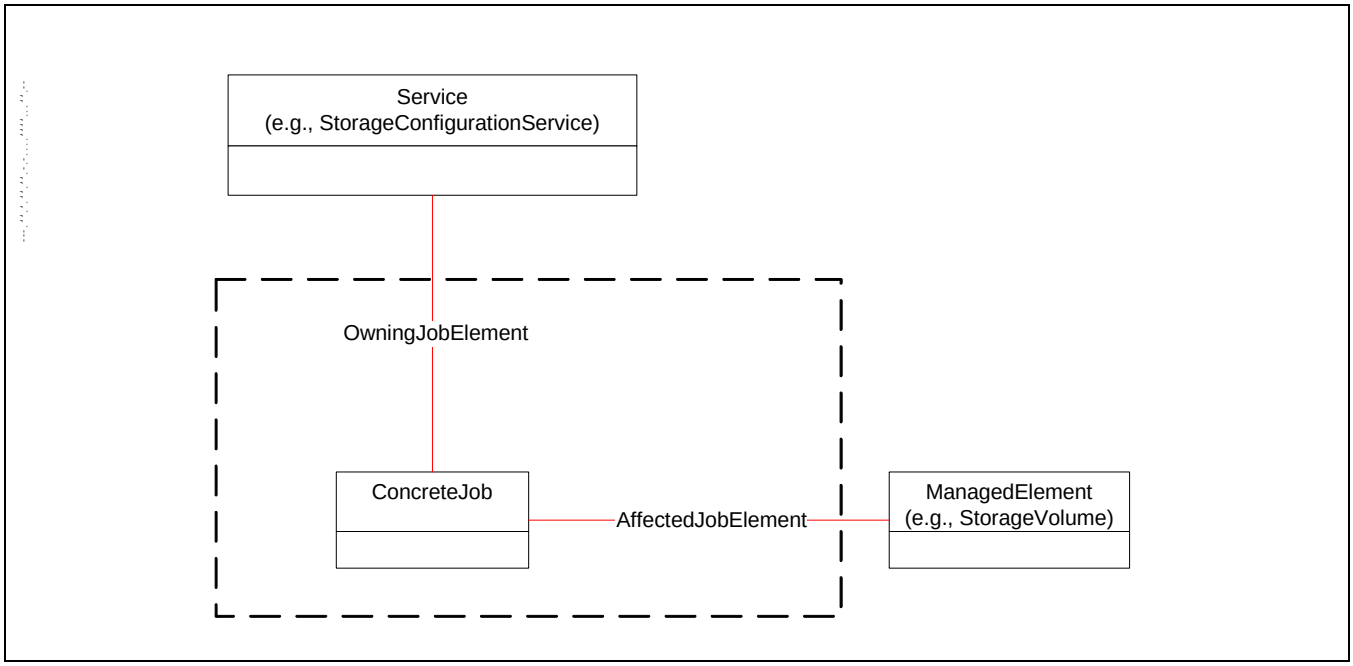


Figure 28 - Job Control Subprofile Model

When the job control subprofile is implemented and a client executes a method with the “ConcreteJob” reference as an output, a reference to an instance of ConcreteJob is returned and the return value for the method is set to “Method parameters checked - job started”.

The ConcreteJob instance allows the progress of the method to be checked and instance Indications can be used to subscribe for Job completion.

The associations OwningJobElement and AffectedJobElement are used to indicate the service that 'owns' the job and the element being affected by the job. The element linked by AffectedJobElement may change through the execution of the job. For instance, for creation of a StorageVolume it may start by pointing to a source pool and then change to the newly created instance of StorageVolume as the method executes.

7.3.3.9.6 Durable names and correlatable IDs

There are no durable names or correlatable IDs for Job Control.

7.3.3.9.7 Methods

Jobs are created as a result of executing methods of the parent profile. The Job Control constructs can be read using intrinsic methods. There are no basic write intrinsic methods supported in the Job Control subprofile.

7.3.3.9.8 Client considerations

If the operation will take a while (longer than an HTTP timeout), a handle to a newly minted ConcreteJob is returned. This allows the job to continue in the background. Note the following points.

- The job may be queued. You may have multiple outstanding jobs against a pool for instance. The job status shows this.
- The job is weak to the Service (shown via OwningJobElement) and is also linked to the object being modified/created via AffectedJobElement. For example, a job to create a StorageVolume may start off pointing to a Pool until the Volume is instantiated at which point the association would change to the StorageVolume.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

- These jobs do not have to get instantiated! If things happen quickly, a null can be returned as a handle.

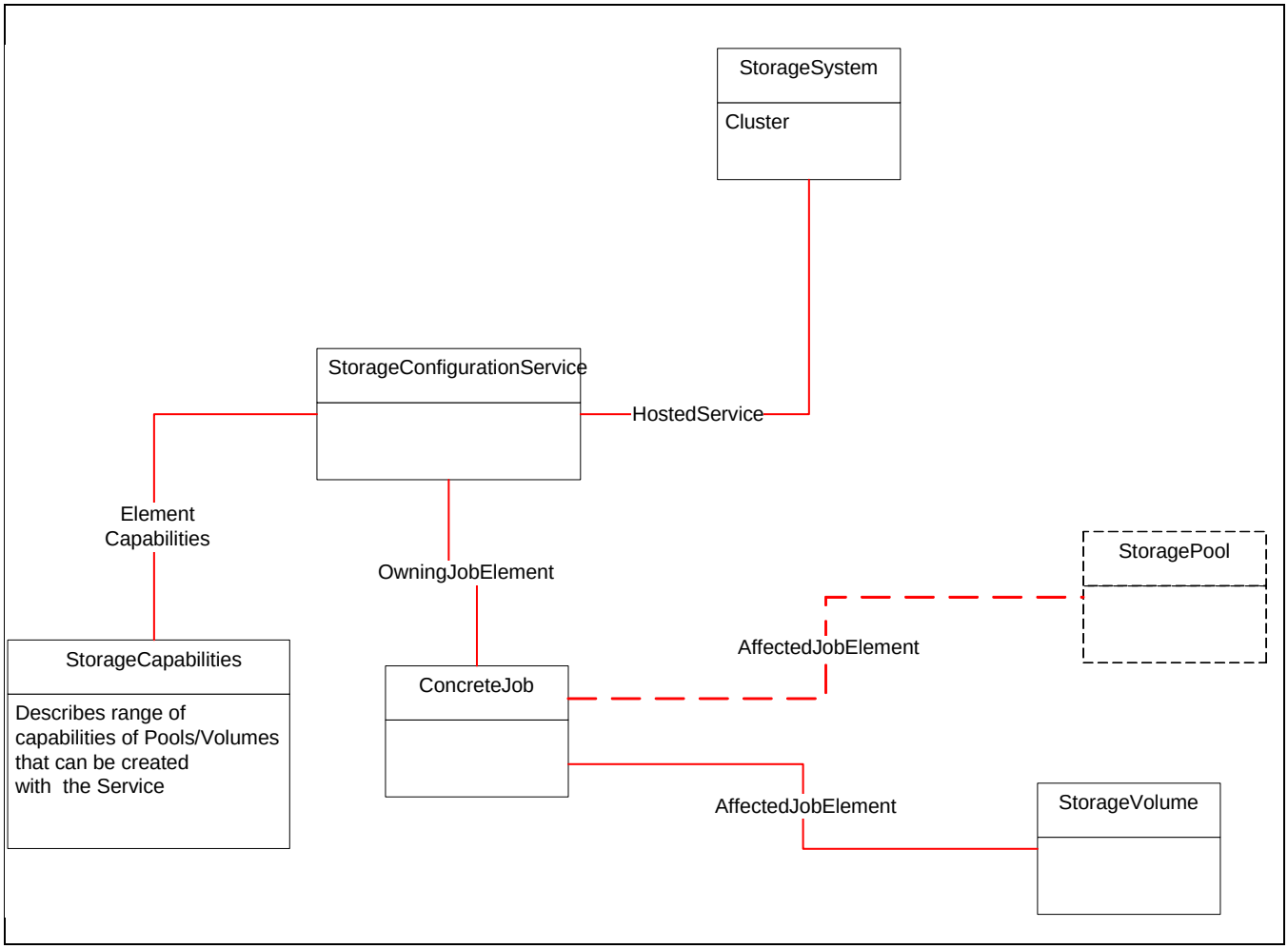


Figure 29 - Storage Configuration

7.3.3.9.9 Recipes

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

7.3.3.9.10 Instrumentation requirements

7.3.3.9.10.1 OperationalStatus for Jobs

The operationalStatus property is used to communicate that status of the job that is created. As such, it is critical that implementations are consistent in how this property is set. The values that shall be supported consistently are:

- “OK” - combined with “Completed” to indicate that the job completed with no error.
- “Error” - combined with “Completed” to indicate that the job did not complete normally and that an error occurred.
- “Stopped” implies a clean and orderly stop.
- “Completed” indicates the Job has completed its operation. This value should be combined with either “OK” or “Error, so that a client can tell if the complete operation passed (completed with OK) and failure (completed with Error).

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.3.3.9.11 Required CIM Elements

Table 71 - Subprofile Required Classes, Associations, Methods and Indications

Subprofile Class & Associations	Notes
AffectedJobElement	
ConcreteJob	
OwningJobElement	
Subprofile Class and Associated Indications	
Changes in OperationalStatus of ConcreteJob	SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_ConcreteJob AND SourceInstance.OperationalStatus = 17 AND SourceInstance.OperationalStatus = 2
Progress toward completion of a ConcreteJob	SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_ConcreteJob AND SourceInstance.OperationalStatus = 17 AND SourceInstance.OperationalStatus = 6 An implementation increment may be 100 %.

7.3.3.9.12 Required Properties for CIM Elements

7.3.3.9.12.1 AffectedJobElement

AffectedJobElement represents an association between a Job and the ManagedElement(s) that may be affected by its execution. It may not be feasible for the Job to describe all of the affected elements. The main purpose of this association is to provide information when a Job requires exclusive use of the 'affected' ManagedElment(s) or when describing that side effects may result.

AffectedJobElement is not subclassed from anything.

Table 72 - AffectedJobElement Required Properties

Class Properties	Type	Qualifier/ Parameter	Notes
AffectedElement	ref	key	The ManagedElement affected by the execution of the Job.
AffectingElement	ref	key	The Job that is affecting the ManagedElement.

7.3.3.9.12.2 ConcreteJob

A concrete version of Job. This class represents a generic and instantiatable unit of work, such as a batch or a print job.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

ConcreteJob is subclassed from Job

Table 73 - Required Properties for ConcreteJob

Class Properties	Type	Qualifier/ Parameter	Notes
OperationalStatus[ ]	uint16		Indicates the current status(es) of the element. Various health and operational statuses are defined.
StatusDescriptions[ ]	string		A string describing the status - used when the OperationalStatus property is set to 1 (\"Other\").
JobStatus	string		A free form string representing the Job's status. The primary status is reflected in the inherited OperationalStatus property. JobStatus provides additional, implementation-specific details.
ElapsedTime	datetime		The time interval that the Job has been executing or the total execution time if the Job is complete. This property is OPTIONAL.
PercentComplete	uint16		The percentage of the job that has completed at the time that this value is requested.
DeleteOnCompletion	boolean	write(true)	Indicates whether or not the job should be automatically deleted upon completion. If this property is set to false and the job completes, then the extrinsic method DeleteInstance shall be used to delete the job versus updating this property.
ErrorCode	uint16		A vendor specific error code. This is set to zero if the job completed without error.
ErrorDescription	string		A free form string containing the vendor error description.
InstanceID	string	key	InstanceID opaquely identifies a unique instance of ConcreteJob. The InstanceID shall be unique within a namespace.
Name	string	override, required	The user friendly name for this instance of Job. In addition, the user friendly name can be used as a property for a search or query. (Note that the name does not have to be unique within a namespace.)

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.3.9.12.3 OwningJobElement Properties.

OwningJobElement represents an association between a Job and the ManagedElement responsible for the creation of the Job. This association may not be possible, given that the execution of jobs can move between systems and that the lifecycle of the creating entity may not persist for the total duration of the job. However, this can be very useful information when available.

OwningJobElement is not subclassed from anything.

Table 74 - Required Properties for OwningJobElement

Class Properties	Type	Qualifier/ Parameter	Notes
OwningElement	ref	max(1), key	The ManagedElement responsible for the creation of the Job. (e.g., StorageConfigurationService).
OwnedElement	ref	key	The Job created by the ManagedElement.

7.3.3.9.13 Optional Subprofiles

Table 75 - Optional Profiles or Subprofiles

Name	Notes
None	

7.3.3.10 Pool Manipulation Capabilities and Settings Subprofile

7.3.3.10.1 Description

Storage Pools

A StoragePool is an abstract notion of a blob of consumable storage space. A pool has certain ‘StorageCapabilities’, which indicate the range of 'Quality of Service' requirements that can be applied to objects created from the pool. In this top-level profile, StorageCapabilities are informational only. Refer to 7.3.3.10, “Pool Manipulation Capabilities and Settings Subprofile” for details on the use of these objects.

Storage pools are scoped relative to the ComputerSystem (indicated by the HostedStoragePool association). Objects created from a pool have the same scope.

Child objects (e.g., StorageVolumes or StoragePools) created from a StoragePool are linked back to the parent pool using an AllocatedFromStoragePool association.

There are two properties on StoragePool that describe the size of the ‘underlying’ storage. TotalManagedSpace describes the total raw storage in the pool and RemainingManagedSpace describes the storage currently remaining in the pool. RemainingManagedSpace plus AllocatedFromStorage-Pool.SpaceConsumed from all of the StorageVolumes allocated from the pool shall equal TotalManagedSpace.

Primordial Pool

The Primordial Pool is a type of StoragePool. Raw storage capacity, unformatted or unprepared capacity, is drawn from the Primordial StoragePool to create concrete StoragePools. The Primordial StoragePool aggregates storage capacity that has not been assigned to a concrete StoragePool. StorageVolumes are allocated from concrete StoragePools.

At least one shall always exist on the array to represent the unallocated storage on the storage device. The sum of TotalManagedStorage attributes for all Primordial StoragePools shall be equal to the total size of the raw storage of the storage system. The Primordial property shall be true for Primordial Pools.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Primordial Pool can be used to determine the amount of raw space left on the array, that is not already assigned to a concrete StoragePool.

Storage Volumes

Storage Volumes are configured pieces of storage that shall be exposed from a system through an external interface. In the class hierarchy they are a subclass of a StorageExtent. In SCSI terms, they are Logical Units.

StoragePools are a required part of modelling disk storage systems (the Array, Out-of-band Virtualization and In-band Virtualization Profiles). However, user manipulation of StoragePools is optional and may not be supported by any given disk storage system. The Pool Manipulation Capabilities and Settings subprofile defines the support required if the storage system exposes functions for creating and modifying storage pools.

The StorageConfigurationService, in conjunction with the abstract concept of a storage pool, allows generic clients to configure pools of storage within storage arrays without having to have specific knowledge about the array configuration. The new service has the following methods.

- CreateOrModifyStoragePool: Create a pool of storage with some set of Capabilities defined by the input StorageSetting. The source of the storage can be other pool(s) or storage extents. Alternatively, an existing pool can be modified.
- DeleteStoragePool: Delete a storage pool and return the freed up storage to the underlying entities.

In addition, there is a capability to create settings for use in pool creation using the following method (part of the StorageCapabilities class).

- CreateSetting: Creates a setting that is consistent with the StorageCapabilities and may be modified before use in creating a StoragePool.

7.3.3.10.2 Standard dependencies

The Pool Manipulation Capabilities and Settings subprofile is defined using the CIM Schema 2.8 preliminary. As such it can be used in profiles at 2.8 and later. It does not require that Profiles be on a later schema. It operates within profiles that are at the CIM schema 2.8 final or later. The subprofile operates correctly with CIM Specification 2.2 (or later) and CIM Operations over HTTP 1.1 (or later).

The Pool Manipulation Capabilities and Settings subprofile is based on the following standards:

Table 76 - Pool Manipulation Capabilities and Settings Standard Dependencies

Standard	Version	Organization
CIM Specification	2.2	DMTF
CIM Operations over HTTP	1.1	DMTF
CIM Schema	2.8 Preliminary	DMTF

7.3.3.10.3 Profile dependencies

The Pool Manipulation Capabilities and Settings subprofile introduces no Profile dependencies.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.3.10.4 CIM Server requirements

7.3.3.10.4.1 Functional Profiles

Table 77 - Required Functional Profiles

Profile Required	Functional Group	Dependency
YES	Basic Read	None
YES	Basic Write	Basic Read
YES	Instance Manipulation	Basic Write
NO	Schema Manipulation	Instance Manipulation
YES	Association Traversal	Basic Read
NO	Query Execution	Basic Read
NO	Qualifier Declaration	Schema Manipulation
YES	Indication	None

7.3.3.10.4.2 Extrinsic methods

The CIM Server shall support extrinsic methods for the Pool Manipulation Capabilities and Settings subprofile.

7.3.3.10.4.3 Discovery

The Pool Manipulation Capabilities and Settings subprofile is not advertised.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.3.10.5 Instance diagrams

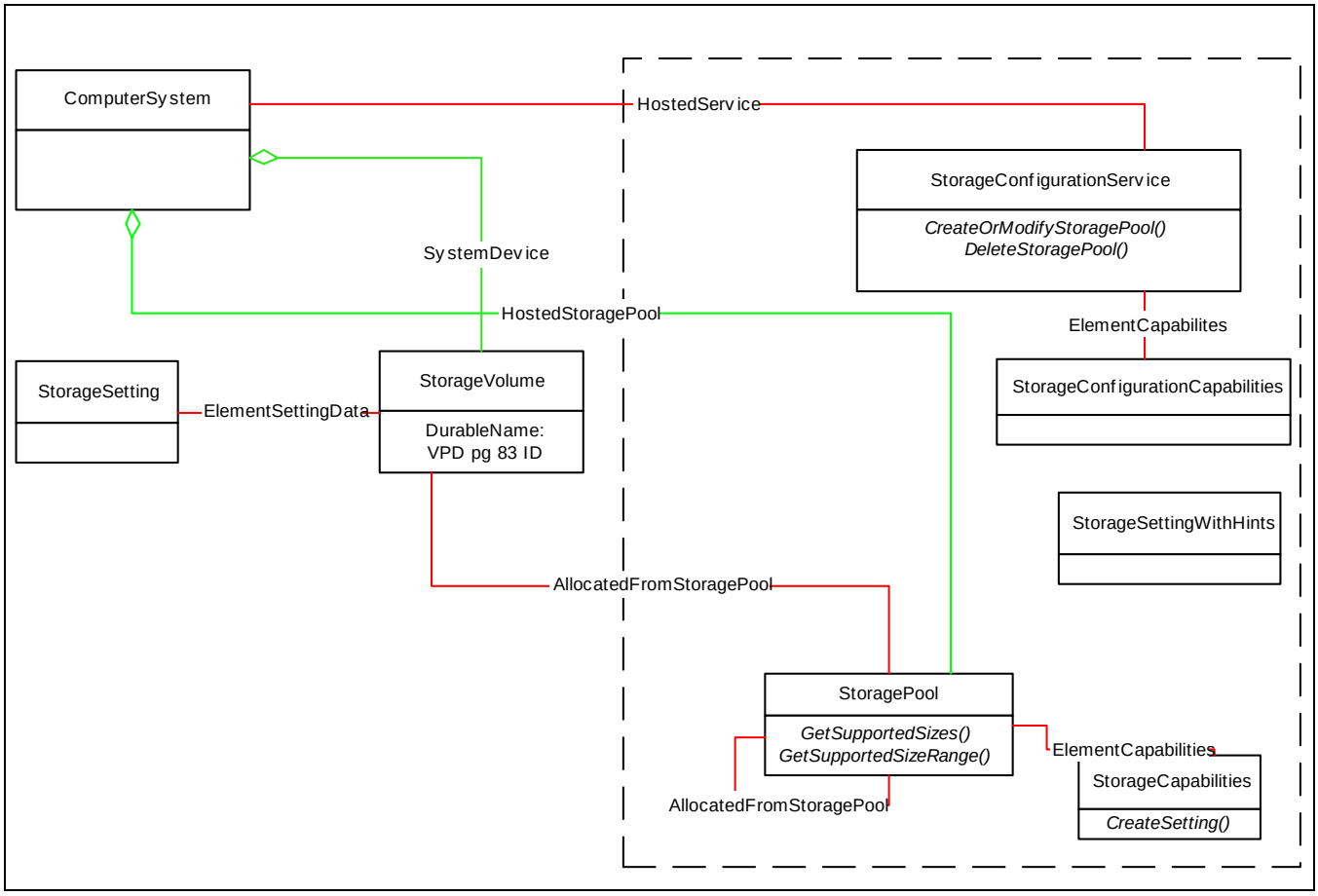


Figure 30 - Pool Manipulation Instance Diagram

7.3.3.10.6 Durable names and correlatable IDs

The Pool Manipulation Capabilities and Settings subprofile does not add any durable names or correlatable IDs to the profiles (or subprofiles) in which it is used.

7.3.3.10.7 Methods

7.3.3.10.7.1 Overview

The Pool Manipulation Capabilities and Settings introduces a number of write intrinsic and extrinsic methods.

7.3.3.10.7.2 StorageSetting methods

7.3.3.10.7.2.1 Extrinsic methods on Storage Setting

CreateSetting is a method in StorageCapabilities and is invoked in the context of a specific StorageCapabilities instance.

Uint32 CreateSetting(

[In] unint16 SettingType

[Out] CIM_StorageSetting REF NewSetting)

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

This method on the StorageCapabilities class is used to create a StorageSetting using the StorageCapabilities as a template. The purpose of this method is to create a StorageSetting that is associated directly with the StorageCapabilities on which this method is invoked and has properties set in line with those StorageCapabilities. The contract defined by the StorageCapabilities shall constrain the StorageSetting used as the Goal.

The StorageCapabilities associated with the StoragePool defines which type of storage can be allocated. The client shall determine which subset of the parent StoragePool capabilities shall be used, albeit a Primordial StoragePool or a concrete StoragePool. The StorageSetting provided to the StoragePool creation method defines which measure of capabilities are desired for the following storage allocation. First, the client retrieves a StorageSetting or creates and optionally modifies an existing StorageSetting. If no satisfactory StorageSetting exists, then the client uses this method to create a StorageSetting.

The client has the option to have a StorageSetting generated with the default capabilities from the StorageCapabilities. If a '2' is passed for the Setting Type parameter, the Max, Goal and Min setting attributes are set to the default values of the parent StorageCapabilities. Otherwise, the new StorageSetting attributes are set to the related attributes of the parent StorageCapabilities, for example, Min to Min and Max to Max. If the StorageSetting requested already exists, associated to the StorageCapabilities, then the method returns this existing StorageSetting. This type of StorageSetting, newly created or already existing, is associated to the StorageCapabilities via the GeneratedStorageSetting association.

Only a StorageSetting created in this manner may be modified or deleted by the client. The client uses the NewSetting parameter to set the new StorageSetting to the values desired (using ModifyInstance or SetProperties intrinsic methods). The StorageSetting can not be used to create storage that is more capable than the parent StorageCapabilities. For example, the set instance operation fails when the setting has a Max value greater (or a Min value less) than the parent StorageCapabilities. If the storage device supports hints, then the new StorageSetting contains the default hint values for the parent StorageCapabilities. The client can use these values as a starting point for hint modification (using intrinsic methods). StorageSetting instances associated with StorageVolume shall not be modified or deleted directly. If a StoragePool is created, then the client shall use the StorageConfigurationService to modify or delete the instance.

If this type of StorageSetting is used as the Goal for the creation of a StoragePool, the Goal StorageSetting is removed. A new StorageCapabilities instance, associated with the new StoragePool, describes the StoragePool.

7.3.3.10.7.2.2 Intrinsic methods on StorageSetting

In addition to these extrinsic methods, the following Intrinsic write methods are supported on StorageSetting:

- DeleteInstance;
- ModifyInstance.

7.3.3.10.7.3 StorageConfigurationService Methods:

CreateOrModifyStoragePool

```
Uint32 CreateOrModifyStoragePool(
                            [Out] CIM_ConcreteJob ref Job,
                            [In] CIM_StorageSetting ref Goal,
                            [In,out] Uint64 Size,
                            [In] string InPool[],
                            [In] string Extent[],
                            [In, out] CIM_StoragePool ref Pool,
                            [In] string ElementName
                            );
```

This method is used to create a Pool from either a source pool or a list of storage extents. Any required associations (such as HostedStoragePool) are created in addition to the instance of Storage Pool. The parameters are as follows:

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

- Job: see 7.3.3.9, “Job Control Subprofile”
- Goal: This is the Service Level that the Pool is expected to provide. This may be a null value in which case a default setting is used.
- Size: As an input this is the desired size of the pool. If it is not possible to create a pool of the desired size, a return code of “Size not supported” is returned with size set to the nearest supported size.
- InPool[ ]: This is an array of strings containing Object references (see 4.11.5 of the CIM Specification for format) to source Storage Pools.
- Extent[ ]: This is an array of strings containing Object references (see 4.11.5 of the CIM Specification for format) to source Storage Extents. Note that either an array of source pools or an array of source extents should be defined, but not both.

DeleteStoragePool

```
Uint32 DeleteStoragePool(
                          [Out] CIM_ConcreteJob ref Job,
                          [In] CIM_StoragePool ref Pool
                          );
```

This method is provided to allow a client to delete a previously created storage pool. All associations to the deleted StoragePool are also removed as part of the action. In addition, the TotalManagedStorage and the RemainingManagedStorage of the associated Primordial Storage Pool will change accordingly.

NOTE This method will be denied (“Failed”) if there are any AllocatedFromStoragePool associations where the deleted pool is the Dependent.

Return Values

Each method has this set of defined return codes:

```
ValueMap {“0”, “1”, “2”, “3”, “4”, “5”, “..”, “0x1000”,”0x1001”,
“0x1002..0x7777”, “0x8000..”},
Values {“Job completed with no error”, “Not Supported”, “Unknown”,
“Timeout”, “Failed”, “Invalid Parameter”, “DMTF Reserved”,
“Method parameters checked - job started”,
“Size not supported”, “Method Reserved”, “Vendor Specific”}]
```

If the method completes immediately with no errors (and with no asynchronous execution required), “Job completed with no error” is returned.

If the method parameters have been checked and the method is being executed asynchronously, “Method parameters checked - job started” is returned.

If, for a Create/Modify method, the requested size is not supported then “Size not supported” is returned and the Size parameter is set to the nearest supported size.

If one of the method parameters is incorrect (for instance invalid object paths), then “Invalid Parameter” is returned.

“Timeout” or “Failed” may be returned if the provider has problems accessing the hardware or for other implementation specific reasons.

A vendor may choose to extend the Value map to express vendor specific error codes not catered for by the standard errors.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.3.10.7.4 StoragePool methods

GetSupportedSizes

```
Uint32 GetSupportedSizes(
                            [In] Uint16 ElementType,
                            [In] CIM_StorageSetting ref Goal,
                            [Out] Uint64 Sizes[]
```

This method is used to determine the possible sizes of child elements, for example StoragePool and StorageVolume, that can be created or modified using capacity from the StoragePool. The method is used for storage system where only discrete sizes are possible. One of the reported sizes can be used directly along with the Goal in the creation of a StoragePool or StorageVolume The sizes reported may not differ from each other by a fixed size.

GetSupportedSizeRanges

```
Uint32 GetSupportedSizeRanges(
                            [In] Uint16 ElementType,
                            [In] CIM_StorageSetting ref Goal,
                            [Out] Uint64 MinimumVolumeSize,
                            [Out] Uint64 MaximumVolumeSize,
                            [Out] Uint64 VolumeSizeDivisor
```

This method is used to determine the possible sizes of child element, for example StoragePool and StorageVolume, that can be created or modified using capacity drawn from the StoragePool. The out parameters tell the minimum element size, maximum element size and possible sizes in that range. This method can prove useful when the number of possible sizes is so voluminous that reporting each discrete size would be impractical.

Both or either method may be supported by a storage subsystem, either as a decision made at implementation time or it varies depending on the state of the StoragePool. For example, when a StoragePool is first created that allows for possible sizes to be in 1 024 byte blocks, then the GetSupportedSizeRanges method would be better to report the possible sizes. This example StoragePool does not relocate blocks to avoid fragmentation of the capacity. As StorageVolume is drawn from and returned to the StoragePool, the capacity becomes fragmented. In this case, the GetSupportedSizes method is better in reporting the non-continuous regions of capacity that may be used for element creation. Another example, there are some storage systems that can only allocate StorageVolume in whole disks and these disks need not be of a uniform size. In this case, the storage system would only support the GetSupportedSizes method.

Return Values

Each method has this set of return codes:

```
ValueMap {"0", “1”, “2”},
Values {“Method completed OK”, “Method not supported”,
 “Use <the other method name> instead”}]
```

If the above methods did not complete successfully, then either the method is not supported or it is suggested to use the other method instead. The GetSupportSizes method can notify the SMI-S client that it should use the GetSupportSizeRanges instead, or the GetSupportedSizeRanges method can notify the SMI-S client that it should use the GetSupportedSizes method instead.

7.3.3.10.8 Client considerations

7.3.3.10.8.1 Storage Pools and Storage Capabilities

Capacity Management

The capacity characteristics of many storage system vary greatly in cost and performance. Additionally, the capacity may need to be partitioned by these and other factors. StoragePool provide a means to aggregate

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

this storage by characteristics determined by the storage administrator or determined at the factory when the storage system is assembled.

A Storage Pool is an aggregation of storage suitable for configuration and allocation or “provisioning”. However, it may have been preformatted into a form (such as a RAID group) that makes volume creation easier.

StoragePools can be drawn from a StoragePool (the result of which is indicated with the AllocatedFromStoragePool association).

A StoragePool has a set of capabilities held in the StorageCapabilities class that reflect the configuration parameters that are possible for an element created from this pool. The StorageCapabilities define, in terms common across all storage system implementation, which characteristics an administrator can expect from the storage capacity. These capabilities are expressed in ranges. The storage implementation has the choice to delineate the capabilities and define the ranges of these capabilities, as appropriate. Some implementation may require several narrowly defined capabilities while others may be more flexible.

The capabilities expressed by the storage system can change over time.

The number of primordial storage pools can change over time as well.

These storage capabilities are given the scope of the storage system when they are associated by the StorageConfiguratonService or the scope of a single StoragePool. The capabilities expressed at the service scope is equal to the union of the union of all Primordial StoragePools capabilities. The capabilities can also be given the scope of a concrete StoragePool.

The storage administrator has the choice of any capability expressed by the storage system. The administrator should use this opportunity to partition the capacity. As soon as storage elements are drawn from the StoragePool, the administrator can be assured that the elements produced will have the capabilities previously defined.

The model allows for automation of the allocation process. An automaton can use the capabilities’ properties to search across subsystems for storage providing desired capabilities and having done so, create StoragePools and/or storage elements as necessary. Inventories may be made of the capacity by capabilities.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

The model also provides a means by which some common characteristics of all available storage systems can be inventoried and managed. Note that the storage system will differ in other significant ways and these characters can also be the basis for capacity pooling decisions.

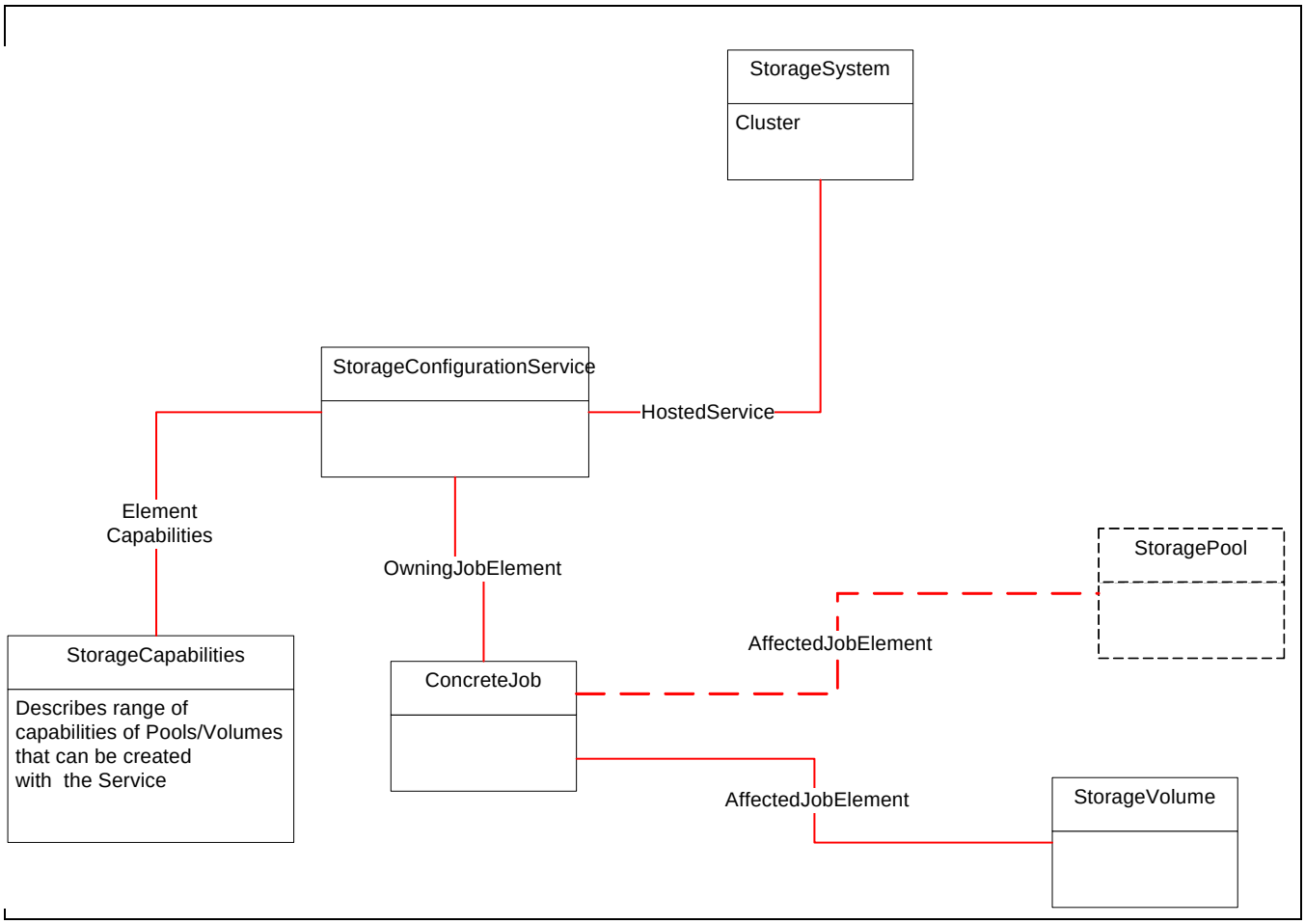


Figure 31 - Storage Configuration

The definition of storage capabilities in this way intentionally avoids vendor-specific details of volume configuration such as RAID types. Although RAID types imply performance and availability levels, these levels cannot be easily compared between vendor implementations - particularly in comparison with reliability of non-RAID storage (i.e., certain virtualization appliances). Furthermore, there are capabilities of reliability and availability other than data redundancy. The StorageSetting class is provided by clients to describe the desired configuration of the allocated storage. In general, the types of parameters exposed and controlled via the StorageCapabilities/StorageSetting classes are:

- NSPOF (No Single Point of Failure). Indicates whether the pool can support storage configured with No Single Points of Failure within the storage system. This does not include the path from the system to the host.
- Data Redundancy. This describes the number of complete copies of data maintained. Examples would be RAID 5 where 1 copy is maintained and mirroring where 2 or more copies are maintained.
- Package Redundancy. This describes how many physical components (packages), like disk spindles, can fail without data loss (including a spare, but not more than a single global spare). Examples would be RAID5 with a Package Redundancy of 1, RAID6 with 2, RAID6 with 2 global (to the system) spares would be 3.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

- Delta Reservation. This is a number between 1 (1 %) and a 100 (100 %) that specifies how much space should be reserved in a replica for caching changes. For a complete copy this would be 100 %, but it can be lower in some implementations.

An example of what the Package Redundancy and Data Redundancy means in terms of RAID levels is defined in Table 78.

7.3.3.10.8.1.1 Example mapping of RAID levels to Data Redundancy, Package Redundancy

Table 78 - Example RAID Mapping Table

RAID Level	Package Redundancy	Data Redundancy
0 (Striping)	0	1
1	1	2
3 or 4	1	1
5	1	1
6	2	1
10	1	2
15	2	2
50	1	1
51	2	2

The above example was produced using generally available definitions of RAID levels. It is the nature of RAID technology that even though the RAID Level is named the same, the storage service provided could differ depending on the storage device implementations. Expressing the storage service level provided in end-user terms relieves the SMI-S Client and end-user from having to know what RAID Levels means for a particular implementation and instead defines the storage provided in service level terms.

If a single storage device implements RAID levels that have the same package redundancy and data redundancy, the implementer is encouraged to have the SMI-S Client differentiate via StorageSettingsWithHints. Additionally, the SMI-S Provider author can predefine StorageCapabilities that match exactly with best practice RAID Levels, including differentiation with StorageSettingWithHints when StorageVolume exists. In this case, the ElementName property is used to correlate between the capability and device documentation. Alternatively, it may sense for the capability to be expressed in broader ranges for more flexible storage systems.

Storage Pool Manipulation

The StorageConfigurationService class contains methods to allow creation, modification and deletion of a StoragePool. The capabilities of a StorageConfigurationService or StoragePool to provide storage are indicated using the StorageCapabilities class. This class allows the Service or Pool to advertise its capabilities (using implementation independent attributes) and a client to set the attributes it desires. The concept of ‘hints’ is also included that allows a client to provide clues to the system as to how it expects to use the storage for optimization purposes. For example, if the array supports the creation of Pools that can tolerate the loss of two disks, then the ‘package redundancy’ attribute includes 2 in its range of supported values. The client would create an instance of StorageSetting, set ‘package redundancy’ to 2, and pass a reference to the class to the StorageConfigurationService.CreateOrModifyStoragePool.

Pool creation works as follows.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

a) Figure 32 shows the initial state of the array - a single ‘primordial pool that advertises its capabilities. One can make use of the GetSupportedSizes() and GetSupportedSizeRanges() methods to determine what sizes of pools can be created from the primordial pool. One needs to check the StorageConfigurationCapabilities to ensure that creation of StoragePools is supported or not.

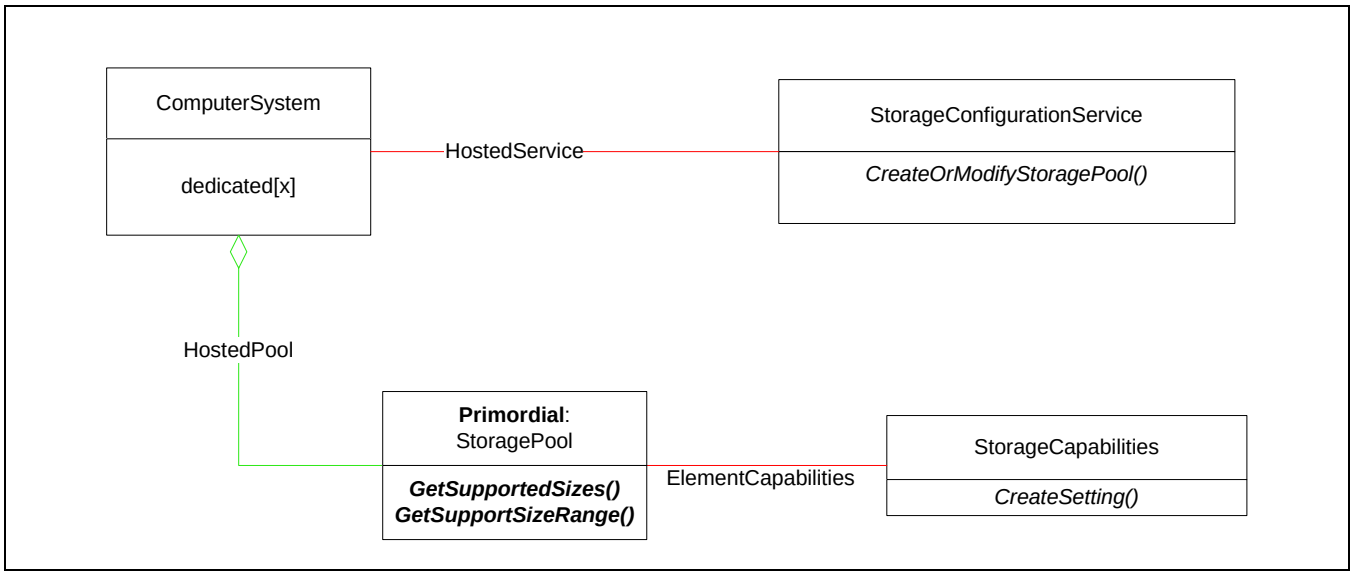


Figure 32 - Pool Creation - Initial State

b) Next, (Figure 33) the client uses the CreateSetting method on the StorageCapabilities instance to create an instance of a StorageSetting. This Setting object can be altered as desired. If the array supports StorageSettingWithHints, an instance of this subclass is created rather than the StorageSetting superclass.

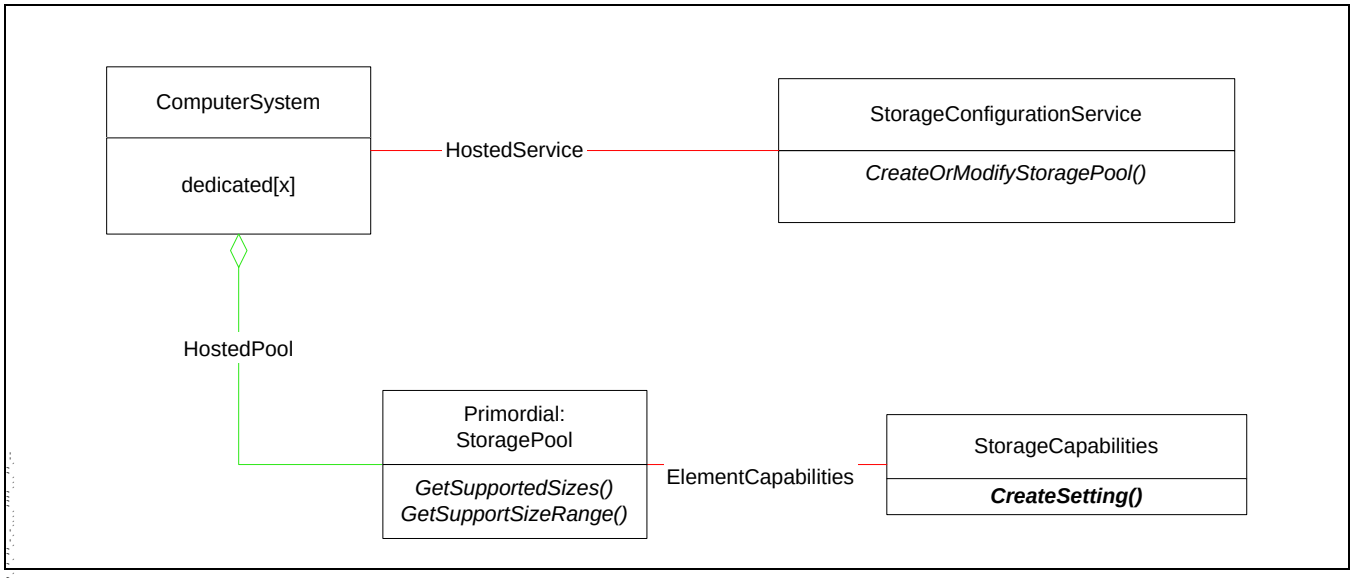


Figure 33 - Pool Creation - Step 2

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

c) As soon as this Setting as been altered as required, it is passed as an argument to the CreateOrModifyStoragePool method in the StorageConfigurationService (as shown in Figure 34).

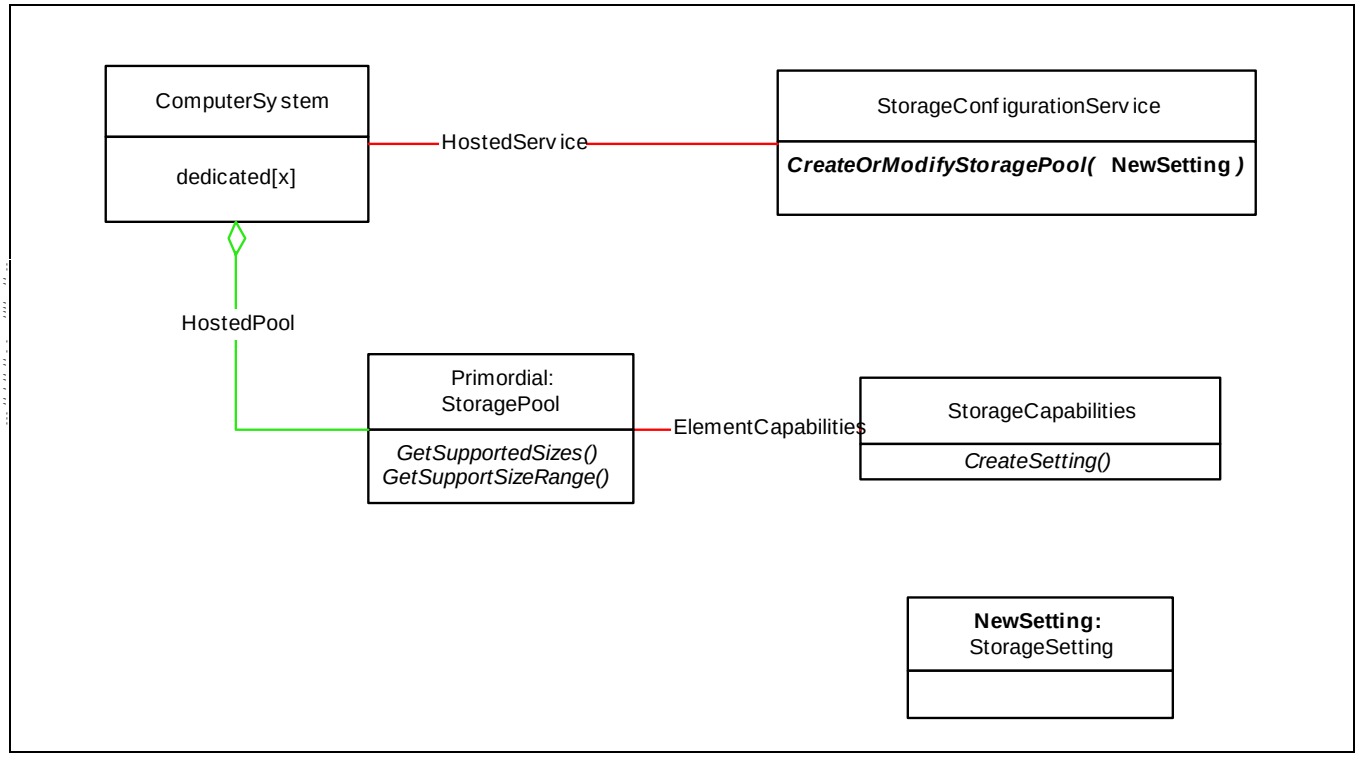


Figure 34 - Pool Creation - Step 3

d) The pool is then created. The 'temporary' StorageSetting is replaced with an equivalent StorageCapabilities object linked to the new pool with ElementCapabilities (as shown in Figure 35)

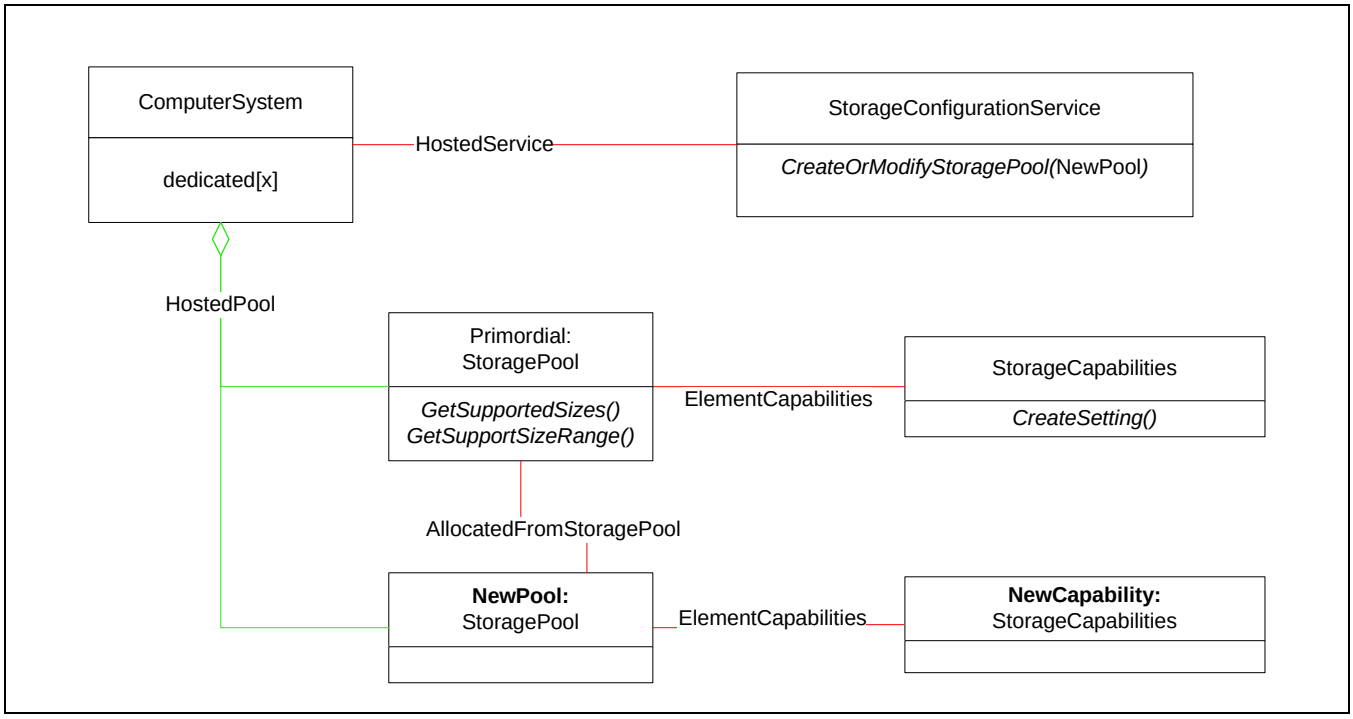


Figure 35 - Pool Creation - Step 4

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.3.3.10.8.2 The CreateOrModifyStoragePool method and the Primordial Pool

The InPool array for the StorageConfigurationService.CreateOrModifyStoragePool() method shall always contain at least a string reference to the Primordial Pool. The InPool is therefore a required parameter. If the Primordial Pool is passed as the only element in the InPool parameter to the CreateOrModifyStoragePool, then the size requested is prepared according to the specification of the Goal parameter and drawn from the Primordial Pool. If the Primordial Pool is one of many Pool passed to the method, then the Size is drawn from the Pools and/or the Extents that match the Goal. The Primordial Pool matches all Goals possible for the device. If another Pool matches the Goal other than the Primordial Pool, then the Size requested is drawn from the other Pool. Any remaining Size not satisfied from the other Pool, is drawn from the Primordial Pool.

As capacity is drawn from the Primordial Pool or any Pool, then the size of the Primordial Pool shrinks until such time as all allocated or raw storage is consumed by all children Pools of the Primordial Pool.

7.3.3.10.9 Recipes

See Create Storage Pool and Storage Volume on array (7.3.3.11.9.1) for an example recipe.

7.3.3.10.10 Instrumentation Requirements

See details in related profile section.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.3.10.11 Required CIM Elements

Table 79 - Required CIM Elements

Profile Classes & Associations	Notes
AllocatedFromStoragePool (7.3.6.1.12.1)	AllocationFromStoragePool as defined in the Array Profile.
StoragePool (7.3.6.1.12.11)	StoragePool as defined in the Array profile.
ElementCapabilities (7.3.3.10.12.1)	Associates StorageConfigurationCapabilities with StorageConfigurationService.
StorageConfigurationCapabilities (7.3.3.10.12.3)	
ElementSettingData (7.3.3.10.12.5)	
StorageSetting (7.3.3.10.12.6)	
StorageSettingWithHints (7.3.3.10.12.7)	
StorageConfigurationService (7.3.3.10.12.2)	
StorageCapabilities (7.3.3.10.12.4)	
HostedService (7.3.3.10.12.8)	
Packages	
None.	
Methods	
CreateOrModifyStoragePool()	
DeleteStoragePool()	
CreateSetting()	
GetSupportedSizes()	
GetSupportSizeRanges()	
SubProfile Indications	
Creation/Deletion of StoragePool	SELECT * from CIM_InstCreation where SourceInstance ISA CIM_StoragePool SELECT * from CIM_InstDeletion where SourceInstance ISA CIM_StoragePool

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.3.10.12 Required Properties for CIM Elements

7.3.3.10.12.1 ElementCapabilities

Table 80 - Required Properties for ElementCapabilities

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
ManagedElement	ref	key, min(1), max(1)	The managed element.
Capabilities	ref	key	The Capabilities object associated with the element.

7.3.3.10.12.2 StorageConfigurationService

Table 81 - Required Properties for StorageConfigurationService

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
ElementName	string		User Friendly name.
SystemCreationClassName	string	maxlen(256), key, propagated	The scoping System's CreationClassName.
SystemName	string	maxlen(256), key, propagated	The scoping System's Name.
CreationClassName	string	maxlen(256), key	The name of the concrete subclass.
Name	string	maxlen(256), key, override	
CreateOrModifyStoragePool()	uint32		Create (or modify) a StoragePool. A job may be created as well.
DeleteStoragePool ()	uint32		Start a job to delete a StoragePool.

7.3.3.10.12.3 StorageConfigurationCapabilities

Table 82 - Required Properties for StorageConfigurationCapabilities

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
ElementName	string	req	
InstanceID	string	key	InstanceID opaquely identifies a unique instance of Capabilities. The InstanceID shall be unique within a namespace.
SupportedStoragePoolFeatures[ ]	uint16		Lists which StorageConfigurationService methods are implemented.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table 82 - Required Properties for StorageConfigurationCapabilities

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
SupportedSynchronousActions[ ]	uint16		
SupportedAsynchronousActions[ ]	uint16		Lists which actions, invoked through StorageConfigurationService methods, may produce Concrete jobs
SupportedStorageElementTypes[ ]	uint16		
SupportedStorageElementFeatures[ ]	uint16		Lists which actions are supported through the invocation of StorageServiceService.CreateOrModifyElementFromStoragePool()

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.3.3.10.12.4 StorageCapabilities

Table 83 - Required Properties for StorageCapabilities

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
InstanceID	string	key	InstanceID opaquely identifies a unique instance of Capabilities. The InstanceID shall be unique within a namespace.
ElementName	string	override, required	The user friendly name for this instance of Capabilities. In addition, the user friendly name can be used as a index property for a search or query. (Note that ElementName does not have to be unique within a namespace.) If the capabilities are fixed, then this property should be used as a means for the client application to correlate between capabilities and device documentation.
ElementType	uint16		Enumeration indicating the type of instance to which this StorageCapabilities applies. Only '6', StorageConfigurationService and '5' StoragePool are valid.
NoSinglePointOfFailure	boolean		Indicates whether or not the associated instance supports no single point of failure. Values are: FALSE = does not support "no single point of failure" and TRUE = supports "no single point of failure".
NoSinglePointOfFailureDefault	boolean		Indicates the default value for the NoSinglePointOfFailure property.
DataRedundancyMax	uint16	minvalue(1)	DataRedundancyMax describes the maximum number of complete copies of data that can be maintained. Examples would be RAID 5 where 1 copy is maintained and RAID 1 where 2 or more copies are maintained. Possible values are 1 to *n*.
DataRedundancyMin	uint16	minvalue(1)	DataRedundancyMin describes the minimum number of complete copies of data that can be maintained. Examples would be RAID 5 where 1 copy is maintained and RAID 1 where 2 or more copies are maintained. Possible values are 1 to *n*.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table 83 - Required Properties for StorageCapabilities (Continued)

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
DataRedundancyDefault	uint16	minvalue(1)	DataRedundancyDefault describes the default number of complete copies of data that can be maintained. Examples would be RAID 5 where 1 copy is maintained and RAID 1 where 2 or more copies are maintained. Possible values are 1 to *n*.
PackageRedundancyMax	uint16	write(true)	PackageRedundancyMax describes the maximum number of spindles that can be used. Package redundancy describes how many disk spindles can fail without data loss including, at most, one spare. Examples would be RAID5 with a Package Redundancy of 1, RAID6 with 2. Possible values are 0 to *n*.
PackageRedundancyMin	uint16	write(true)	PackageRedundancyMin describes the minimum number of spindles that can be used. Package redundancy describes how many disk spindles can fail without data loss including, at most, one spare. Examples would be RAID5 with a Package Redundancy of 1, RAID6 with 2. Possible values are 0 to *n*.
PackageRedundancyDefault	uint16	write(true)	PackageRedundancyDefault describes the default number of spindles that can be used. Package redundancy describes how many disk spindles can fail without data loss including, at most, one spare. Examples would be RAID5 with a Package Redundancy of 1, RAID6 with 2. Possible values are 0 to *n*.
DeltaReservationMax	uint16	minvalue(1) maxvalue(100)	Delta reservation is a number between 1 (1 %) and a 100 (100 %) that specifies how much space should be reserved in a replica for caching changes. For a complete copy this would be 100 %, but it can be lower in some implementations. This parameter sets the upper limit.
DeltaReservationMin	uint16	minvalue(1) maxvalue(100)	Delta reservation is a number between 1 (1 %) and a 100 (100 %) that specifies how much space should be reserved in a replica for caching changes. For a complete copy this would be 100 %, but it can be lower in some implementations. This parameter sets the lower limit.

--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table 83 - Required Properties for StorageCapabilities (Continued)

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
DeltaReservationDefault	uint16	minvalue(1) maxvalue(100)	Delta reservation is a number between 1 (1 %) and a 100 (100 %) that specifies how much space should be reserved in a replica for caching changes. For a complete copy this would be 100 %, but it can be lower in some implementations. This parameter sets the default value.
Methods			
CreateSetting()			

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.3.3.10.12.5 ElementSettingData

Table 84 - Required Properties for ElementSettingData

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
ManagedElement	ref	key	The ManagedElement.
SettingData	ref	key	The Setting Data object associated with the ManagedElement.
IsDefault	uint16		An enumerated integer indicating that the referenced setting is a default setting for the element or that this information is unknown."),||ValueMap {"0", "1", "2"}, ||Values {"Unknown", "Is Default", "Is Not Default"}
IsCurrent	uint16		An enumerated integer indicating that the referenced setting is currently being used in the operation of the element or that this information is unknown."),||ValueMap {"0", "1", "2"}, ||Values {"Unknown", "Is Current", "Is Not Current"}

7.3.3.10.12.6 StorageSetting

Table 85 - Required Properties for StorageSetting

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
InstanceID	string	key	InstanceID opaquely identifies a unique instance of SettingData. The InstanceID shall be unique within a namespace.
ElementName	string	override, required	The user friendly name for this instance of SettingData. In addition, the user friendly name can be used as a index property for a search of query. (Note that the name does not have to be unique within a namespace.)
NoSinglePointOfFailure	boolean	write(true)	Indicates the desired value for No Single Point of Failure. Possible values are false = single point of failure and true = no single point of failure.
DataRedundancyMax	uint16	minvalue(1) write(true)	DataRedundancyMax describes the maximum number of complete copies of data to be maintained. Examples would be RAID 5 where 1 copy is maintained and RAID 1 where 2 or more copies are maintained. Possible values are 1 to *n*.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table 85 - Required Properties for StorageSetting (Continued)

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
DataRedundancyMin	uint16	minvalue(1) write(true)	DataRedundancyMin describes the minimum number of complete copies of data to be maintained. Examples would be RAID 5 where 1 copy is maintained and RAID 1 where 2 or more copies are maintained. Possible values are 1 to *n*.
PackageRedundancyMax	uint16	write(true)	PackageRedundancyMax describes the maximum number of spindles to be used. Package redundancy describes how many disk spindles can fail without data loss including, at most, one spare. Examples would be RAID5 with a Package Redundancy of 1, RAID6 with 2. Possible values are 0 to *n*.
PackageRedundancyMin	uint16	write(true)	PackageRedundancyMin describes the minimum number of spindles to be used. Package redundancy describes how many disk spindles can fail without data loss including, at most, one spare. Examples would be RAID5 with a Package Redundancy of 1, RAID6 with 2. Possible values are 0 to n.
DeltaReservationMax	uint16	minvalue(1), maxvalue(100)	Delta reservation is a number between 0 (0 %) and a 100 (100 %) that specifies how much space should be reserved in a replica for caching changes. For a complete copy this would be 100 %, but it can be lower in some implementations. Use 0 if copy service Subprofile is not supported.
DeltaReservationMin	uint16	minvalue(1), maxvalue(100)	
DeltaReservationGoal	uint16	minvalue(1), maxvalue(100)	

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.3.10.12.7 StorageSettingWithHints

Table 86 - Required Properties for StorageSettingWithHints

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
InstanceID	string	key	InstanceID opaquely identifies a unique instance of SettingData. The InstanceID shall be unique within a namespace.
ElementName	string	override, required	The user friendly name for this instance of SettingData. In addition, the user friendly name can be used as a index property for a search of query. (Note that the name does not have to be unique within a namespace.)
NoSinglePointOfFailure	boolean	write(true)	See description in StorageSetting table
DataRedundancyMax	uint16	write(true) minvalue(1)	See description in StorageSetting table.
DataRedundancyMin	uint16	write(true) minvalue(1)	See description in StorageSetting table.
PackageRedundancyMax	uint16	write(true)	See description in StorageSetting table.
PackageRedundancyMin	uint16	write(true)	See description in StorageSetting table.
DataAvailabilityHint	uint16	minvalue(0) maxvalue(10)	This hint is an indication from a client of the importance placed on data availability. Values are 0 = Don't Care to 10 = Very Important.
AccessRandomnessHint	uint16	minvalue(0) maxvalue(10)	This hint is an indication from a client of the randomness of accesses. Values are 0 = Entirely Sequential to 10 = Entirely Random.
AccessDirectionHint	uint16		This hint is an indication from a client of the direction of accesses. Values are 0 = Entirely Read to 10 = Entirely Write.
AccessSizeHint[ ]	uint16	minvalue(0) maxvalue(10)	This hint is an indication from a client of the optimal access sizes. Several sizes can be specified. Units(“Megabytes”).
AccessLatencyHint	uint16	minvalue(0) maxvalue(10)	This hint is an indication from a client how important access latency is. Values are 0 = Don't Care to 10 = Very Important.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table 86 - Required Properties for StorageSettingWithHints (Continued)

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
AccessBandwidthWeight	uint16	minvalue(0) maxvalue(10)	This hint is an indication from a client of bandwidth prioritization. Values are 0 = Don't Care to 10 = Very Important.
StorageCostHint	uint16	minvalue(0) maxvalue(10)	This hint is an indication of the importance the client places on the cost of storage. Values are 0 = Don't Care to 10 = Very Important. A StorageVolume provider might choose to place data on low-cost or high-cost drives based on this parameter.
StorageEfficiencyHint	uint16	minvalue(0) maxvalue(10)	This hint is an indication of the importance placed on storage efficiency by the client. Values are 0 = Don't Care to 10 = Very Important. A StorageVolume provider might choose different RAID levels based on this hint.
DeltaReservationMax	uint16	minvalue(1) maxvalue(100)	Delta reservation is a number between 0 (0 %) and a 100 (100 %) that specifies how much space should be reserved in a replica for caching changes. For a complete copy this would be 100 %, but it can be lower in some implementations. Use 0 if copy service Subprofile is not supported.
DeltaReservationMin	uint16	minvalue(1) maxvalue(100)	
DeltaReservationGoal	uint16	minvalue(1) maxvalue(100)	

7.3.3.10.12.8 HostedService

(As defined by CIM)

HostedService is an association between a Service and the System on which the functionality resides. The cardinality of this association is 1-to-many. A System may host many Services. Services are weak with respect to their hosting System. Heuristic: A Service is hosted on the System where the LogicalDevices or SoftwareFeatures that implement the Service are located. The model does not represent Services hosted across multiple systems. This is modelled as an ApplicationSystem that acts as an aggregation point for Services that are each located on a single host.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

HostedService is subclassed from Dependency**.**

Table 87 - HostedService Required Properties

Class Properties	Type	Qualifier/ Parameter	Notes
Antecedent	ref	override, max(1), min(1)	The hosting System.
Dependent	ref	override, weak	The Service hosted on the System.

7.3.3.10.13 Optional Subprofiles

Table 88 - Optional Profiles or Subprofiles

Name	Notes
Job Control	This subprofile is used to support copy services that run for a long time. The extrinsic methods support the “ConcreteJob” output. If job control is not supported this output is null.

7.3.3.11 LUN Creation Subprofile

7.3.3.11.1 Description

StorageVolumes are a required part of modelling disk storage systems (the Array, Out-of-band Virtualization and In-band Virtualization Profiles). However, user creation of storage volumes from pools is optional and may not be supported by a given disk storage system. The LUN Creation subprofile defines the support required if the storage system exposes functions for creating storage volumes from storage pools.

The StorageConfigurationService allows generic clients to configure storage arrays with volumes (for example, LUNs) without having to have specific knowledge about the storage system capacity. The service has the following methods for Storage Volume manipulation:

- CreateOrModifyElementFromStoragePool: Create a StorageVolume, possibly with a specific StorageSetting, from a source StoragePool. Note that this call is extensible to cover other types of object (e.g., NAS file systems) in the future;
- ReturnToStoragePool: Return an Element previously created with CreateOrModifyElement-FromStoragePool to the originating StoragePool.

The StorageCapabilities instances provide the ability to create settings for use in volume creation using the following method (part of the StorageCapabilities class):

- CreateSetting: Creates a setting that is consistent with the StorageCapabilities and may be modified before use in creating a StorageVolume.

The StoragePool instances provide the ability to retrieve the possible sizes for the StorageVolume creation or modification given a StorageSetting as a goal.

- GetAvailableSizes: Returns a list of discrete sizes given a goal.
- GetAvailableSizeRanges: Returns the range of possible sizes given a goal.

See Storage Pools and Storage Capabilities (7.3.3.10.8.1)

7.3.3.11.2 Standard dependencies

The LUN Creation subprofile is defined using the CIM Schema 2.8 final. As such, it can be used in profiles at 2.8 and later. It does not require that Profiles be on a later schema. It will operate within profiles that are at the

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

CIM schema 2.8 final or later. The subprofile will operate correctly with CIM Specification 2.2 (or later) and CIM Operations over HTTP 1.1 (or later).The Pool Manipulation Capabilities and Settings subprofile is based on the following standards:

Table 89 - LUN Creation Standard Dependencies

Standard	Version	Organization
CIM Specification	2.2	DMTF
CIM Operations over HTTP	1.1	DMTF
CIM Schema	2.8 Prelminary	DMTF

7.3.3.11.3 Profile dependencies

The LUN Creation subprofile introduces no Profile dependencies.

7.3.3.11.4 CIM Server requirements

7.3.3.11.4.1 Functional Profiles

Table 90 - Required Functional Profiles

Profile Required	Functional Group	Dependency
YES	Basic Read	None
YES	Basic Write	Basic Read
YES	Instance Manipulation	Basic Write
NO	Schema Manipulation	Instance Manipulation
YES	Association Traversal	Basic Read
NO	Query Execution	Basic Read
NO	Qualifier Declaration	Schema Manipulation
YES	Indication	None

7.3.3.11.4.2 Extrinsic methods

The CIM Server shall support extrinsic methods for the LUN Creation subprofile.

7.3.3.11.4.3 Discovery

The LUN Creation subprofile is not advertised.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.3.11.5 Instance diagrams

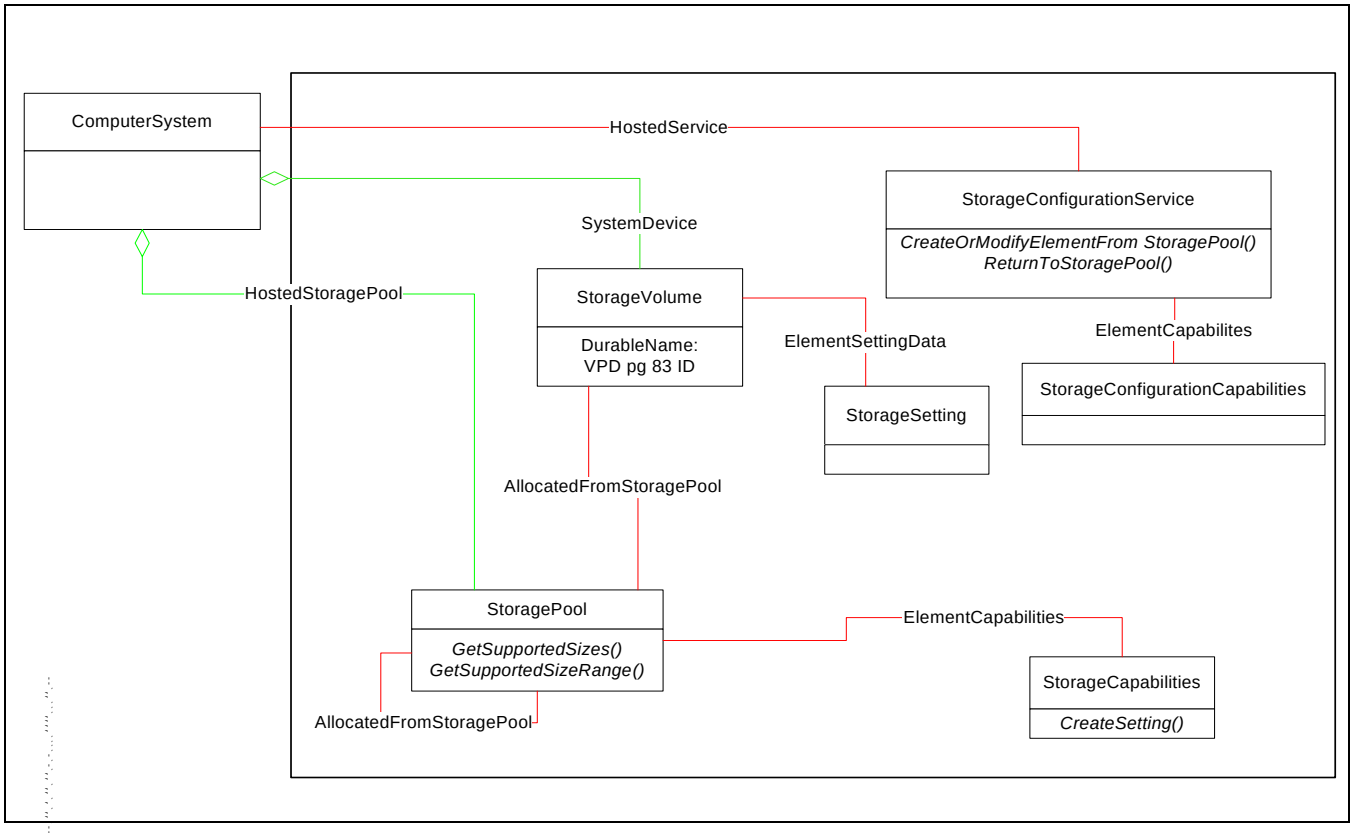


Figure 36 - LUN Creation Instance Diagram

7.3.3.11.6 Durable names and correlatable IDs

The LUN Creation subprofile does not add any durable names or correlatable IDs to the profiles (or subprofiles) in which it is used.

7.3.3.11.7 Methods

The LUN Creation subprofile introduces a number of write intrinsic and extrinsic methods.

StorageConfigurationService Methods:

CreateOrModifyElementFromStoragePool

```
Uint32 CreateOrModifyElementFromStoragePool (
    [In, Values {“Unknown”, “Reserved”, “StorageVolume”,
“StorageExtent”, “DMTF Reserved”, “Vendor Specific”},
ValueMap{“0”,”1”,”2”,”3”,”..”,”0x8000..”}]
Uint16 ElementType;
[Out] CIM_ConcreteJob ref Job,
 [In] CIM_StorageSetting ref Goal,
 [In, out] Uint64 Size,
 [In] CIM_StoragePool ref InPool,
 [Out, in] CIM_LogicalElement ref TheElement
);
```

- This method allows an Element of a type specified by the enumeration ElementType to be created from the input Storage Pool. The parameters are as follows:

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

- ElementType: This enumeration specifies what type of object to create. At present, only StorageVolume and StorageExtents are defined as values. However, other values (such as share) could be added in future.
- Job: see 7.3.3.9, “Job Control Subprofile”
- Goal: This is the Service Level that the Storage Volume is expected to provide. The setting shall be a subset of the Capabilities available from the parent Storage Pool. Goal may be a null value, in which case the default setting for the pool is used.
- Size: As an input this is the desired size of the Storage Volume. If it is not possible to create a volume of the desired size, a return code of “Size not supported” is returned with size set to the nearest supported size.
- InPool: This is a reference to a source Storage Pool.
- TheElement: If a reference is passed in, then that Element is modified, else this is a reference to the created element.

ReturnToStoragePool

```
Uint32 ReturnToStoragePool (
                          [Out] CIM_ConcreteJob ref Job,
                          [In] CIM_LogicalElement ref Element
                          );
```

This method is provided to allow a client to delete a previously created element such as a Storage Volume.

Return Values

Each method has a set of defined return codes as defined below:

```
ValueMap {“0”, “1”, “2”, “3”, “4”, “5”, “..”, “0x1000”,”0x1001”,
“0x1002..0x7777”, “0x8000..”},
Values {“Job completed with no error”, “Not Supported”, “Unknown”,
“Timeout”, “Failed”, “Invalid Parameter”, “DMTF Reserved”,
“Method parameters checked - job started”,
“Size not supported”, “Method Reserved”, “Vendor Specific”}]
```

If the method completes immediately with no errors (and with no asynchronous execution required), “Job completed with no error” is returned.

If the method parameters have been checked and the method is being executed asynchronously, “Method parameters checked - job started” is returned.

If, for a Create/Modify method, the requested size is not supported then “Size not supported” is returned and the Size parameter is set to the nearest supported size.

If one of the method parameters is incorrect (for instance invalid object paths), then “Invalid Parameter” is returned.

“Timeout” or “Failed” may be returned if the provider has problems accessing the hardware or for other implementation specific reasons.

A vendor may choose to extend the Value map to express vendor specific error codes not catered for by the standard errors.

7.3.3.11.8 Client considerations

Storage Volume Manipulation

The StorageConfigurationService class contains methods to allow creation, modification and deletion of StorageVolumes. The capabilities of a StorageConfigurationService or StoragePool to provide storage are

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

indicated using the StorageCapabilities class. This class allows the Service or Pool to advertise its capabilities (using implementation independent attributes) and a client to set the attributes it desires. The concept of ‘hints’ is also included that allows a client to provide clues to the system as to how it expects to use the storage for optimization purposes. These allow a client to provide extra information to 'tune' a StorageVolume. If a client chooses to supply these hints when creating a StorageVolume, the StorageSystem can either use them in determining a matching configuration or it can choose to ignore the hints. See the Storage Pools and Storage Capabilities (7.3.3.10.8.1) for further details on Storage Capabilities.

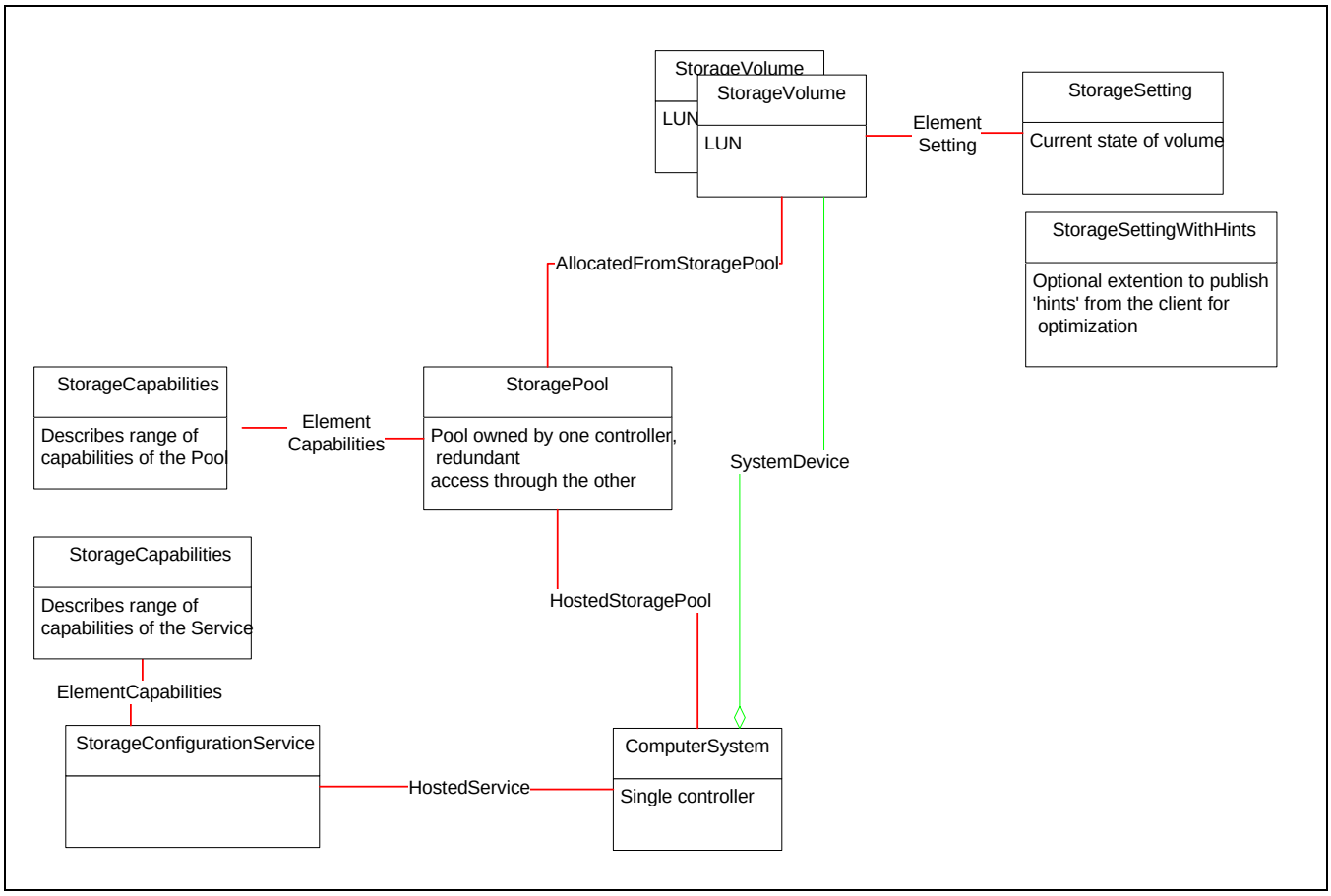


Figure 37 - Storage Pool Example

When creating a StorageVolume, an instance of StorageSetting is passed as a parameter to the StorageConfigurationService.CreateOrModifyElementFromStoragePool method. This forms an objective for that element to attempt to meet. The current ‘service level’ being achieved is reported via the StorageVolume class itself.

StorageVolumes are created from StoragePools via a StorageConfigurationService’s CreateOrModifyElementFromStoragePool() method. A volume create operation may take some period of time, however, and a Client needs to be aware that the operation is not complete until the StorageVoume.OperationalStatus is OK. A Client may also follow the progress of the operation using the ConcreteJob class and its properties.

The example below shows the classes and associations needed to model a single Pool with two StorageVolumes.

The methods are used as follows to create a Storage Volume.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

a) Similarly to Storage Pools, a client chooses a suitable source pool by referencing the StorageCapabilities objects and use, using the GetSupportedSizes and GetSupportSizeRange() objects. This is indicated in Figure 38.

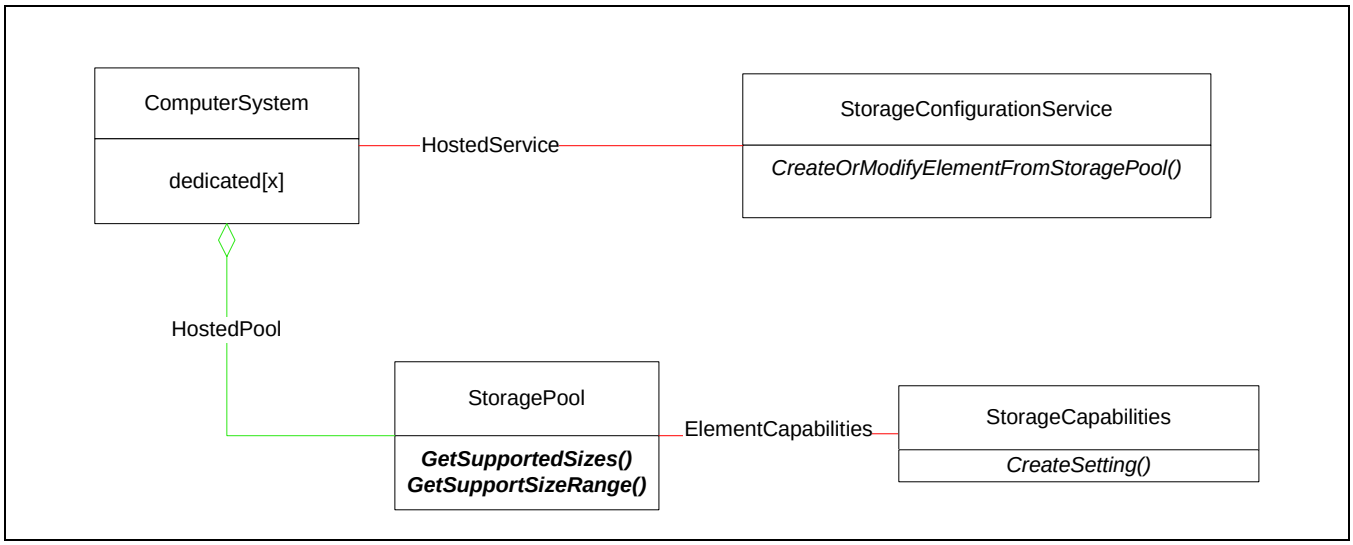


Figure 38 - Volume Creation - Initial State

b) Once a suitable pool is found, a StorageSetting instance can be created using the CreateSetting method on the StorageCapabilities object (see Figure 39). If a suitable StorageSetting already exists, it could be used instead.

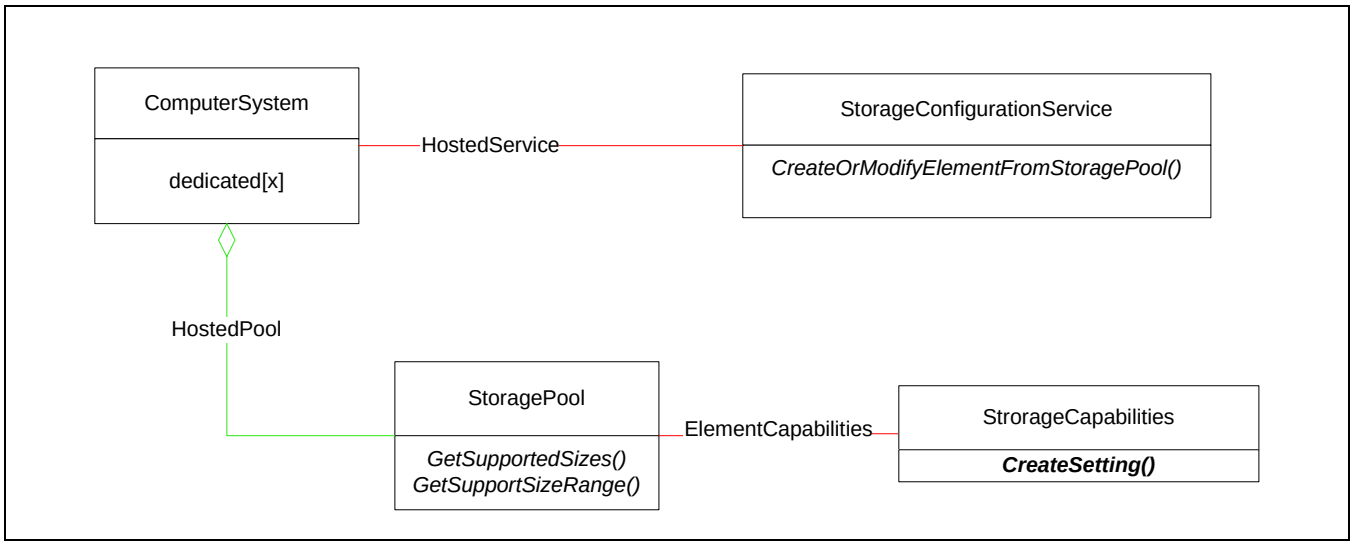


Figure 39 - Volume Creation - Step 1

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

c) If a new Setting is created, it is not linked back to the originating StorageCapabilities object until it is used as an argument in a StorageConfiguration method (see Figure 40).

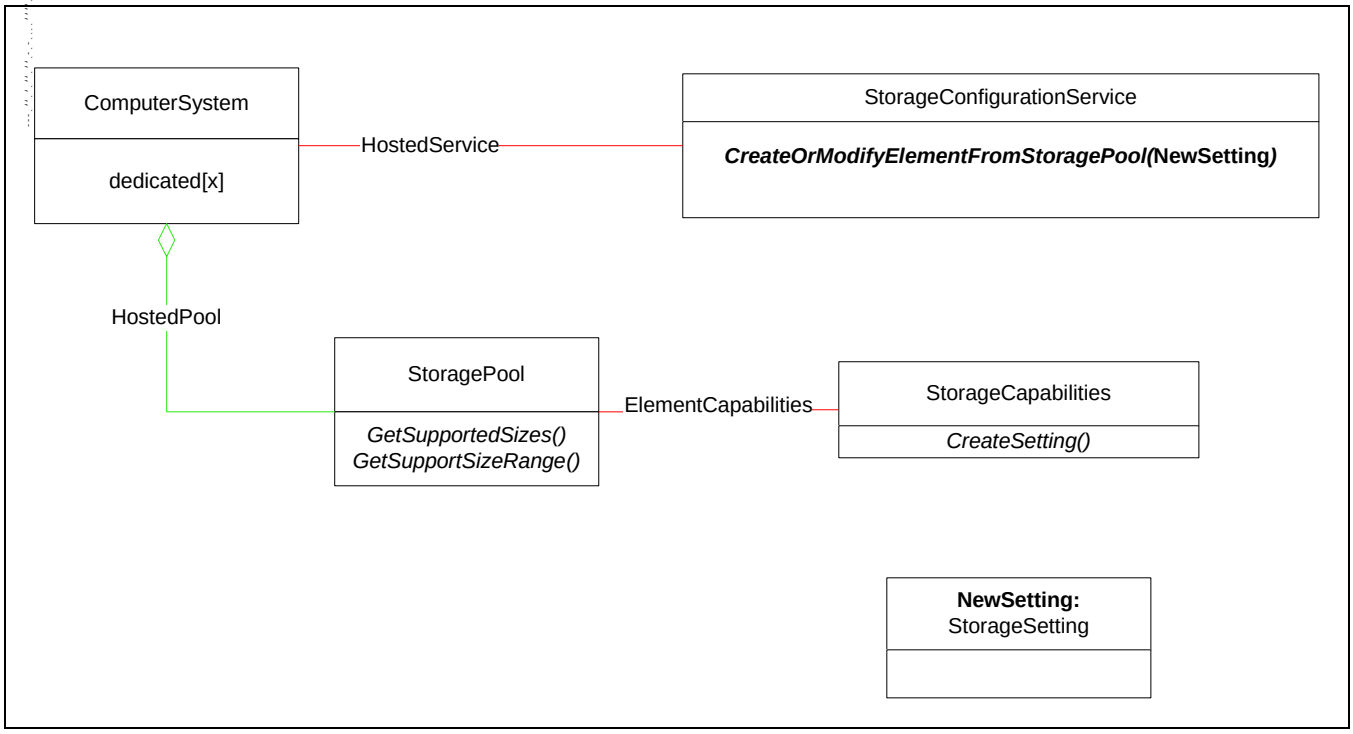


Figure 40 - Volume Creation - Step 2

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

d) Once the volume has been created, the new setting is ‘snapped’ to the new volume using the ElementSettingData association’ (see Figure 41).

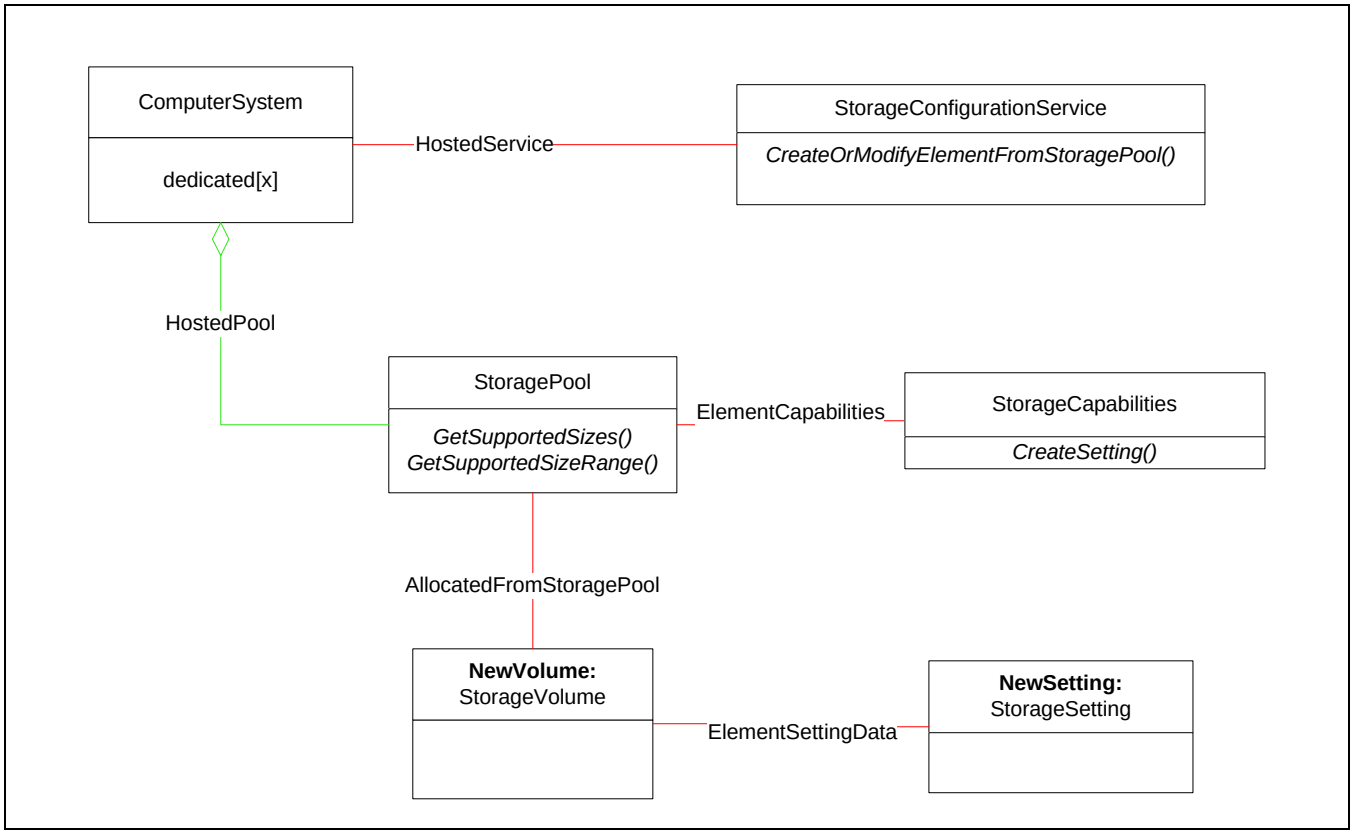


Figure 41 - Volume Creation - Step 3

7.3.3.11.9 Recipes

7.3.3.11.9.1 Create Storage Pool and Storage Volume on array

```
// DESCRIPTION
// The goal is to create the equivalent of a 10 GB RAID 5 Volume to
// house a tablespaces in a high-volume, transactional database.
// Records in the tablespace are about 1kB in size and reads of records
// occur much more frequently than writes.  First we have to create a
//  new Storage Pool to contain the volume
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
// 1.A reference to a CIM_ComputerSystem storage array is previously
//   defined in the $StorageArray-> variable
// 2.The settings for the new Storage Pool and Storage Volume are
//   defined in the following variables:
//      #RequestedSize     = 10 * 1024 * 1024 * 1024 // 10 GB
//      #NoSinglePOF       = true
//      #DataRedundancy    = 1
//      #PackageRedundancy = 1

// Function StorageSettingRequirementsAreSatisfiedBy
```

--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
// Determine if the storage requirements specified by required are
// met by offered.
sub boolean
                          StorageSettingRequirementsAreSatisfiedBy($Ca
                          pabilitiesOffered)
{
     if(#NoSinglePOF == $CapabilitiesOffered.NoSinglePointOfFailure
        && #DataRedundancy <= $CapabilitiesOffered.DataRedundancyMax
        && #DataRedundancy >= $CapabilitiesOffered.DataRedundancyMin
        && #PackageRedundancy <=
                          $CapabilitiesOffered.PackageRedundancyMax
        && #PackageRedundancy >=
                          $CapabilitiesOffered.PacakageRedundancyMin)
     {
        return true
     }
     else
     {
        return false
     }
}

// Function PoolSizeAvailable
// A return value of 0 means that no size is available
sub unit32 PoolSizeAvailable($PoolToDrawFrom->,
        $StorageSetting->, #RequestedSize)

     #ResultSize = 0
     %InArguments[“ElementType”] = 2 // StoragePool
     %InArguments[“Goal”] = $StorageSetting->
     #MethodReturn = InvokeMethod(
        $PoolToDrawFrom->,
        “GetSupportedSizes”,
        %InArguments,
        %OutArguments)
     if(#MethodReturn == 0)
     {
         // this method is supported
        #SupportedSizes[] = %OutArguments[“Sizes”]
        #i = 0
        #max = #SupportedSizes[].length
        while(#i < #max && #RequestedSize > #ResultSize)
        {
           #ResultSize = #SupportedSizes[#i++]
        }
        if(#RequestedSize > #ResultSize)
        {
           // we did not find a size
```

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
            #ResultSize = 0
         }
      }
      else if (#MethodReturn == 2)
      { // call GetSupportedSizeRange
         #MethodReturn =
         InvokeMethod(
            $PooltoDrawFrom->,
            “GetSupportedSizeRange”,
            %InArguments,
            %OutArguments)
         if(#MethodReturn != 1 && #MethodReturn != 2)
         {
            // this method is supported
            #MaximumVolumeSize = %OutArguments[“MaximumVolumeSize”]
            #MinimumVolumeSize = %OutArguments[“MinimumVolumeSize”]
            #VolumeSizeDivisor = %OutArguments[“VolumeSizeDivisor”]
            if(#RequestedSize >= #MinimumVolumeSize &&
               #RequestedSize <= #MaximumVolumeSize)
            {
               // Rounding up using integer arthimetic
               #ResultSize =
                  (#RequestedSize / #VolumeSizeDivisor + 1 )
                     * #VolumeSizeDivisor
            }
         }
      }
      return #ResultSize
}

// MAIN
// Step 1. Get the configuration services and determine the service
//   capabilities
$Services->[] = AssociatorNames(
      $StorageArray->,
      “CIM_HostedService”,
      “CIM_StorageConfigurationService”,
      null,
      null)

// There should be only one storage configuration service
// Associated with the system
$StorageConfigurationService-> = $Services->[0]
$ServiceCapabilities[] = Associators(
      $StorageConfigurationService->,
      “CIM_ElementCapabilities”,
      “CIM_StorageConfigurationCapabilities”,
```

--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
     null,
     null,
     false,
     false,
     null)

// There should be only one StorageConfigurationCapabilities instance
#SupportsPoolCreation = contains(
     2, // Storage Pool Creation
     $ServiceCapabilities[0].SupportedSynchronousActions[]) ||
     contains(
     2, // Storage Pool Creation
     $ServiceCapabilities[0].SupportedAsynchronousActions[]))
#PoolCreationProducesJob = contains(
     2, // Storage Pool Creation
     $ServiceCapabilities[0].SupportedAsyncronousActions[])
#SupportsVolumeCreation1 = contains(
     5, // Storage Element Creation
     $ServiceCapabilities[0].SupportedSynchronousActions[])
#SupportsVolumeCreation2 = contains(
     3, // StorageVolumeCreation
     $ServiceCapabilities[0].SupportedStorageElementFeatures[])
#VolumeCreationProducesJob = contains(
     5, // Storage Element Creation
     $ServiceCapabilities[0].SupportedAsynchronousActions[])
// If a volume can not be created and that volume is
// neither created synchronously or asynchronously, then fail the test
if (!#SupportedVolumeCreation2 &&
     !(#SupportedVolumeCreation1 || #VolumeCreationProducesJob))
{
     <EXIT; The StoragePool can be created, but the
        StorageVolume creation is not supported.>
}

// Step 2. Enumerate over the CIM_HostedStoragePool associations to
                            find
//   all the StoragePools from which volumes might be created.
$StoragePools[] = Associators(
     $StorageArray->,
     “CIM_HostedStoragePool”,
     “CIM_StoragePool”,
     null,
     null,
     false,
     false,
     {“InstanceID”, “Primordial”})
```

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
// Step 3. For each StoragePool, follow the CIM_ElementCapabilities
//   asociation to the StorageCapabilities of that pool. Compare the
//   StorageCapabilities to the desired StorageSetting and find the
//   best match.
// Step 4. Determine if the selected pool has enough space for
//   another pool.  Change the StorageSetting to what is desired.
//   If the array supports hints, then the Storage Setting returned
//   will contain default hints
$PoolToDrawFrom-> = null
for #i in $StoragePools[]
{
     // If we can not create Storage Pool, then find a ‘concrete’
     // Storage Pool from which to create a Storage Volume
     #UsePrimordial = false
     if(#SupportsPoolCreation)
     {
        #UsePrimordial = true
     }
     if ($StoragePools[#i].Primodial == #UsePrimordial)
     {
        $CapabilitiesOffered[] = Associators(
           $StoragePools[#i].getObjectPath(),
           “CIM_ElementCapabilities”,
           “CIM_StorageCapabilities”,
           null,
           null,
           false,
           false,
           null)
        $StorageCapabilitiesOffered = $CapabilitiesOffered[0]
        if(&StorageSettingRequirementsAreSatisfiedBy(
           $StorageCapabilitiesOffered))
        {
           $PoolToDrawFrom-> = $StoragePool[#i].getObjectPath()
           // --> Step 4
           // Create a setting
           %InArguments[“SettingType”] = 2
           // Default
           #ReturnValue = InvokeMethod(
              $StorageCapabilitiesOffered.getObjectPath(),
              “CreateSetting”,
              %InArguments,
              %OutArguments)
           if (#ReturnValue != 0)
           {
              <ERROR! Unable to create storage setting >
           }
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
                $GeneratedStorageSetting-> = %OutArguments[“NewSetting”]
                $ModifiedSetting = GetInstance(
                   $GeneratedStorageSetting->,
                   false,
                   false,
                   false,
                   null)
                $ModifiedSetting.DataRedundancyGoal = #DataRedundancy
                $ModifiedSetting.PackageRedundancyGoal = #PackageRedundancy
                ModifyInstance(
                   $ModifiedSetting,
                   false,
                   NULL)
                // Determine the possible size, closest to the requested
                                size
                #PossibleSize = &PoolSizeAvailable(
                   $PoolToDrawFrom->,
                   $ModifiedSetting.getObjectPath(),
                   #RequestedSize)
                if(0 != #ResultSize)
                {
                   break;
                }
                else
                {
                // Cause failure if there are no more candidate Pools
                   $PoolToDrawFrom-> = NULL;
                }
             }
          }
       }
       if ($PoolToDrawFrom-> == NULL)
       {
          < ERROR! Unable to find a suitable pool from which to create the
                                volume >
       }

       // Step 5. Register for indications on configuration jobs
       If(#PoolCreationProducesJob || #VolumeCreateProducesJob)
       {
          %Filter = “SELECT * FROM CIM_InstModification
                    WHERE SourceInstance ISA CIM_ConcreteJob
                      AND PreviousInstance.OperationalStatus !=
                                SourceInstance.OperationalStatus
          @{Determine if Indications already exist or have to be created}
          &createIndication(%Filter)
       }
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
// Step 6. Create the Storage Pool
if(#SupportsPoolCreation)
{
     %InArguments[“ElementName”] = NULL// we do not care what
                     // the name is
     %InArguments[“Goal”] = $ModifiedSetting.getObjectPath()
     %InArguments[“Size”] = #PossibleSize
     %InArguments[“InExtents”] = null
     %InArguments[“Pool”] = null
     $InPools->[0] = $PoolToDrawFrom->
     %InArguments[“InPools”] = $InPools->[]
     #ReturnValue = InvokeMethod(
        $StorageConfigurationService->,
        “CreateOrModifyStoragePool”,
        %InArguments, %OutArguments)
     if(#ReturnValue != 0 || #ReturnValue != 4096)
     {   // Storage Pool was not created
        <ERROR! Failed >
     }
     $PoolToDrawFrom-> = %OutArguments[“Pool”]
     $PoolCreationJob-> = %OutArguments[“Job”]

     if(#PoolCreationProducesJob && $PoolCreationJob-> != null)
     {
        <Wait until the completion of the job
         using $PoolCreationJob-> as a filter
         Verify that the OperationalStatus array
         contains ‘2’, “OK”, or ‘17’, “Completed”>
     }
     $CapabilitiesOffered[] = Associators(
        $PoolToDrawFrom->,
        “CIM_ElementCapabilities”,
        “CIM_StorageCapabilities”,
        null,
        null,
        false,
        false,
        null)
     $StorageCapabilitiesOffered = $CapabilitiesOffered[0]
}

// Step 7. Create Storage Volume.
%InArguments[“SettingType”] = 2 // “Default”
InvokeMethod(
     $StorageCapabilitiesOffered.getObjectPath(),
     “CreateSetting”,
     %InArguments,
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
     %OutArguments)
$GeneratedStorageSetting-> = %OutArguments[“NewSetting”]
%InArguments[“ElementType”] = 2 // Storage Volume
%InArguments[“Goal”] = $GeneratedStorageSetting->
%InArguments[“Size”] = #PossibleSize
$InPools->[0] = $PoolToDrawFrom->
%InArguments[“InPool”] = $InPools->
%InArguments[“TheElement”] = null
#ReturnValue = InvokeMethod(
     $StorageConfigurationService->,
     “CreateOrModifyElementFromStoragePool”,
     %InArguments, %OutArguments)
if(#ReturnValue != 0 || #ReturnValue != 4096)
{   // Method did not succeeded or succeeded but did not create a job
     <ERROR! Failed >
}
else if(#ReturnValue == 0 ||
     #ReturnValue == 4096 && %OutArguments[“TheElement”] != null))
{
     $CreatedVolume-> = %OutArguments[“TheElement”]
}
else // a Job was created and TheElement is null
{
     <Coerse the string returned as the Job into
      a ObjectName yielding the $Job-> variable>
     <Wait until the completion of the job using $Job-> as a filter
      Verify that the OperationalStatus array
      contains ‘2’, “OK”, or ‘17’, “Completed”>
     <Once the ‘Job’ has completed, see step 5, then follow the
      AffectedJobElement association from the ‘Job’ to retrieve
      the volume that was created.>
     $CreateVolumes[] = Associators(
        $Job->, // Object Name coersed from %OutArguments[“Job”]
        “CIM_AffectedJobElement”,
        “CIM_StorageVolume”,
        null,
        null,
        false,
        false,
        null)
     // Only one StorageVolume will be created,
    $CreatedVolume-> = $CreatedVolume[0].getObjectPath()
}
```

7.3.3.11.9.2 Expand Storage Volume on storage array

```
// DESCRIPTION
// In this recipe, we attempt to expand a LUN on an array by 50%.
```

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
// 1.A reference to the CIM_ComputerSystem that represents the array
//   $StorageArray->
// 2.A reference to the particular StorageVolume we wish to expand.
//   $VolumeToExpand->
// 3.It is assumed that to expand a Storage Volume there needs to be
//   enough space available in the parent StoragePool to contain
//   another copy of the Storage Volume whose size is equal to the
//   new size requested.    This is especially the case if we were
//   modifying the settings as well as the size.

// Step 1. Get the configuration services and determine the service
//   capabilities
$Services->[] = AssociatorNames(
     $StorageArray->,
     “CIM_HostedService”,
     “CIM_StorageConfigurationService”,
     null,
     null)

// There should be only one storage configuration service
// Associated with the system
$StorageConfigurationService-> = $Services->[0]
$ServiceCapabilities[] = Associators(
     $StorageArray->,
     “CIM_ElementCapabilities”,
     “CIM_StorageConfigurationCapabilities”,
     null,
     null,
     false,
     false,
     null)

// There should be only one StorageConfigurationCapabilities instance
#SupportsVolumeModification1 = contains(
     7, // Storage Element Modification
     $ServiceCapabilities[0].SupportedSynchronousActions[]) ||
     contains(
     7, // Storage Element Modification
     $ServiceCapabilities[0].SupportedAsynchronousActions[])
#SupportsVolumeModification2 = contains(
     5, // StorageVolume Modification
     $ServiceCapabilities[0].SupportedStorageElementFeatures[])
#VolumeModificationProducesJob = contains(
     7, // Storage Element Modification
     $ServiceCapabilities[0].SupportedAsynchronousActions[])
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
if(!#SupportedVolumeModification1 || !#SupportedVolumeModification2)
{
     <EXIT: The ability to modify an existing StorageVolume must be
                          supported
      to continue.>
}

// Step 2. Read the current size of the StorageVolume.
$Volume = GetInstance(
     $VolumeToExpand->,
     false,
     false,
     false,
     {“BlockSize”, “NumberOfBlocks”})
#PreviousSize = $Volume.BlockSize * $Volume.NumberOfBlocks

// Step 3. Follow the AllocatedFromStoragePool association from the
//   volume to find the pool from whence it came.
$Pools->[] = AssociatorNames(
     $VolumeToExpand->,
     “CIM_AllocatedFromStoragePool”,
     “CIM_StoragePool”,
     null,
     null)

// A Storage Volume has only one Pool parent
$ParentPool-> = $Pools->[0]

// Step 4. Determine whether the desired space for which to expand the
//   volume exists within the pool.
$StorageSetting->[] = AssociatorNames(
     $VolumeToExpand->,
     “CIM_ElementSettingData”,
     “CIM_StorageSetting”,
     null,
     null)
$CurrentVolumeSetting-> = $StorageSetting->[0]
// Calculate the additional space needed
#SizeToExpand   = 0.5 * #PreviousSize
// Calculate 150% of previous volume size
#SizeToExpandTo = #PreviousSize + (0.5 * #PreviousSize)
#NewSizeAvailable =
     @<Create Storage Pool and Volume on array>
        &PoolSizeAvailable(
            $ParentPool->,
            $CurrentVolumeSetting->,
            #SizeToExpand)
if (!#NewSizeAvailable)
```

--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
{
     < ERROR! Unable to proceed because the requested size is
                          unavailable >
}

// Step 5. Register for indications on configuration jobs
If(#VolumeModificationProducesJob)
{
     %Filter = “SELECT * FROM CIM_InstModification
                WHERE SourceInstance ISA CIM_ConcreteJob
                 AND PreviousInstance.OperationalStatus !=
                          SourceInstance.OperationalStatus”
     @{Determine if Indications already exist or have to be created}
     &createIndication(%Filter)
}

// Step 6. Modify the Storage Volume
// If there is a Job produced, wait for Job completion
%InArguments[“ElementName”] = null// we do not care what the name is
%InArguments[“ElementType”] = 2// Storage Volume
%InArguments[“Goal”] = $CurrentVolumeSetting
%InArguments[“Size”] = #SizeToExpandTo
%InArguments[“InPool”] = null
%InArguments[“TheElement”] = $VolumeToExpand->
#ReturnValue = InvokeMethod(
     $StorageConfigurationService->
     “CreateOrModifyElementFromStoragePool”
     %InArguments
     %OutArgument
     )
if(#ReturnValue != 0 || #ReturnValue != 4096)
{   // Method succeeded or validated arguments and started a job
     <ERROR! Failed >
}
else if(#ReturnValue == 0 ||
     #ReturnValue == 4096 && %OutArguments[“TheElement”] != null)
{
     $CreatedVolume-> = %OutArguments[“TheElement”]
}
else // a Job was created and TheElement is null
{
     <Coerse the string returned as the Job into
      a ObjectName yielding the $Job-> variable>
     <Wait until the completion of the job
      using $PoolCreationJob as a filter
      Verify that the OperationalStatus array
      contains ‘2’, “OK”, or ‘17’, “Completed”>
     <Once the ‘Job’ has stopped, see step 4,then follow the
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
      AffectedJobElement association from the 'Job' to retrieve
      the volume that was created.>
     $CreateVolumes[] = Associators(
        $Job->, // Object Name coersed from %OutArguments["Job"]
        "CIM_AffectedJobElement",
        "CIM_StorageVolume",
        null,
        null,
        false,
        false,
        null)
     // Only one StorageVolume will be created,
    $CreatedVolume-> = $CreatedVolume[0].getObjectPath()
}

// Step 7. Check the value of the "Size" out parameter.  See if it is
//   equal to size expected. If so, we got what we asked for and we're
                              done.
#SizeExpandedTo = %OutArguments["Size"]
if (#SizeExpandedTo == #SizeToExpandTo)
{
     < indicate the volume was successfully expanded >
}
else
{
     if (#SizeExpandedTo <= #PreviousSize)
     {
        < indicate the volume was not expanded >
     }
     else
     {
        < indicate the volume was only partially expanded to
                              #SizeExpandedTo >
     }
}
```

7.3.3.11.10 Instrumentation requirements

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.3.11.10.1 Required CIM Elements

Table 91 - Required CIM Elements

Profile Classes & Associations	Notes
StorageConfigurationService (7.3.3.11.11.1)	
Packages/Profiles	
Pool Manipulation Capabilities and Settings Subprofile (7.3.3.10)	
Methods	
CreateOrModifyElementFromStoragePool()	
ReturnToStoragePool()	
SubProfile Indications	
Creation/Deletion of StorageVolume	SELECT * from CIM_InstCreation where SourceInstance ISA CIM_StorageVolume SELECT * from CIM_InstDeletion where SourceInstance ISA CIM_StorageVolume

7.3.3.11.11 Required Properties for CIM Elements

7.3.3.11.11.1 StorageConfigurationService

Table 92 - Required Properties for StorageConfigurationService

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
ElementName	string		User Friendly name.
SystemCreationClassName	string	maxlen(256), key, propagated	The scoping System's CreationClassName.
SystemName	string	maxlen(256), key, propagated	The scoping System's Name.
CreationClassName	string	maxlen(256), key	The name of the concrete subclass.
Name	string	maxlen(256), key, override	
CreateOrModifyElementFromStoragePool()	uint32		
ReturnToStoragePool ()	uint32		

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.3.11.12 Optional subprofiles

Table 93 - Optional Profiles or Subprofiles

Name	Notes
Job Control Subprofile (7.3.3.9)	This subprofile is used to support copy services that run for a long time. The extrinsic methods support the “ConcreteJob” output. If job control is not supported this output is null.

7.3.3.12 Device Credentials subprofile

7.3.3.12.1 Description

Many devices require a shared secret to be provided to access them. This shared secret is different from the credentials used by the SMI-S Client for authentication with the CIM Server. This Subprofile is used to change this device’s shared secrets.

The SMI-S Client shall not be provided with the password, only the principle. The SMI-S Client can use the principle to change the shared secret appropriately.

The device credentials can be exposed throughout the CIM model such that a CIM Client may manipulate them. The credentials are modelled as shared secrets.

7.3.3.12.2 Standard dependencies

The Device Credentials subprofile is based on the following standards:

Table 94 - Device Credentials Standard Dependencies

Standard	Version	Organization
CIM Specification	2.2	DMTF
CIM Operations over HTTP	1.1	DMTF
CIM Schema	2.7	DMTF

7.3.3.12.3 Profile dependencies

The Device Credentials subprofile does not require any other Profiles.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.3.12.4 CIM Server requirements

7.3.3.12.4.1 Functional Profiles

Table 95 - Required Functional Profiles

Profile Required	Functional Group	Dependency
YES	Basic Read	None
YES	Basic Write	Basic Read
YES	Instance Manipulation	Basic Write
NO	Schema Manipulation	Instance Manipulation
YES	Association Traversal	Basic Read
NO	Query Execution	Basic Read
NO	Qualifier Declaration	Schema Manipulation
YES	Indication	None

7.3.3.12.4.2 Extrinsic methods

The CIM Server support for extrinsic methods is optional for the Device Credentials subprofile.

7.3.3.12.4.3 Discovery

The DeviceCredentials subprofile is not advertised.

7.3.3.12.5 Instance diagrams

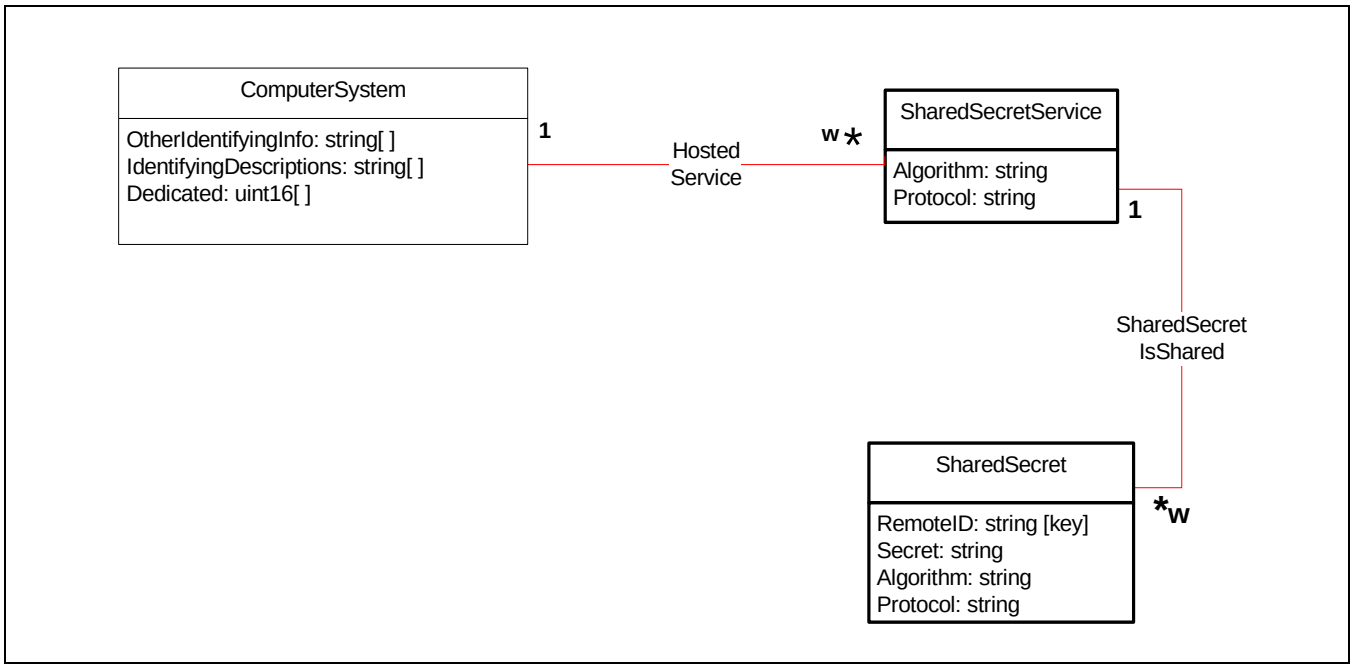


Figure 42 - DeviceCredentials Subprofile Model

7.3.3.12.6 Durable names and correlatable IDs

There are no durable names nor correlatable IDs for the Device Credentials subprofile.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.3.3.12.7 Methods

ModifyInstance:

Only the SharedSecret in this SubProfile may be modified. SharedSecrets shall not be created nor deleted.

If that instance is modified then the CIM Agent attempts to authenticate again with the managed device using the current RemoteID and Secret. If the RemoteID is changed, then the user name is being changed. If the Secret is changed, then the password is being changed.

If the re-authentication fails, then the RemoteID and Secret revert back to their previous state.

7.3.3.12.8 Client considerations

There shall be only one shared secret per device accessible to a given CIM Client.

7.3.3.12.9 Recipes

No recipes have been defined for this subprofile.

7.3.3.12.10 Instrumentation requirements

None.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.3.12.11 Required CIM Elements

Table 96 - Required CIM Elements

Profile Classes & Associations	Notes
SharedSecretService (7.3.3.12.12.1)	
SharedSecret (7.3.3.12.12.2)	
SharedSecretIsShared (7.3.3.12.12.3)	
HostedService (7.3.3.12.12.4)	
Packages	
None	
Methods	
None	
SubProfile Indications	
None	

7.3.3.12.12 Required Properties for CIM Elements

7.3.3.12.12.1 SharedSecretService

Table 97 - Required Properties for SharedSecretService

Class Properties	Type	Qualifier/ Parameter	Notes
ElementName	string		User Friendly name.
SystemCreationClass Name	string	maxlen(256), key, propagated	The scoping System's CreationClassName..
SystemName	string	maxlen(256), key, propagated	The scoping System's Name.
CreationClassName	string	maxlen(256), key	The name of the concrete subclass.
Name	string	maxlen(256), key, override	

7.3.3.12.12.2 SharedSecret

CIM_SharedSecret is subclassed from CIM_Credential

Table 98 - Required Properties for SharedSecret

Class Properties	Type	Qualifier/ Parameter	Notes
SystemCreationClass Name	string	maxlen(256), key, propagated	The scoping System's CreationClassName.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table 98 - Required Properties for SharedSecret (Continued)

Class Properties	Type	Qualifier/ Parameter	Notes
SystemName	string	maxlen(256), key, propagated	The scoping System's Name.
ServiceCreationClass Name	string	maxlen(256), key, propagated	The CreationClassName of the associated SharedSecretService.
ServiceName	string	maxlen(256), key, propagated	The Name of the associated SharedSecretService.
RemoteID	string	maxlen(256), key	The user name in the device's shared secret.
Secret	string		The password in the device's shared secret. The property shall not be shown in its true form for a SharedSecret once the SharedSecret has been created. Instead, some printable character should be repeated for each character in the Secret.

7.3.3.12.12.3 SharedSecretIsShared

CIM_SharedSecretIsShared is subclassed from CIM_ManagedCredential

Table 99 - SharedSecretIsShared Required Properties

Class Properties	Type	Qualifier/ Parameter	Notes
Antecedent	ref	override, max(1), min(1)	The credential management service.
Dependent	ref	override, weak	The managed credential.

7.3.3.12.12.4 HostedService

(As defined by CIM.)

HostedService is an association between a Service and the System on which the functionality resides. The cardinality of this association is one-to-many. A System may host many Services. Services are weak with respect to their hosting System. Heuristic: A Service is hosted on the System where the LogicalDevices or SoftwareFeatures that implement the Service are located. The model does not represent Services hosted across multiple systems. This is modelled as an ApplicationSystem that acts as an aggregation point for Services that are each located on a single host.

HostedService is subclassed from Dependency.

Table 100 - HostedService Required Properties

Class Properties	Type	Qualifier/ Parameter	Notes
Antecedent	ref	override, max(1), min(1)	The hosting System.
Dependent	ref	override, weak	The Service hosted on the System.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.3.12.12.5 Optional subprofiles

Table 101 - Optional Profiles or Subprofiles

Name	Notes
None	

7.3.3.13 Backend Ports subprofile

7.3.3.13.1 Description

Some RAID systems provide interfaces to discover and manage the internal connections between the RAID processors and physical disks. For example, an array may have an interface to acquire and optimize the utilization of separate buses, loops or fabrics to back-end storage. In this case, the ports to individual disks can be modelled similarly to a JBOD configuration as well as the ports on the RAID processors.

A property on FCPort called UsageRestriction is available to indicate whether the controller is providing a front end (target) or back end (initiator) interface.

The RAID controller itself has front-end ports (connected to customer hosts or switches) and back-end ports (connected to the internal disks). Table 43 shows an instance diagram for three disks (StorageExent only shown) in an array, connected by a FC loop. The full model for the disk is shown in 7.3.3.4, “Disk Drive subprofile”.

7.3.3.13.2 Standard dependencies

The Backend Ports subprofile is based on the following standards:

Table 102 - Device Credentials Standard Dependencies

Standard	Version	Organization
CIM Specification	2.2	DMTF
CIM Operations over HTTP	1.1	DMTF
CIM Schema	2.8 Preliminary	DMTF

7.3.3.13.3 Profile dependencies

The Backend Ports subprofile introduces no Profile dependencies.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.3.13.4 CIM Server requirements

7.3.3.13.4.1 Functional Profiles

Table 103 - Required Functional Profiles

Profile Required	Functional Group	Dependency
YES	Basic Read	None
NO	Basic Write	Basic Read
NO	Instance Manipulation	Basic Write
NO	Schema Manipulation	Instance Manipulation
YES	Association Traversal	Basic Read
NO	Query Execution	Basic Read
NO	Qualifier Declaration	Schema Manipulation
YES	Indication	None

7.3.3.13.4.2 Extrinsic methods

The CIM Server support for extrinsic methods is optional for the Backend Ports subprofile.

7.3.3.13.4.3 Discovery

The Backend Ports subprofile is not advertised.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.3.13.5 Instance diagrams

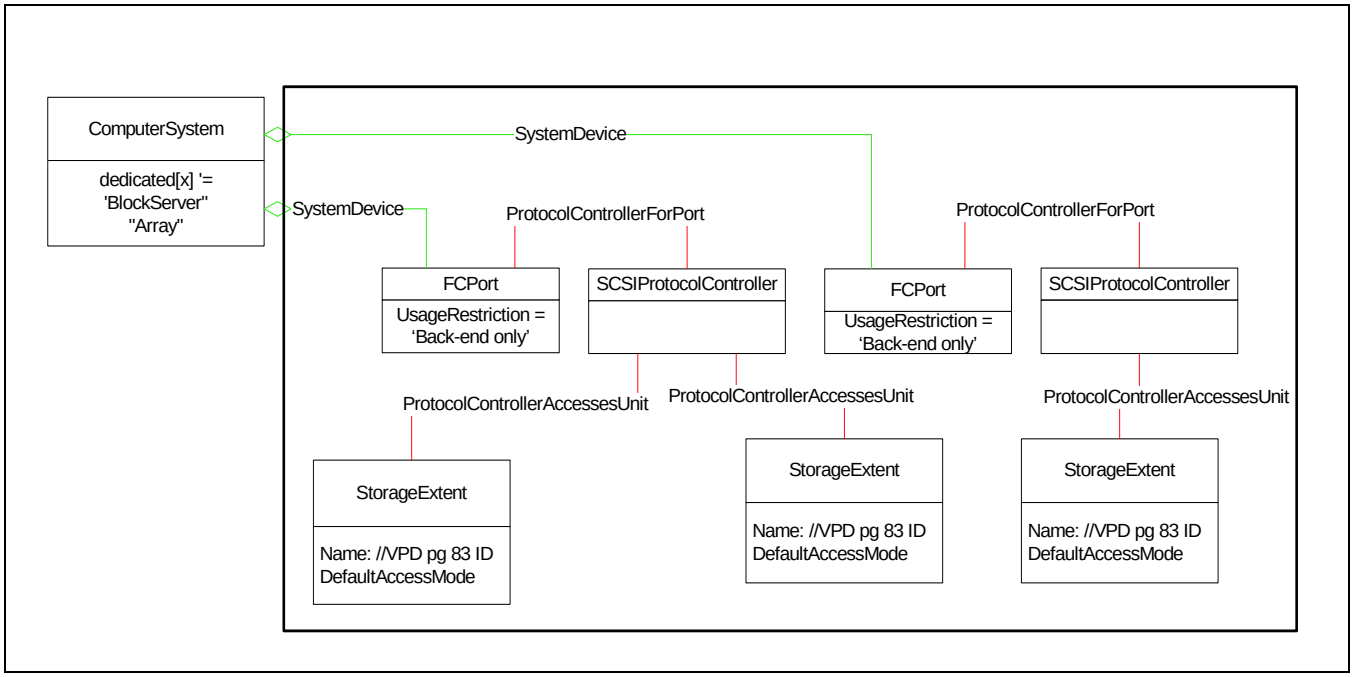


Figure 43 - Back-end Ports Instance

7.3.3.13.6 Durable names and correlatable IDs

See parent clauses.

7.3.3.13.7 Methods

See parent clauses.

7.3.3.13.8 Client considerations

See parent clauses.

7.3.3.13.9 Recipes

See parent clauses.

7.3.3.13.10 Instrumentation requirements

See parent clauses.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.3.13.11 Required CIM Elements

Table 104 - Required CIM Elements

Profile Classes & Associations	Notes
FCPort (7.3.3.13.12.1)	
ProtocolControllerForPort (7.3.3.13.12.2)	OPTIONAL; mandatory for FCPorts.
ProtocolControllerAccessesUnit (7.3.3.13.12.3)	
SCSIProtocolController (7.3.3.13.12.4)	
StorageExtent (7.3.3.13.12.5)	
SystemDevice (7.3.3.13.12.6)	
Packages	
None	
Associated Indications	
None	

7.3.3.13.12 Required Properties for CIM Elements

7.3.3.13.12.1 FCPort

Table 105 - Required Properties for FCPort

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
SystemCreationClassName	string	key	
SystemName	string	key	
CreationClassName	string	key	
UsageRestriction	uint16	req	Usage is Backend Port.
ElementName	string		User friendly name/caption for port. This property is OPTIONAL.
OperationalStatus[ ]	Uint16		Status of device.
DeviceID	string	key	Opaque.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table 105 - Required Properties for FCPort (Continued)

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
PortType	uint16		FC-GS Port.Type The specific mode currently enabled for the Port. The values: \"N\" = Node Port, \"NL\" = Node Port supporting FC arbitrated loop, \"E\" = Expansion Port connecting fabric elements (for example, FC switches), \"F\" = Fabric (element) Port, \"FL\" = Fabric (element) Port supporting FC arbitrated loop, and \"B\" = Bridge Port. PortTypes are defined in ISO/IEC 14165-414. When set to 1 (\"Other\"), the related property OtherPortType contains a string description of the port's type. PortType is defined to force consistent naming of the 'type' property in subclasses and to guarantee unique enum values for all instances of NetworkPort. A range of values, DMTF_Reserved, has been defined that allows subclasses to override and define their specific port types. Vendor Reserved = 16000..65535 can be used if the PortType is not one already defined in the above enumerations and a vendor subclass is defined specifying the appropriate “value” and “valuemap”. This property is OPTIONAL.
PermanentAddress	string		The WWN of the port. This property is OPTIONAL.
NetworkAddresses[ ]	string		The Fibre Channel address of the port. This property is OPTIONAL.
Speed	uint64		Speed of zero represents a link not established. 1 Gbit is 1 062 500 000 bit/s 2 Gbit is 2 125 000 000 bit/s 4 Gbit is 4 250 000 000 bit/s) 10 Gbit single channel variants are 10 518 750 000 bit/s 10 Gbit four channel variants are 12 750 000 000 bit/s This is the raw bit rate. This property is OPTIONAL.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.3.13.12.2 ProtocolControllerForPort

Table 106 - Required Properties from ProtocolControllerForPort

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Antecedent	ref	key	The SCSIProtocolController for this port.
Dependent	ref	key	The port.
AccessPriority	unit16		The priority of access through this port for this ProtocolController (optional)ProtocolControllerForPortl (optional).

7.3.3.13.12.3 ProtocolControllerAccessesUnit

Table 107 - Required Properties from ProtocolControllerAccessesUnit

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Antecedent	ref	key	The protocol controller.
Dependent	ref	key	The exposed logical unit.
DeviceNumber	unit16		Logical Unit Number. Formatted as unseparated uppercase hexadecimal digits, with no leading “0x”.
TargetControllerNumber			TargetID This property is OPTIONAL.

7.3.3.13.12.4 SCSIProtocolController

Table 108 - Required Properties for SCSIProtocolController

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
SystemCreationClassName	string	key	
SystemName	string	key	
CreationClassName	string	key	
ElementName	string		User friendly name/caption for port. This property is OPTIONAL.
OperationalStatus[ ]	uint16		Status of device This property is OPTIONAL.
DeviceID	string	key	Opaque.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.3.13.12.5 StorageExtent

Table 109 - Required Properties for StorageExtent

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
SystemCreationClassName	string	maxlen(256), key	The scoping System's CreationClassName.
SystemName	string	maxlen(256), key	The scoping System's Name.
CreationClassName	string	maxlen(256), key	The name of the concrete subclass.
DeviceID	string	maxlen(64), key	Unique identifying information.
BlockSize	uint64		
NumberOfBlocks	uint64		
ConsumableBlocks	uint64		
ExtentStatus[ ]	uint16		
NoSinglePointOfFailure	boolean		
DataRedundancy	uint16		
PackageRedundancy	uint16	write(true)	
DeltaReservation	uint16		

7.3.3.13.12.6 SystemDevice

Table 110 - Required Properties for SystemDevice

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
GroupComponent	ref		System Reference.
PartComponent	ref		LogicalDevice Reference.

7.3.3.13.13 Optional subprofiles

Table 111 - Optional Profiles or Subprofiles

Name	Notes
None	

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.3.14 LUN Masking and Mapping

7.3.3.14.1 Description

> **NOTE** - In this version of the subprofile, all logical units are treated as read-write. Read-only permissions will be added in the next revision.

Many disk arrays provide an interface for the administrator to specify which initiators (Hosts or HBA Ports, by WWN) can access which volumes, through which target ports (by WWN). The effect is that the given volume is only visible to SCSI commands that originate from the specified initiators through specific sets of target ports. There may also be a capability to select the SCSI Logical Unit Number as seen by an initiator through a specific set of ports. The ability to limit access is called **Device Masking**; the ability to specify the device address seen by particular initiators is called **Device Mapping**. (For SCSI systems, these terms are known as LUN Masking and LUN Mapping.)

The model described here is generalized to include access management in disk arrays, virtualization systems and routers used in tape libraries. The model is also generalized beyond just SCSI and Fibre Channel implementations. Many of the examples and use cases refer to LUN masking in Fibre Channel arrays, but the model is general.

7.3.3.14.2 Standard dependencies

The LUN Masking and Mapping subprofile is based on the following standards:

Table 112 - LUN Masking Standard Dependencies

Standard	Version	Organization
CIM Specification	2.2	DMTF
CIM Operations over HTTP	1.1	DMTF
CIM Schema	2.8 Preliminary	DMTF

7.3.3.14.3 Profile dependencies

The LUN Masking and Mapping subprofile does not require any other Profiles.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.3.14.4 CIM Server requirements

7.3.3.14.4.1 Functional Profiles

Table 113 - Required Functional Profiles

Profile Required	Functional Group	Dependency
YES	Basic Read	None
YES	Basic Write	Basic Read
YES	Instance Manipulation	Basic Write
NO	Schema Manipulation	Instance Manipulation
YES	Association Traversal	Basic Read
NO	Query Execution	Basic Read
NO	Qualifier Declaration	Schema Manipulation
YES	Indication	None

7.3.3.14.4.2 Extrinsic methods

The CIM Server shall support extrinsic methods for the LUN Masking and Mapping subprofile.

7.3.3.14.4.3 Discovery

The LUN Masking and Mapping subprofile, as currently defined, is not an advertised subprofile. Support for it can be discovered through the Server profile.

7.3.3.14.5 Instance diagrams

7.3.3.14.5.1 Overview

Given a storage system with no LUN masking or mapping, all hosts/initiators see the same elements when they discover a storage system. LUN masking and mapping interfaces allow an administrator to customize the “view” of elements that are discovered. The effect is that the real storage system appears to be a number of smaller virtual storage systems, each virtual storage system exposing a view customized for a particular set of initiators.

The management model is built on these “views” of a storage system, each view is a subset of components the administrator exposes to certain hosts and the classes that model the authorization and access rights.

The model uses three basic types of objects:

- **LogicalDevice**, the superclass of volumes and tape drives
- **ProtocolController,** the superclass for controllers of various protocols, models the “view” described above.
- **LogicalPort,** the superclass of target ports for various transports.

These objects are related by two associations:

- **ProtocolControllerForUnit** associates a ProtocolController with its LogicalDevices. The controller-relative address (such as a SCSI Logical Unit Number) is modelled as the DeviceNumber property of ProtocolControllerForUnit.
- **ProtocolControllerForPort** associates a controller to one or more LogicalPorts.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.3.14.5.2 Protocol Controllers

Each ProtocolController is an implied namespace (or 'view") for LogicalDevices. The ProtocolControllerForUnit.DeviceNumber property represents the name of the LogicalDevice in the ProtocolController's namespace. For SCSI protocol storage, the name within this namespace is the Logical Unit Number.

In this subprofile, the existence of a ControllerConfigurationService with a ConcreteDependency association to a ProtocolController governs the high-level device masking and mapping policy for that protocol controller.

- If the service does not exist, then regardless of host port, the policy is that ProtocolControllerForPort connects a ProtocolController associated to a LogicalPort.
- If it is present, then for a particular host port, the policy is that ProtocolControllerForPort connects a ProtocolController to a LogicalPort only when access is explicitly granted to the ProtocolController.

Figure 44 and Figure 45 depict an instance diagram of a generic storage system with dual-port access to four logical devices. Figure 44 depicts an implementation with no device mapping and masking services. All of the LogicalDevices are exposed to all initiators with the same DeviceNumber.

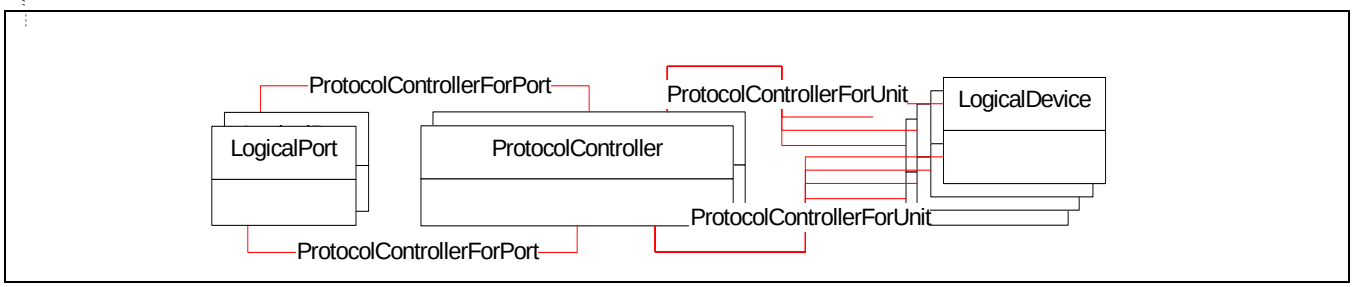


Figure 44 - Generic System with no ConfigurationService

Figure 45 depicts the same configuration in an implementation with an ControllerConfigurationService defined. In this case, access to the ProtocolController is denied to each host port unless it is specifically granted access. The means to grant access is discussed in 7.3.3.14.5.4, "Authorization".

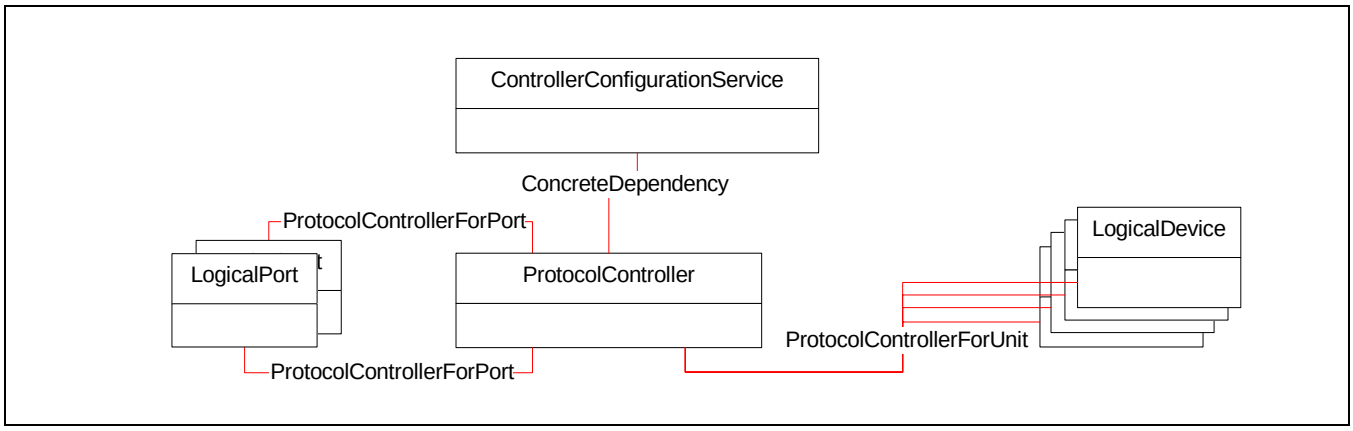


Figure 45 - Generic System with ControllerConfiguration Service

7.3.3.14.5.3 ProtocolController Views

Device Masking limits the devices seen by particular host HBAs. For example, some hosts may see two of four LogicalDevices, other hosts may see no LogicalDevices and yet other hosts may only see LogicalDevices through a subset of target ports.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Device Mapping allows the same LogicalDevice to be assigned different DeviceNumber (LUN) as seen by different host HBAs. This would allow each of four LogicalDevices to appear to be Logical Unit zero to four different hosts.

An initiator sees a single view (ProtocolController) through a target port. This view includes LogicalDevices explicitly exposed to specified initiators and “promiscuous” LogicalDevices (that are exposed to all initiators).

An administrator can use the ControllerConfigurationService interfaces to create “views” (SCSIProtocolControllers) of a storage system – each view exposes a subset of components that are intended to behave as a cohesive virtual storage system. In particular,

a) a view:
   1) is associated with a set of LogicalDevices;
   2) may be exposed to zero or more host ports;
   3) is associated with one or more target device ports;
   4) shall not be exposed through a particular host/target port pair that is in use by another view. (In other words, a view corresponds to the logical unit inventory provided by SCSI REPORT LUNS and INQUIRY commands.

b) each LogicalDevice in a view shall have a unique DeviceNumber (Logical Unit).

c) a LogicalDevice may be in multiple views and in each may be assigned the same or different DeviceNumbers (Logical Units);

A view shall conform to the protocol requirements of the ProtocolController it virtualizes. For instance, if it models SCSI hardware, then the Device Numbers assigned to devices shall conform to the Logical Unit assignment rules of SCSI.

7.3.3.14.5.4 Authorization

The array uses the Port WWN to authorize access and to determine the view to present to the HBA. The Port WWN is modelled as a subclass of Identity (part of the User and Security common model) called StorageHardwareID. As used in this subprofile, AuthorizedSubject associates an AuthorizedPrivilege with a StorageHardwareID or a SystemSpecificCollection of StorageHardwareIDs. As used in this subprofile, AuthorizedTarget associates an AuthorizedPrivilege with a ProtocolController (view).

An implementation is not required to model the SystemSpecificCollection. However, for hardware implementations that allow customers to name initiator collections, this collection provides a way to model that name. Clients should be able to operate with implementations that include or omit the SystemSpecificCollection. ProtocolControllerMaskingCapabilities.ProtocolControllerSupportsCollections tells a client whether the profile supports SystemSpecificCollections of StorageHardwareIDs. Throughout this

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

subclause, the term “subject” refers to either a specific StorageHardwareID or a collection of StorageHardwareIDs, as appropriate.

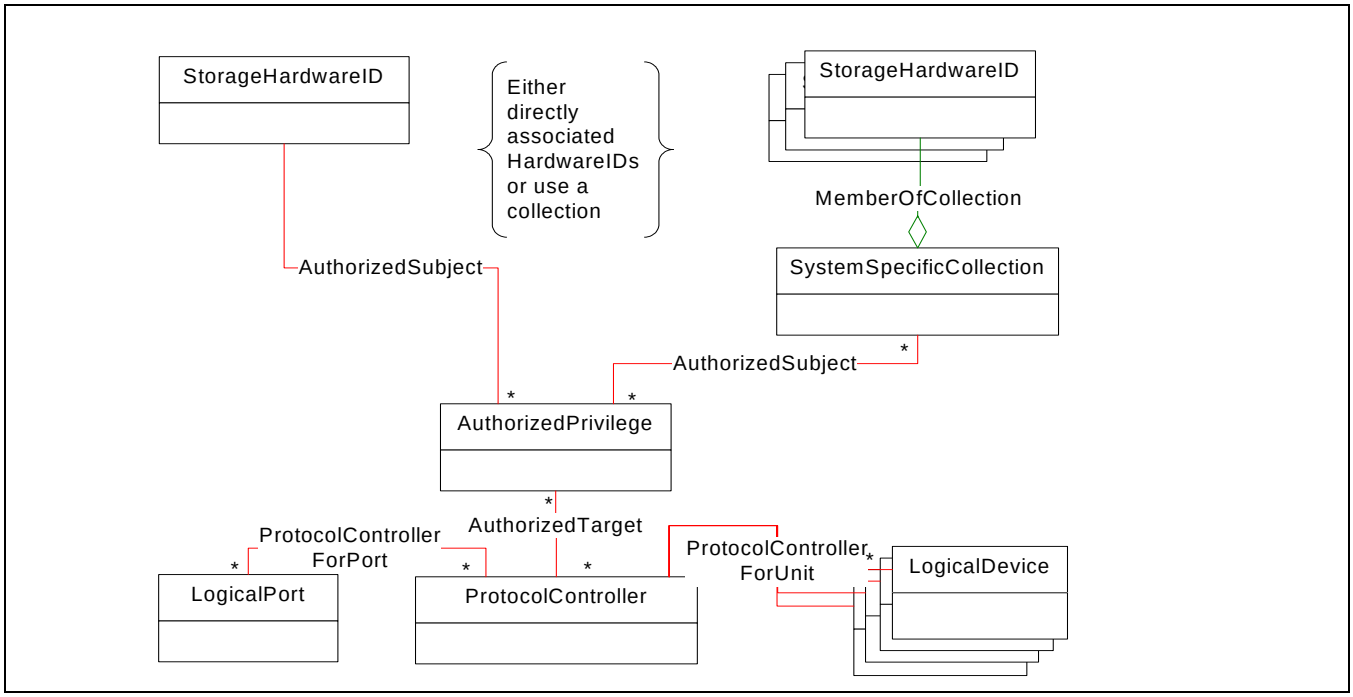


Figure 46 - Authorization and Access Rights

7.3.3.14.5.5 StorageClientSettingData

Some storage systems allow a customer (or host-side agent) to provide information about OS hosting initiators. The storage system uses this information to provide OS-specialized behavior (for example, SCSI responses). This information is modelled as StorageClientSettingData. StorageClientSettingData.ClientTypes[ ] is an array of OS names. This array property allows a single StorageClientSettingData instance to apply to multiple OS Types.

Where possible, the instrumentation will provide a meaningful name for each StorageClientSettingData instance. Typically, this will be names already exposed via existing management tools and documentation.

StorageClientSettingData instances are not created by clients. Any storage system that provides OS type behavior advertises these instances (via EnumerateInstance and GetInstance) and associates them (using ElementSettingData) with elements previously configured with the setting behavior.

A client can associate StorageHardwareIDs to a StorageClientSettingData instance (when a customer or host agent maps an initiator to an OS type). This is done by specifying the Setting parameter to CreateStorageHardwareID). A client can also associate an StorageClientSettingData instance to a storage

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

system element (such as a Port, a ProtocolController, or a Volume) to request that this element exhibit the setting-specific behavior.

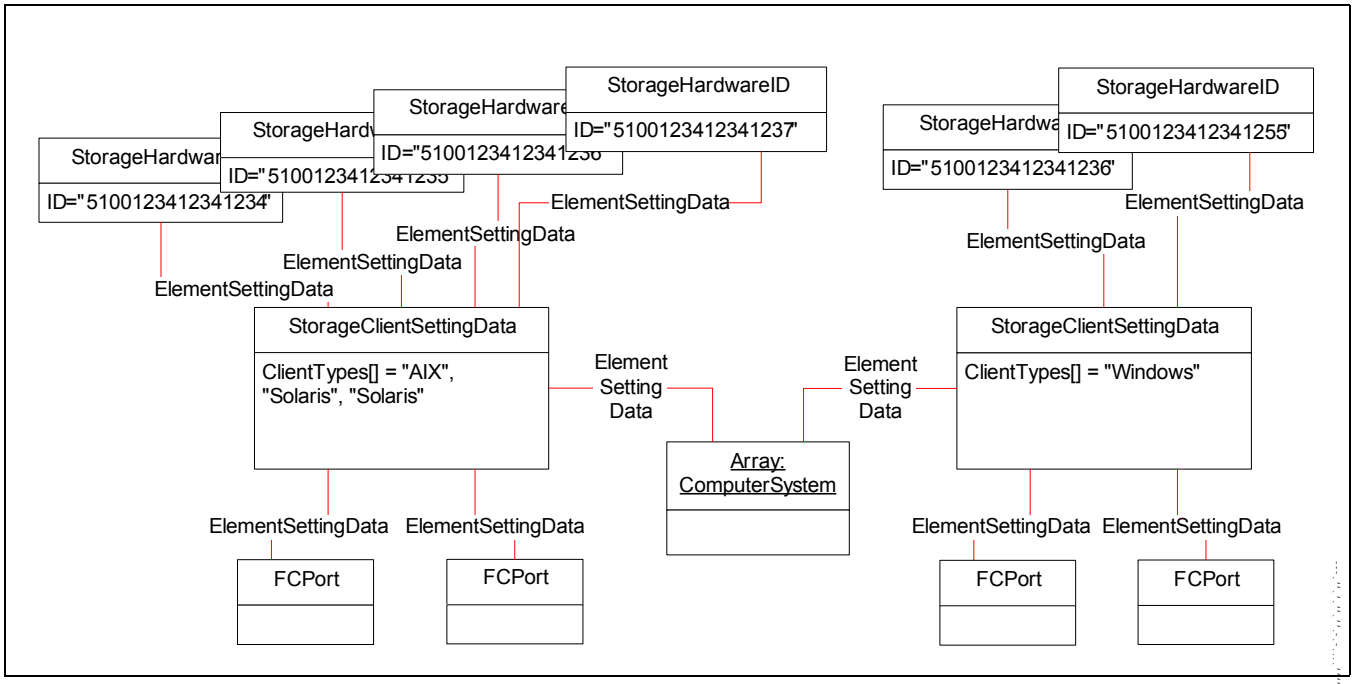


Figure 47 - Initiator Setting Data Example

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.3.3.14.5.6 The Entire Model

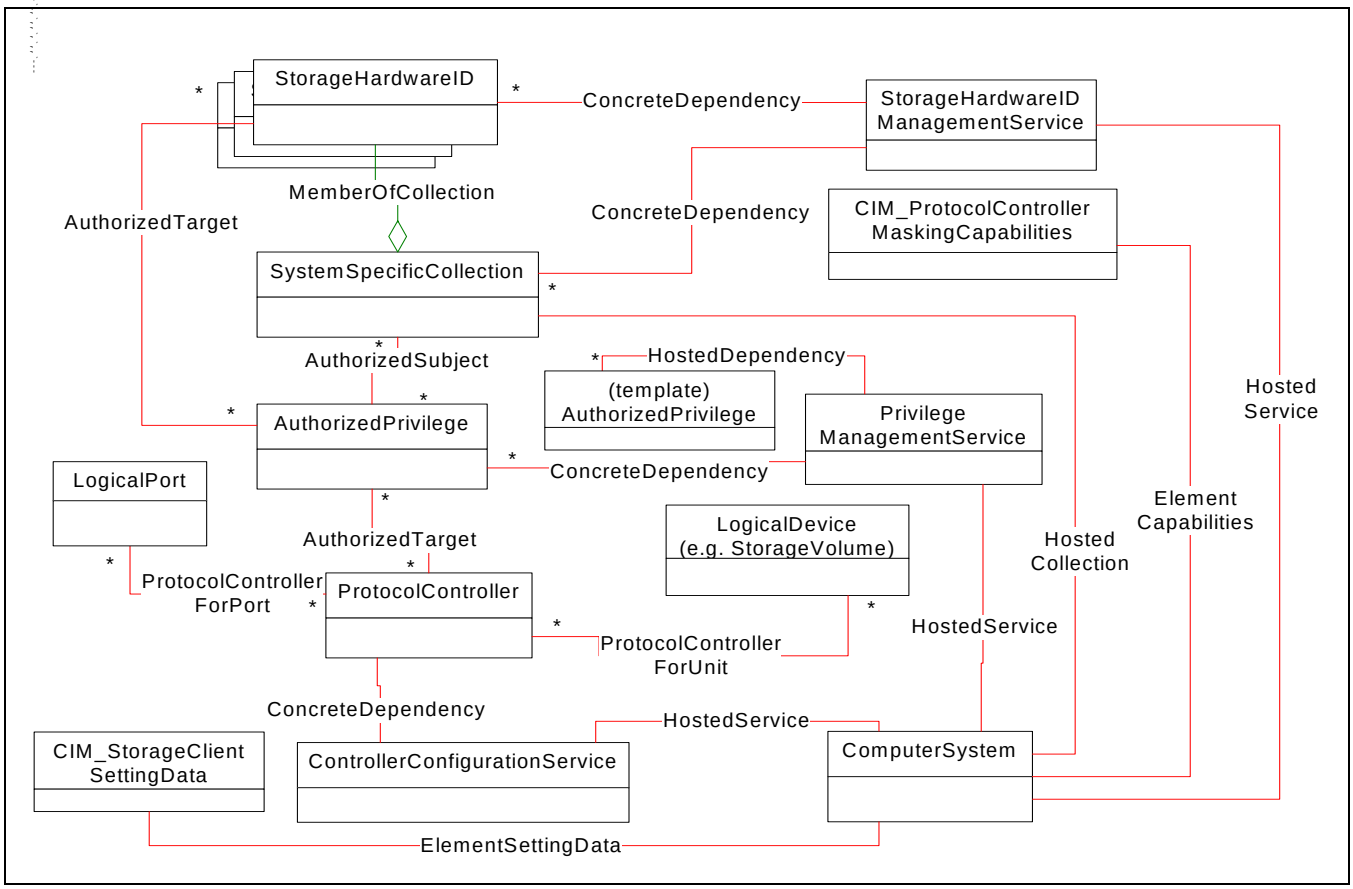


Figure 48 - Entire Model

7.3.3.14.6 Durable names and correlatable IDs

The LUN Masking and Mapping subprofile uses the durable names/correlatable IDs for ports and logical devices as defined by the parent profile. In addition, the subprofile adds the following durable name:

- **StorageHardwareID** - StorageHardwareID.StorageID and the StorageHardwareID.IDType. The valid IDTypes are:
  - "PortWWN" - 16 unseparated upper case hex digits
  - "NodeWWN" - 16 unseparated upper case hex digits
  - "Hostname" - 32 unseparated upper case hex digits. This corresponds to the Platform identifier and is defined in FC-GS specifications.

7.3.3.14.7 Methods

7.3.3.14.7.1 ControllerConfigurationService methods:

CreateProtocolControllerWithPorts creates a ProtocolController that is used as an AuthorizedTarget. If multiple target ports are passed in, all expose the same view (i.e., the same devices with the same unit numbers.) This method does not create the port instances, but does create ProtocolControllerForPort associations between the ports and the new ProtocolController. The new ProtocolController is weakly associated to the same System as the ControllerConfigurationService.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

SMI-S compliant implementations shall use the AuthorizedPrivilege subclass for the Privilege parameter. This parameter does not refer to a specific AuthorizedPrivilege instance that will be associated with the ProtocolController. Instead, a new AuthorizedPrivilege instance is created each time CreateProtocol-ControllerWithPorts is called, and the instance referenced in the parameter is used as a template. The Activities and PrivilegeGranted properties from the template are copied to the new AuthorizedPrivilege instance.

Implementations of this subprofile for SMI-S 1.0.x shall include a single template AuthorizedPrivilege instance with Activities set to Read, Write. This template instance is associated to the PrivilegeManagement Service with HostedDependency, the dynamic AuthorizedPrivilege instances created by CreateProtocolControllerWithPorts are associated to PrivilegeManagementService with ConcreteDependency.

```
Uint32 CreateProtocolControllerWithPorts(
            The string to be used in the ElementName of the new
            ProtocolController.
    string ElementName
            Array of strings containing representations of
            references to instances of CIM_LogicalPort (or subclass)
            instances. This is the list of target ports that are
            associated to the ProtocolController.
            ProtocolControllerForPort associations are created by
            the instrumentation associating the new
            ProtocolController to these ports. If this parameter is
            null, then all ports in the storage system (this
            Service's 'scoping' System and all its ComponentCS
            Systems) are attached to the new ProtocolController.
    string Ports []
            The protocol type for the new ProtocolController.
    Uint16 Protocol
            Reference to an AuthorizedPrivilege instance to be used
            as a template. If supplied, it has a 1:1 correspondence
            with the Identity parameter and this method will
            atomically create the new ProtocolController instance
            and related AuthorizedPrivilege, AuthorizedTarget and
            AuthorizedSubject instances. Note: if
            ProtocolControllerRequiresAuthorizedIdentity is true,
            then the Identity/AuthorizedPrivilege pair shall be
            specified. If false, then the Identity/
            AuthorizedPrivilege pair shall not be specified.
    CIM_Privilege REF Privilege
            Reference to either a StorageHardwareID or a
            SystemSpecificCollection that has StorageHardwareIDs as
            MemberOfCollection. If
            ProtocolControllerMaskingCapabilities.ProtocolController
            SupportsCollection is false, the reference shall be to a
            StorageHardwareID. If present, it has a 1:1
            correspondence with the Privilege property. If present,
            this method will automically create the new
            ProtocolController instance and instantiate a missing
            AuthorizedSubject association between the
            AuthorizedPrivilege/Identity pair; and instantiate an
            AuthorizedTarget association between the
            AuthorizedPrivilege and the new ProtocolController. Note
            that if ProtocolControllerRequiresAuthorizedIdentity is
            true, then at least the Identity/AuthorizedPrivilege
            pair shall be specified.
    CIM_ManagedElement REF Identity
```

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
                    A reference to the new ProtocolController that is
                    created.
          CIM_ProtocolController REF ProtocolController
)
```

DeleteProtocolController deletes the ProtocolController and all associations connected directly to this ProtocolController. If the DeleteChildrenProtocolControllers parameter is True, the provider also deletes child ProtocolControllers (those at the dependent end of AssociatedProtocolController associations from this ProtocolController) plus all child ProtocolControllers' direct associations. If the DeleteLogicalUnits parameter is True, the provider also deletes LogicalDevice instances associated via ProtocolControllerForUnit to this ProtocolController and its children. LogicalDevice instances are only deleted when they are not part of any ProtocolControllerForUnit associations.

```
Uint32 DeleteProtocolController(
                    The ProtocolController to be deleted.
          CIM_ProtocolController REF ProtocolController
                    If true, the management instrumentation provider will
                    also delete 'child' ProtocolControllers (i.e., those
                    defined as Dependent references in instances of
                    AssociatedProtocolController where this
                    ProtocolController is the Antecedent reference). Also,
                    all direct associations involving the 'child'
                    ProtocolControllers will be removed.
          boolean DeleteChildrenProtocolControllers
                    If true, the management instrumentation provider will
                    also delete LogicalDevice instances associated via
                    ProtocolControllerForUnit, to this ProtocolController
                    and its children. (Note that 'child' controllers will
                    only be affected if the
                    DeleteChildrenProtocolControllers input parameter is
                    TRUE). LogicalDevice instances are only deleted if there
                    are NO remaining ProtocolControllerForUnit associations,
                    to other ProtocolControllers.
          boolean DeleteUnits
)
```

AttachDevice associates a LogicalDevice subclass to a ProtocolController. The provider shall verify that unit numbers are unique for each initiator. When the ProtocolController is already part of an AuthorizedTarget association, the provider should update the access configuration in the underlying hardware when AttachDevice is called.

```
Uint32 AttachDevice(
                    The ProtocolController instance.
          CIM_ProtocolController REF ProtocolController
                    The LogicalDevice instance to attach.
          CIM_LogicalDevice REF Device
                    The number assigned to
                    ProtocolControllerForUnit.DeviceNumber (if supported by
                    the hardware). Hardware support is indicated by
                    StorageMaskingCapabilities.ClientSelectableDeviceNumbers
                    ). If the hardware does not support setting the number,
                    but the DeviceNumber has not been established in an
                    existing ProtocolControllerForDevice subclass, then this
                    parameter's value will be used. If the DeviceNumber has
                    been established, then the current number will be
                    reused.
          string DeviceNumber
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
)
```

DetachDevice removes the ProtocolControllerForDevice association subclass between the ProtocolController and device.

```
Uint32 DetachDevice(
                The ProtocolController instance.
     CIM_ProtocolController REF ProtocolController
                The LogicalDevice instance to detach.
     CIM_LogicalDevice REF Device
)
```

7.3.3.14.7.2 PrivilegeManagementService methods

AssignAccess associates a subject ManagedElement (either a StorageHardwareID or a SystemSPecificCollection containing StorageHardwareID members) and a target ManagedElement (ProtocolController). The Privilege parameter shall be a reference to a template AuthorizedPrivilege instance as defined above for CreateProtocolControllerWithPorts().

```
Uint32 AssignAccess(
                The Subject parameter is a reference to a ManagedElement
                instance that will be associated via AuthorizedSubject
                to the AuthorizedPrivilege.
     CIM_ManagedElement REF Subject
                PrivilegesGranted shall be TRUE in SMI-S 1.0.
     boolean PrivilegeGranted
                Activities shall be “Read”, “Write” in SMI-S 1.0
     Uint16 Activities []
                ActivityQualifiers not used in SMI-S 1.0.
     string ActivityQualifiers []
                QualifierFormats not used in SMI-S 1.0
     Uint16 QualifierFormats []
                The Target parameter is a reference to a ManagedElement
                that will be associated via AuthorizedTarget to the
                AuthorizedPrivilege.
     CIM_ManagedElement REF Target
                 Reference to the AuthorizedPrivilege to be used as a
                template.
     CIM_AuthorizedPrivilege REF Privilege
)
```

RemoveAccess This method revokes all privileges or the specified AuthorizedPrivilege for a named target/ subject. If an AuthorizedPrivilege reference is supplied, then this method acts only on that specific AuthorizedPrivilege, otherwise, all AuthorizedPrivilege instances associated with the named subject/target are affected by the operation. If an AuthorizedPrivilege instance is left with no target associations, it is deleted.

```
Uint32 RemoveAccess(
                The Subject parameter is a reference to a ManagedElement
                instance for which privileges will be revoked.
     CIM_ManagedElement REF Subject
                A reference to the AuthorizedPrivilege to be revoked.
     CIM_Privilege REF Privilege
                The Target parameter is a reference to a ManagedElement
                associated via AuthorizedTarget to AuthorizedPrivilege
                instances.
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
        CIM_ManagedElement REF Target
    )
```

7.3.3.14.7.3 StorageHardwareIDManagementService methods:

CreateStorageHardwareID creates a StorageHardwareID and the association CIM_ConcreteDependency between this service and the new StorageHardwareID.

```
    Uint32 CreateStorageHardwareID(
                The ElementName of the new StorageHardwareID instance.
        string ElementName
                StorageID is the value used by the SecurityService to
                represent identity - in this case, a hardware worldwide
                unique name.
        string StorageID
                The type of the StorageID property.
        Uint16 IDType
                The type of the storage ID, when IDType is 'Other'.
        string OtherIDType
                REF to the StorageClientSettingData containing the
                OSType appropriate for this initiator. If left NULL, the
                instrumentation assumes a standard OSType - i.e., that
                no OS-specific behavior for this initiator is defined.
        CIM_StorageClientSettingData REF Setting
                REF to the new StorageHardwareID instance.
        CIM_StorageHardwareID REF HardwareID
    )
```

DeleteStorageHardwareID deletes a StorageHardwareID and the ConcreteDependency association between the ID and the service.

```
    Uint32 DeleteStorageHardwareID(
        CIM_StorageHardwareID REF HardwareID
    )
```

7.3.3.14.8 Client Considerations

7.3.3.14.8.1 Client Discovery Algorithms

a) Determining through which target ports a view is exposed

Follow the ProtocolControllerForPort associations from a ProtocolController to Ports.

b) Determining which LogicalDevices are exposed to a subject

Follow the AuthorizedSubject associations to all Privileges Follow AuthorizedTarget associations to a ProtocolController, all LogicalDevices connected via ProtocolControllerForUnit are exposed.

c) Finding the next available SCSI Logical Unit Number

Use the algorithm above to determine exposed LogicalDevices and create a list of ProtocolControllerForUnit.DeviceNumbers. DeviceNumbers missing from this list (up to ProtocolController.MaxUnitsControlled) are available.

d) Finding unexposed LogicalDevices

Enumerate appropriate subclasses of LogicalDevices (for example, StorageVolume) with SystemDevice associations to the appropriate ComputerSystem and follow ProtocolControllerForUnit associations. If a LogicalDevice has no ProtocolControllerForUnit association then it has not been exposed to a subject.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.3.14.8.2 Use Cases

7.3.3.14.8.2.1 Overview

The first few use cases show what a client would discover in certain configurations. The configuration consists of a single Fibre Channel port array with four volumes. If the array had no LUN masking in place, the basic components would be port, controller and volumes, as depicted in Figure 49

The notation ProtocolControllerForUnit(*n*) is used as a shorthand for a ProtocolControllerForUnit instance with DeviceNumber set to *n.*

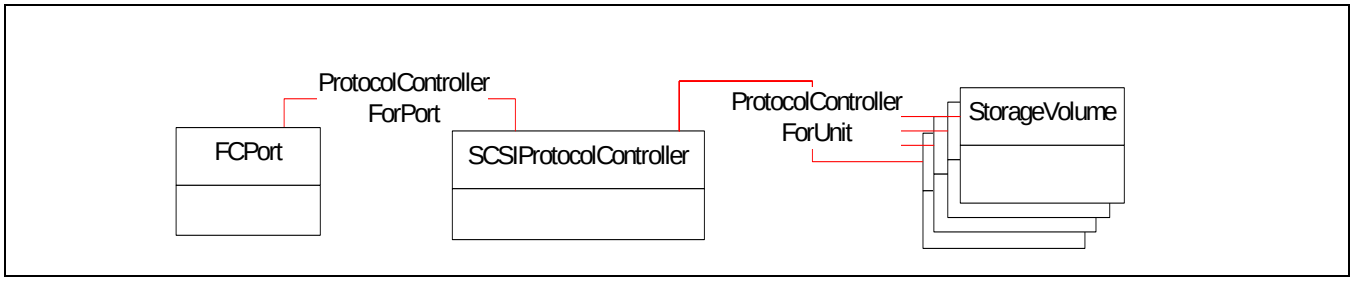


Figure 49 - Simple StorageVolume Model

7.3.3.14.8.2.2 Use Case 1 - Two Views, DeviceNumber Overlap

In Use Case 1, there are two views, each including two LogicalDevices exposed to two different HBAs. The HBA Port WWNs are properties of the StorageHardwareID instances. Either view exposes Logical Unit Numbers 0 and 1, but map to different volumes.

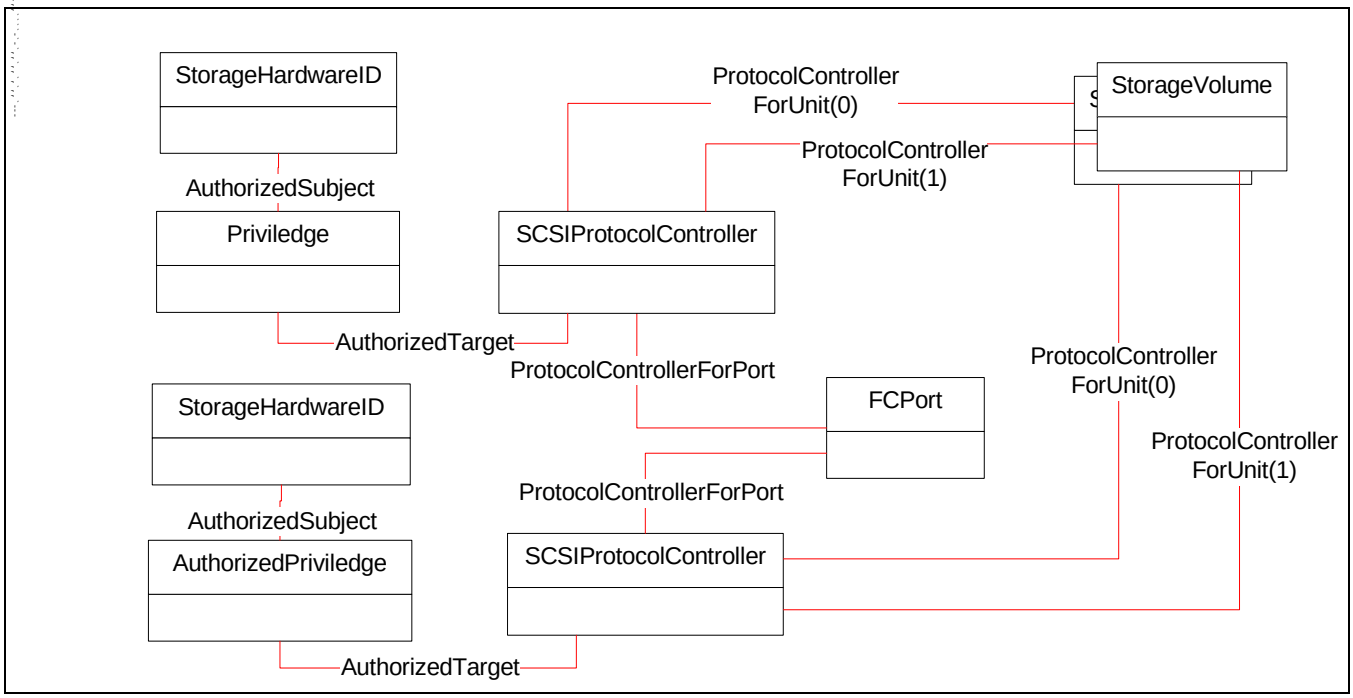


Figure 50 - Two view/Two LogicalDevice Use Case

7.3.3.14.8.2.3 Use Case 2 - Volume in multiple views

In this use case there are two initiators accessing three volumes. Initiator “A” accesses volumes 0 and 1 and initiator “B” accesses volumes 0, 1 and 2. There are two Views - one for each unique access combination. Volumes 0 and 1 have multiple ProtocolControllerForUnit associations. Note that there can be more than one initiator associated to the AuthorizedPrivilege object, but all those initiators access the same set of volumes.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

The fact that initiator “B” wanted access to a different set of volumes than initiator “A” resulted in the need for another AuthorizedPrivilege instance.

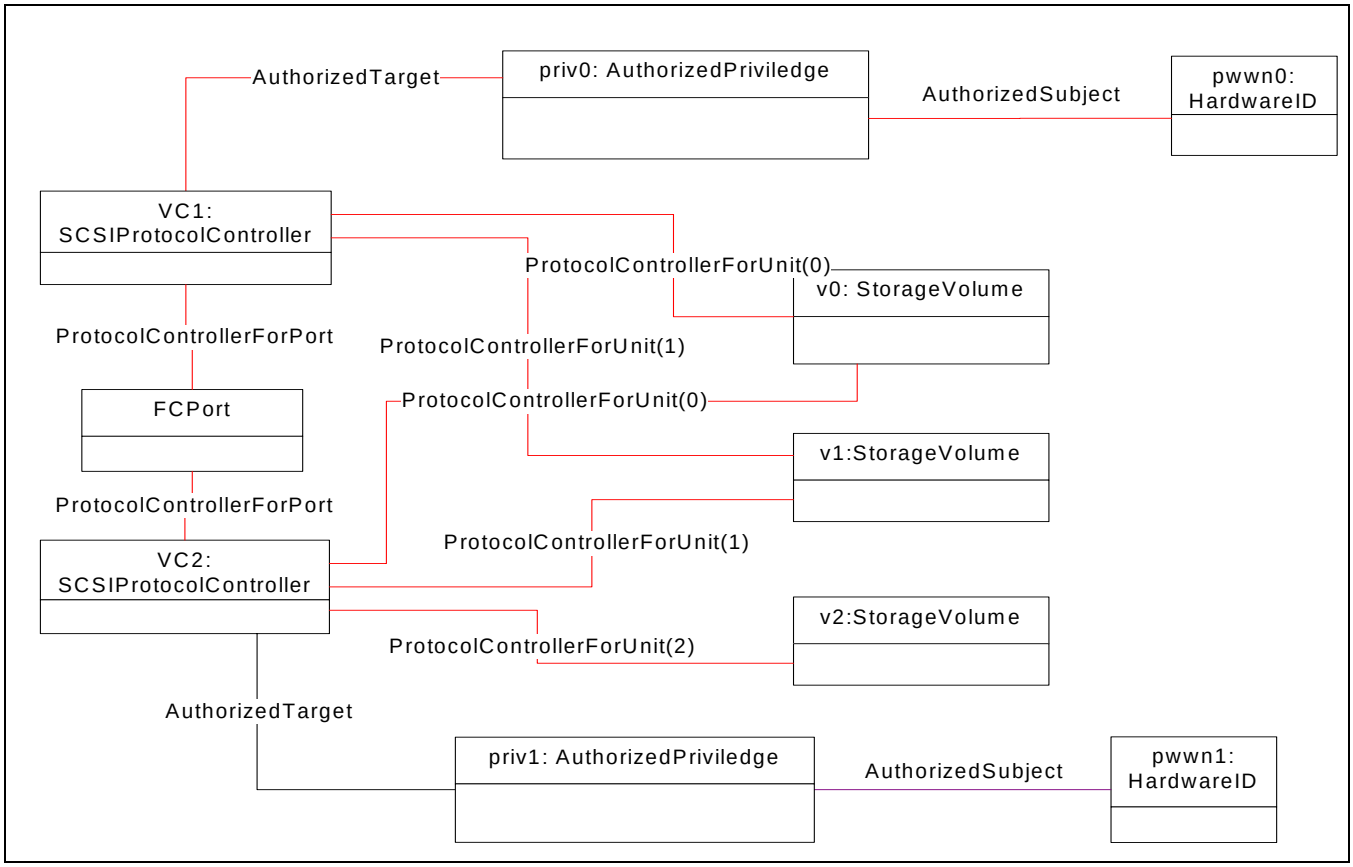


Figure 51 - Volume used in multiple views

7.3.3.14.9 Recipes

7.3.3.14.9.1 Recipes for General Functions

7.3.3.14.9.1.1 Get a ControllerConfigurationService CIMObjectPath for a StorageSystem

```
// DESCRIPTION
// Get a ControllerConfigurationService CIMObjectPath for a
// StorageSystem
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
// 1. The Storage System of interest has previously been identified
// and defined in the $StorageSystem-> variable.

// Step 1: Get the ControllerConfigurationService that is associated to
// the Storage System.
$ControllerConfigurationService->[] = AssociatorNames(
     $StorageSystem->,
     “CIM_HostedService”,
     “CIM_ControllerConfigurationService”,
     null,
     null);
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
// Assume one ControllerConfigurationService per storage system
$ControllerConfigurationService-> = $ControllerConfigurationService-
                            >[0];
```

7.3.3.14.9.1.2 Determining the capabilities for this array

```
// DESCRIPTION
// Since arrays have different capabilities,
                            ProtocolControllerMaskingCapabilities can be
// used to determine features supported. A client can then use that
// information to adjust the way it does LUN Masking to make the best
// use of the array's features.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
// 1. The CIM Object Path for the Storage system (StorageSystem) of
// interest has been identified and defined in $StorageSystem->

$Capabilities[] = Associators(
     $StorageSystem->,
     "CIM_ElementCapabilities",
     "CIM_ProtocolControllerMaskingCapabilities",
     "ManagedElement",// source object role
     "Capabilities") // resulting object role
$TheCapabilities = $Capabilities[0]

// Current values uint16 values 2, 3, 4, which map to
// "Port WWN", "Node WWN", and "Host Name"
// Only 2 ("Port WWN") is used at present in the recipes below
#HardwareIdTypes[] =
     $TheCapabilities.GetProperty("ValidHardwareIdTypes");

// Possible values: true, false
// If true, the storage system allows the client to specify the
// DeviceNumber parameter when calling AttachDevice() on
// ProtocolController instances.
// If false, the implementation will ignore the DeviceNumber value
#ClientSelectableDeviceNumbers =
     $TheCapabilities.GetProperty("ClientSelectableDeviceNumbers")

// Determine Ports per View capability
// Possible values:
// 2 = One Port per View
// 3 = Multiple Ports per View
// 4 = All Ports share the same View
#PortsPerView =
     $TheCapabilities.GetProperty("PortsPerView")

// Possible values: true, false
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
// Set to true if this storage system limits configurations to
// a single subject hardware ID per view. Otherwise, multiple hardware
// ID types can be used. The default is FALSE, that is multiple ID types
// may be used in a single view.
#OneHardwareIDPerView =
     $TheCapabilities.GetProperty("OneHardwareIDPerView")

// Possible values: true, false
// Set to true if this storage system supports the AttachDevice method.
#AttachDeviceSupported =
     $TheCapabilities.GetProperty("AttachDeviceSupported")

// Possible values: true, false
// When set to false, different ProtocolContollers attached
// to a LogicalPort can expose the same unit numbers. If true,
// then this storage system requires unique unit numbers across all
// the ProtocolControllers connected to a LogicalPort.
#UniqueUnitNumbersPerPort =
     $TheCapabilities.GetProperty("UniqueUnitNumbersPerPort")

// Possible values: true, false
// If true, this property indicates that the Identity parameter of
// CreateProtocolControllerWithPorts() MUST contain a reference to
// a CIM_Collection (or subclass) or to a CIM_Identity (or subclass).
#ProtocolControllerSupportsCollections =

                          $TheCapabilities.GetProperty("ProtocolContro
                          llerSupportsCollections")

// Possible values: true, false
// If true, this property indicates that a Privilege/Identity pair MUST
                          be
// specified when CreateProtocolControllerWithPorts() is called.
// If false, this property indicates that the Privilege/Identity pair in
// CreateProtocolControllerWithPorts() MUST NOT be set.
#ProtocolControllerRequiresAuthorizedIdentity =

                          $TheCapabilities.GetProperty("ProtocolContro
                          llerRequiresAuthorizedIdentity")
```

7.3.3.14.9.1.3 Finding a Service

```
// DESCRIPTION
// The following is used to find the proper service associated with an
// object (usually a ComputerSystem). It returns the object path of the
// service so the client can invoke its methods.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
// 1. there is only one service of a given name for an object. Since
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
// Service.Name is one of the keys, this is a reasonable assumption.

sub CIMObjectPath FindService(
     CIMObjectPath $AnObject->,
     string #AServiceName)
{
     CIMObjectPath $theService->
     // Find all services with this name that are associated to this
     // object via the HostedService association
     $ServiceList->[] = AssociatorNames(
        $AnObject->,
        “CIM_HostedService”, // Association Class
        #AServiceName, // Class name to find
        “Antecedent”, // Role
        “Dependent”) // ResultRole
     if ($ServiceList->[].length > 0)
        $theService-> = $ServiceList->[0]
     return $theService->
}
```

7.3.3.14.9.2 User Task Recipes

7.3.3.14.9.2.1 Define a view (ProtocolController)

```
// DESCRIPTION
// Define a permissions view (ProtocolController)
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
// 1.The name of the ProtocolController to create has previously been
//   decided to be named #ControllerName.
// 2.An array of target FCPorts to expose the view through has
//   previously been decided as $Port->[].
// 3.An array of target LogicalDevices to associate the view with has
//   previously been decided as $LogicalDevices->[].
// 3.1None of the instances of $LogicalDevices->[] may be assciated
//   to a ProtocolController.  Either delete the ProtocolController
//   or detach the device.
// 4.An array of device numbers for the LogicalDevices in this view,
//   which correspond with the values in $LogicalDevices->[], has
//   previously been decided as #deviceNumbers[].
// 5. If ProtocolControllerRequiresAuthorizedIdentity is true:
// 5.1 Then: If ProtocolControllerSupportsCollections is false or
//   OneHardwareIDPerView is true
// 5.1.1 Then: An Identity reference (StorageHardwareID ) to assign
                          access
//   in this view has previously been decided as #Identity.
// 5.1.2 Else: If ProtocolControllerSupportsCollections is false
// 5.1.2.1 Then: An Identity reference (StorageHardwareID ) to assign
//   access in this view has previously been decided as #Identity.
```

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
//   Note that subsequent Identities may be added via the
//   PrivilegeManagementService.AssignAccess() method.
// 5.1.2.2 Else: An Identity reference (StorageHardwareID or
//   SystemSpecificCollection) to assign access in this view has
//   previously been decided as #Identity
//   Note: if SystemSpecificCollection is used, it MUST be a
//   collection of StorageHardwareIDs
// 5.2 An AuthorizedPrivilege reference to specify authorization of
                          access
//   in this view has previously been decided as #Privilege.  This
//   AuthorizedPrivilege is used as a template.  It MUST be one of the
//   AuthorizedPrivileges associated to the PriviligeManagementService
//   via a HostedDependency association.  Note that the
//   AuthorizedPrivilege.PrivilegeGranted MUST be true.

// Step 1: Get the ControllerConfigurationService that is associated to
//   the Storage System.
@{Get a ControllerConfigurationService CIMObjectPath for a
                          StorageSystem}

// Step 2: Get the ProtocolController capabilities
// ProtocolControllerMaskingCapabilities.PortsPerView value stored in
                          #PortsPerView

@{Determining the capabilities for this array}
// Step 3: Parameter check: will the array support what we're trying to
                          do?
If (#PortsPerView == 1) { // Unknown
    < Error: Array does not support this action, unknown configuration >
    return;
}
Else if ((#PortsPerView == 2 || // One port per view
        #PortsPerView == 3) && // multiple ports per view
        $Port->[] == null) {
    < Error: Array does not support this configuration, at least one
                          port must be specified >
    return;
}
Else if ((#PortsPerView == 4) && // all ports per view
        $Port->[] != null) {
    < Error: Array does not support this configuration, no ports should
                          be specified >
return;
}

// Step 4: Use the ControllerConfigurationService's
//   'CreateProtocolControllerWithPorts' method to create the view
                          (i.e.
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
//   to create the ProtocolController object). Pass in the desired
                          Ports
//   and Controller name for the view as arguments to the extrinsic
//   'CreateProtocolControllerWithPorts' method.
%InArguments["ports"] = $Port->[];
%InArguments["name"] = #ControllerName;
%InArguments["Protocol"] = 2;
If (#ProtocolControllerRequiresAuthorizedIdentity == true) {
// Need to pass Identity/Privilege
%InArguments["Identity"] = #Identity ;
%InArguments["Privilege"] = #Privilege;
}invokeMethod(
     $ControllerConfigurationService->,
     "CreateProtocolControllerWithPorts",
     %InArguments[ ],
     %OutArguments[ ]);
$ProtocolController-> = $OutArguments["ProtocolController"];

// Step 5: Iterate through the LogicalDevices that will be exposed
//   through this view.
for $i in $LogicalDevices->[ ]
{
     // Step 3.1: Attach the LogicalDevice to the view by invoking
     // the ControllerConfigurationService's 'AttachDevice'
     // extrinsic method. Pass in as arguments to the method:
     // the CIMObjectPath reference to the ProtocolController
     // (view), the CIMObjectPath reference to the LogicalDevice,
     // and the 'DeviceNumber' String.
     $InArguments["ProtocolController"] = $ProtocolController->;
     %InArguments["LogicalDevice"] = $LogicalDevices->[#i];
     %InArguments["DeviceNumber"] = #deviceNumbers[i];
     invokeMethod(
        $ControllerConfigurationService->,
        "AttachDevice",
        %InArguments[ ],
        %OutArguments[ ]);
}
```

7.3.3.14.9.2.2 Remove a view (ProtocolController)

```
// DESCRIPTION
// Remove a permissions view
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
// 1. The Storage System of interest has previously been identified
// and defined in the $StorageSystem-> variable.
// 2. The CIMObjectPath of the Controller view to be removed is
// $controller->.
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
// Step 1: Get the CIM_ControllerConfigurationService that is
                              associated to
// the Storage System $StorageSystem->. The CIMObjectPath is stored in
// $ControllerConfigurationService->.
@{Get a ControllerConfigurationService CIMObjectPath for a
                              StorageSystem}

// Step 2: Invoke the ControllerConfigurationService's
// 'DeleteProtocolController' method to delete the view (i.e. to
// remove the ProtocolController object) specifying the reference to
// the ProtocolController as an input argument.
%InArguments["ProtocolController"] = $controller->
#ReturnValue = invokeMethod(
     $ControllerConfigurationService->,
     "DeleteProtocolController",
     %InArguments[],
     %OutArguments[]);
if (#ReturnValue != 0 && #ReturnValue != 4096) {
     <ERROR! Failed>
}
```

7.3.3.14.9.2.3 Define initiator authorization for a storage volume (read, read/write, none)

```
// DESCRIPTION
// This recipe will use the AuthorizationService to define which
// initiators (specified as Port HBA WWNs) are allowed access to one or
// more StorageVolumes connected to a ProtocolController. It assumes
// that the mapping of the StorageVolumes to ProtocolController has
// already been done.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
// 1.Namespace has been identified and defined in the $NameSpace
//   variable
// 2.The CIM Object Path for the Storage system (ComputerSystem) of
//   interest has been identified and defined in $StorageSystem->
// 3.The storage volumes to use have been identified and their CIM
//   Object Paths saved as $StorageVolume->[]
// 4.The SCSIProtocolControllers have been identified and their CIM
//   Object Paths saved as $ProtocolController->[]
// 5.The HBA Port WWNs have been identified and stored as strings in
//   the array variable #wwns[] (format is as specified in SMIS)

// Step 1: Locate the StorageHardwareIDService for this ComputerSystem
$StorageHardwareIDService-> = FindService(
     $StorageSystem->,
     "StorageHardwareIDService")
```

--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
// Step 2: Create Hardware IDs for each initiator WWN. This also creates
//   the association ConcreteDependency between this service and the
//   new StorageHardwareID.
For #x in #wwns[]
{
     %InArguments[“ID”] = #wwn[#x]
     %InArguments[“IDType”] = 2  // 2 = PortWWN format
     returnCode = InvokeMethod(
        $StorageHardwareIDService->,
        “CreateStorageHardwareID”,
        %InArguments[],
        %OutArguments[])
}

// Step 3: Build a list of CIM object paths for WWN StorageHardwareIDs
//   just created. This is not needed if CreateStorageHardwareID
//   returned StorageHardwareID object path
$StorageHardwareID[] = EnumInstances(“StorageHardwareID”, true)

// Build a set of just our WWNs
for #x in $StorageHardwareID[]
{
     hwID = GetProperty($StorageHardwareID[#x], “ID”)
     if (! contains(hwID, #wwns[]))
     {
        delete $StorageHardwareID[x]
     }
}

// An alternative to steps 2 and 3 is to create the StorageHardwareIDs
// and then call CreateHardwareIDCollection and
// AddHardwareIDsToCollection to associate the StorageHardwareIDs to
                           the
// collection via the MemberOfCollection association. The client can
// then use the SystemSpecificCollection instead of the
// StorageHardwareIDs array in the steps below.

// Step 4: Find the Controllers associated to the storage volume
$Controllers[] = Associators(
     $StorageVolume->,
     “ProtocolControllerForUnit”
     null,
     null,
     “Antecedent”,
     false,
     false,
     false,
     null)
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
// Step 5: Find the AuthorizationService for this ComputerSystem
$AuthorizationService-> = FindService(
     $StorageSystem->,
     “AuthorizationService”)

// Step 6: Create an AuthorizedPrivilege instance
$Privilege = newInstance(
     “AuthorizedPrivilege”,
     #NameSpace)
// Now set the permissions
$Privilege.setProperty(
     “Activities”,
     [‘Read’, ‘Write’])
$Privilege.setProperty(“PrivilegeGranted”, [True])
CreateInstance($Privilege)

// Step 7: Assign access - links Privilege to Controller and
//   StorageHardwareIDs (WWNs)
// We need to convert the object paths into strings for InvokeMethod()
// No way to pass array of REFs
Array #ControllersAsStrings[] = {}
for #x in $SCSIProtocolControllers->[]
{
     #ControllersAsStrings.add(string($SCSIProtocolControllers->[#x]))
}

for #x in $StorageHardwareID->[]
{
     %InArguments[“Subject”] = $StorageHardwareID->[#x])
     %InArguments[“AccessRights”] = $Privilege->
     %InArguments[“Target”] = #ControllersAsStrings[]
}

returnCode = InvokeMethod(
     $AuthorizationService->,
     “AssignAccess”,
     %InArguments[],
     %OutArguments[])
```

7.3.3.14.9.2.4 Remove specific authorization for an initiator

```
// DESCRIPTION
// This recipe shows how to remove access that was assigned in another
//
// PRE-EXISTING CONDITIONS AND ASSUMPTIONS
// 1.authorization has already been setup for the initiator in
//   question by the prior grant of access
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
// 2.The CIM Object Path for the Storage system (StorageSystem) of
//   interest has been identified and defined in $StorageSystem->
// 3.HBA Port WWN of the initiator is known and stored in $wwn

// Step 1: Find the CIM object path to the Initiator
$StorageHardwareID[] = EnumerateInstances(“StorageHardwareID”, true)

// Go through the list to find a match for our WWN
for #x in $StorageHardwareID[]
{
     #id = GetProperty($StorageHardwareID[x], “ID”)
     #idType  = GetProperty($StorageHardwareID[x], “IDType”)
     if (#idType = 2  && #id = #wwn)
     {
        // Save off the object path and we’re done
        $Initiator-> = $StorageHardwareID[x]->
        break;
     }
}

// Step 2: Get the authorization service
$AuthorizationService-> = FindService(
     $StorageSystem->,
     “AuthorizationService”)

// Step 3: Find the Privilege objects associated to this initiator
$Privileges[] = Associators(
     $Initiator->,
     “AuthorizedSubject”
     null,
     null,
     “Antecedent”,
     false,
     false,
     false,
     null)

// Step 4: We need to convert the object paths into strings for
//   InvokeMethod()
Array #PrivilegesAsStrings[] = {}
for #x in $Privileges[]
{
     #PrivilegesAsStrings.add(string($Privileges->[#x]))
}

// Step 5: Find Controllers associated to each initiator
Array #ControllersAsStrings[]
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
Array $TheseControllers[]
for #x in $Privileges[]
{
     $TheseControllers[] = Associators(
        $Privilege[#x],
        “AuthorizedTarget”,
        null,
        null,
        “Antecedent”,
        false,
        false,
        false,
        null)
     // Each Privilege could have multiple controllers associated
     // with it,so we iterate through the list
     for #y in $TheseControllers[]
     {
        #ControllersAsStrings.add(string($TheseControllers->[#y]))
     }
}

// Step 6: Remove access for that initiator
%InArguments[“Subject”] = $Initiator->
%InArguments[“Access”] = $PrivilegesAsStrings[]
%InArguments[“Target”] = #ControllersAsStrings[]
returnCode = InvokeMethod(
     $AuthorizationService->,
     “RemoveAccess”,
     %InArguments[],
     %OutArguments[])
```

7.3.3.14.10 Instrumentation Requirements

- The subject can either be a single StorageHardwareID or a collection of StorageHardwareIDs. If the underlying implementation supports initiator groups, this can be modelled with the collection. For implementations without hardware initiator groups, the agent can simulate them if the membership information can be persisted.
- If a PrivilegeManagementService is not present, then all access is assumed. If an PrivilegeManagementService is present, then access shall be specifically granted.
- A StorageHardwareID with Name set to "" is a wildcard that matches any name. This is a zero-length (empty) string.
- If a storage system supports wildcard access, it shall keep all ProtocolControllers with explicit StorageHardwareIDs up-to-date when wildcard access changes for the connected ports. For example, volume X is already exposed to one PortWWN when a client exposes volume Y through the same device port with a wildcard StorageHardwareID. The client would create a new ProtocolController with the wildcard as the authorized subject and would attach volume Y to this new ProtocolController. The instrumentation would also need to implicitly add ProtocolControllerForUnit between volume Y and the existing ProtocolController.
- A LogicalDevice may have ProtocolControllerForUnit associations to multiple ProtocolController. This models a device shared by different subject sets, as stated in Figure 51.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

- The view controller shall present a view consistent with the semantics of the protocol. For example, a SCSI implementation shall not overlap Logical Unit Numbers.
- Clients may need to know the range of possible unit numbers supported by a storage system. The agent should set ProtocolController.MaxUnitsControlled.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.3.14.11 Required CIM Elements

Table 114 - Subprofile Required Classes, Associations, Methods and Indications

Subprofile Class & Associations	Notes
AuthorizedSubject (7.3.3.14.12.2)	
AuthorizedTarget (7.3.3.14.12.3)	
ConcreteDependency (7.3.3.14.12.4)	
ControllerConfigurationService (7.3.3.14.12.5)	
ElementCapabilities (7.3.3.14.12.6)	
ElementSettingData (7.3.3.14.12.7)	
HostedCollection (7.3.3.14.12.8)	
HostedService (7.3.3.14.12.10)	
LogicalDevice Properties (7.3.3.14.12.11) (e.g., StorageVolume)	
LogicalPort Properties (7.3.3.14.12.12) (e.g., FCPort)	
MemberOfCollection (7.3.3.14.12.14)	OPTIONAL
AuthorizedPrivilege (7.3.3.14.12.1)	
PrivilegeManagementService (7.3.3.14.12.15)	
ProtocolController (7.3.3.14.12.16) (e.g., SCSIProtocolController)	
ProtocolControllerForPort (7.3.3.14.12.18)	OPTIONAL; mandatory for FCPorts.
ProtocolControllerForUnit (7.3.3.14.12.17)	
ProtocolControllerMaskingCapabilities (7.3.3.14.12.13)	
StorageClientSettingData (7.3.3.14.12.19)	
StorageHardwareID (7.3.3.14.12.20)	
StorageHardwareIDManagementService (7.3.3.14.12.21)	
SystemSpecificCollection (7.3.3.14.12.22)	
Profile Methods	Notes
CreateProtocolControllerWithPorts()	
DeleteProtocolController()	
AttachDevice()	
DetachDevice()	

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table 114 - Subprofile Required Classes, Associations, Methods and Indications (Continued)

Subprofile Class & Associations	Notes
AssignAccess()	
RemoveAccess()	
CreateStorageHardwareID	
DeleteStorageHardwareID	
CreateHardwareIDCollection	
AddHardwareIDsToCollection	
Intrinsic DeleteInstance is used to delete a SystemSpecificCollection of StorageHardwareIDs	
Intrinsic DeleteInstance is used to delete a MemeberOfCollection between a StorageHardwareID and SystemSpecificCollection. This causes the StorageHardwareID to be removed from the collection, but the StorageHardwareID itself is not deleted.	
Profile Indications	**Notes**
Creation of a Protocol Controller	SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_ProtocolControllerForUnit
Deletion of a Protocol Controller	SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_ProtocolControllerForUnit

7.3.3.14.12 Required Properties for CIM Elements

7.3.3.14.12.1 AuthorizedPrivilege

AuthorizedPrivilege defines the activities that are granted to an Identity (initiator StorageHardwareID).

The association of an Identity to AuthorizedPrivilege is accomplished using the AuthorizedSubject relationship. The entities that are protected are defined using the AuthorizedTarget relationship.

For SMI-S 1.0, the ActivityQualifiers and QualifierFormats properties are not used.

AuthorizedPrivilege is subclassed from ManagedElement.

Table 115 - Required Properties for AuthorizedPrivilege

Class Properties	Type	Qualifier/ Parameter	Notes
InstanceID	string	opaque, key	
ElementName	string		User Friendly name
PrivilegeGranted	boolean		For SMIS 1.0, shall be TRUE
Activities[ ]	uint16		For SMIS 1.0,, shall be “Read”, “Write”

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.3.14.12.2 AuthorizedSubject

AuthorizedSubject is an association used to tie specific Privileges to specific subjects.

Table 116 - Required Properties for AuthorizedSubject

Class Properties	Type	Qualifier/ Parameter	Notes
AuthorizedPrivilege	ref	override	The AuthorizedPrivilege affecting the target resource.
PrivilegedElement	ref	override	The Subject for which Privileges are granted or denied.

7.3.3.14.12.3 AuthorizedTarget

AuthorizedTarget is an association used to tie an Identity to specific target resources.

Table 117 - Required Properties for AuthorizedTarget

Class Properties	Type	Qualifier/ Parameter	Notes
AuthorizedPrivilege	ref		The AuthorizedPrivilege affecting the target resource.
TargetElement	ref		The target set of resources to which the AuthorizedPrivilege applies.

7.3.3.14.12.4 ConcreteDependency

CIM_ConcreteDependency is a generic association used to establish dependency relationships between ManagedElements.

Table 118 - Required Properties for ConcreteDependency

Class Properties	Type	Qualifier/ Parameter	Notes
Antecedent	ref	key	Antecedent represents the independent object in this association.
Dependent	ref	key	Dependent represents the object dependent on the Antecedent.

7.3.3.14.12.5 ControllerConfigurationService

ControllerConfigurationService provides methods for manipulating ProtocolControllers.

ControllerConfigurationService is subclassed from Service.

Table 119 - Required Properties for ControllerConfigurationService

Class Properties	Type	Qualifier/ Parameter	Notes
SystemCreationClassName	string	maxlen(256), key, propagated	The scoping System's CreationClassName.
SystemName	string	maxlen(256), key, propagated	The scoping System's Name.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table 119 - Required Properties for ControllerConfigurationService

Class Properties	Type	Qualifier/ Parameter	Notes
CreationClassName	string	maxlen(256), key	The name of the concrete subclass.
Name	string	maxlen(256), key, override	
CreateProtocolControllerWithPorts ()			
DeleteProtocolController()			
AttachDevice()			
DetachDevice()			

7.3.3.14.12.6 ElementCapabilities

The LUN Masking subprofile requires, but does alter ElementCapabilities.

7.3.3.14.12.7 ElementSettingData

ElementSettingData represents the association between ManagedElements and applicable setting data.

Table 120 - Required Properties for ElementSettingData

Class Properties	Type	Qualifier/ Parameter	Notes
ManagedElement	ref	key	The managed element.
SettingData	ref	key	The SettingData object associated with the element.

7.3.3.14.12.8 HostedCollection

Table 121 - Required Properties for HostedCollection

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Antecedent	ref	key, min(1), max(1)	ComputerSystem
Dependent	ref	key, weak	LogicalPortGroup

7.3.3.14.12.9 HostedDependency

CIM_ConcreteDependency is a generic association used to establish dependency relationships between ManagedElements.

Table 122 - Required Properties for HostedDependency

Class Properties	Type	Qualifier/ Parameter	Notes
Antecedent	ref	key	Antecedent represents the independent object in this association.
Dependent	ref	key	Dependent represents the object dependent on the Antecedent.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.3.14.12.10 HostedService

The LUN Masking subprofile requires, but does not alter HostedService.

7.3.3.14.12.11 LogicalDevice Properties

The LUN Masking subprofile requires, but does alter LogicalDevice.

7.3.3.14.12.12 LogicalPort Properties

LUN Masking requires, but does not alter LogicalPort.

7.3.3.14.12.13 ProtocolControllerMaskingCapabilities

A subclass of Capabilities that defines the Masking-related Capabilities of a storage system.

ProtocolControllerMaskingCapabilities is subclassed from Capabilities

Table 123 - Required Properties for ProtoclControllerMaskingCapabilities

Class Properties	Type	Qualifier/ Parameter	Notes
InstanceID	string	opaque, key	
ElementName	string	override, required	
ValidHardwareIdTypes	uint16		A list of the valid values for StrorageHardwareID.IDType. ValueMap {"2", "3", "4"}, Values {"PortWWN", "NodeWWN", "Hostname"}].
AccessControlByPorts	boolean		Not used in SMI-S 1.0.
ClientSelectableDeviceNumbers	boolean		Set to true if this storage system allows the client to specify the DeviceNumber parameter when calling AttachDevice() on ProtocolController instances. Set to false if the implementation does not allow unit numbers to vary across ProtocolController.
OneHardwareIDPerView	boolean		Set to true if this storage system limits configurations to a single subject hardware ID per view.
OnePortPerView	boolean		Not Used in SMI-S 1.0.
PortsPerView	uint16		An integer enumeration indicating the number of ports per view (ProtocolController) that the underlying storage system supports. ValueMap { "1", "2", "3", "4"}, Values { "Unknown", "One Port per View", "Multiple Ports per View", "All Ports share the same View"}.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table 123 - Required Properties for ProtoclControllerMaskingCapabilities

Class Properties	Type	Qualifier/ Parameter	Notes
ProtocolController RequiresAuthorized Identity	boolean		If true, this property indicates that an AuthorizedPrivilege/Identity pair shall be specified when CreateProtocolControllerWithPorts() is called. If false then the AuthorizedPrivilege/ Identity pair in CreateProtocolControllerWithPorts shall not be set.
PrivilegeDenied Supported	boolean		Set to true if this storage system allows a client to create an AuthorizedPrivilege instance with PrivilegeGranted set to FALSE. Shall be false in SMI-S 1.0.
UniqueUnitNumbers PerPort	boolean		When set to false, different ProtocolContollers attached to a LogicalPort can expose the same unit numbers. If true, then this storage system requires unique unit numbers across all the ProtocolControllers connected to a LogicalPort.
AttachDeviceSupported	boolean		Set to true if this storage system supports the AttachDevice method.
ProtocolControllerSupports Collections	boolean		If true, this property indicates that the Identity parameter of CreateProtocolConntrollerWithPorts() may contain a reference to a CIM_Collection (or subclass). Otherwise, that parameter shall contain a reference to a CIM_Identity (or subclass).

7.3.3.14.12.14 MemberOfCollection

The LUN Masking subprofile requires, but does not alter MemberOfCollection.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.3.14.12.15 PrivilegeManagementService

PrivilegeManagementService is subclassed from AuthorizationService.

Table 124 - Required Properties for PrivilegeManagementService

Class Properties	Type	Qualifier/ Parameter	Notes
ElementName	string		User Friendly name.
SystemCreationClassName	string	maxlen(256), key, propagated	The scoping System's CreationClass-Name.
SystemName	string	maxlen(256), key, propagated	The scoping System's Name.
CreationClassName	string	maxlen(256), key	The name of the concrete subclass.
Name	string	maxlen(256), key, override	
AssignAccess()			
RemoveAccess()			

7.3.3.14.12.16 ProtocolController

The LUN Masking subprofile requires, but does not alter ProtocolController.

7.3.3.14.12.17 ProtocolControllerForUnit

The LUN Masking subprofile requires, but does not alter ProtocolControllerForUnit.

7.3.3.14.12.18 ProtocolControllerForPort

The LUN Masking subprofile requires, but does not alter ProtocolControllerForPort.

7.3.3.14.12.19 StorageClientSettingData

This class models host environment that influence the behavior of Storage Systems. For example, a disk array has different SCSI responses for initiators configured as AIX verses HPUX. Instances of this setting class are associated via ElementSettingData to device Ports, ProtocolControllers or Volume instances when these elements have host awareness. These associations are created by the provider to reflect the current configuration. A client deletes/creates these associations to request changes in element host-awareness.

This setting class is also associated with StorageHardwareID instances when that HW ID is configured with host information.

An instance of this setting may include several ClientType values if the storage system treats them identically.

The storage system exposes all supported setting instances to an enumerate request; the client uses the returned settings to determine which types are available.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

StorageClientSettingData is subclassed from SettingData.

Table 125 - Required Properties for StorageClientSettingData

Class Properties	Type	Qualifier/ Parameter	Notes
InstanceID	string	key	Opaque provider-generated name.
ElementName	string		User Friendly Name.
ClientTypes[ ]	uint16	arraytype ("indexed")	These names map to operating system and host environment factors that influence the behavior exposed by storage systems. ValueMap {"0", “1”, “2”, “3”, “4”, “5”, “6”, “7”, “8”, “9”, “10”, “11”, “12”, “13”, “14”, “15”, “16”, “17”, “18”, "..", “0x8000..”}, Values {“Unknown”, “Other”, “Standard”, “Solaris”, “HPUX”, “Eponyms”, “Tru64”, “Knit-wear”, “Sequent”, “AIX”, “DGUX”, “Admix”, “Hyrax”, “Cystic iSCSI Storage Router”, “Linux”, “Microsoft Windows", “OS400”, “TRESPASS”, “HI-UX”, “DMTF Reserved”, “Vendor Specific”}.
OtherClientTypeDescriptions [ ]	string	arraytype ("indexed")	A string describing the manufacturer and OS/Environment, used when the Initiator-Types, includes 'Other'.

7.3.3.14.12.20 StorageHardwareID

StorageHardwareID is a hardware ID that serves as an authorization subject.

StorageHardwareID subclasses from Identity.

Table 126 - Required Properties for StorageHardwareID

Class Properties	Type	Qualifier/ Parameter	Notes
InstanceID	string	key	
StorageID	string	required	The hardware worldwide unique ID.
IDType	uint16	required	The type of the ID property. ValueMap {“2”, “3”, “4”}, Values {“PortWWN”, “NodeWWN”, “Hostname”}.

7.3.3.14.12.21 StorageHardwareIDManagementService

StorageHardwareIDManagementService provides methods for creating StorageHardwareIDs.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

StorageHardwareIDManagementService is subclassed from AuthenticationService.

Table 127 - Required Properties for StorageHardwareIDManagementService

Class Properties	Type	Qualifier/ Parameter	Notes
SystemCreationClassName	string	maxlen(256), key, propagated	The scoping System's Creation-ClassName.
SystemName	string	maxlen(256), key, propagated	The scoping System's Name.
CreationClassName	string	maxlen(256), key	The name of the concrete subclass.
Name	string	maxlen(256), key, override	
CreateStorageHardwareID ()			
DeleteStorageHardwareID()			

7.3.3.14.12.22 SystemSpecificCollection

SystemSpecificCollection represents the general concept of a collection that is scoped (or contained) by a System. It represents a Collection that only has meaning in the context of a System and/or whose elements are restricted by the definition of the System. This is explicitly described by the (required) association, HostedCollection.

In the context of LUN Masking and Mapping the collection is a collection of StorageHardwareIDs.

SystemSpecificCollection is subclassed from Collection.

Table 128 - Required Properties for SystemSpecificCollection

Class Properties	Type	Qualifier/ Parameter	Notes
ElementName	string		User Friendly name.
InstanceID	string	opaque	

7.3.3.14.13 Optional subprofiles and profiles

There are no optional subprofiles or profiles for this subprofile.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.4 Fabric

7.3.4.1 Fabric Profile

7.3.4.1.1 Description

7.3.4.1.1.1 SANS and Fabrics as AdminDomains

A SAN and Fabric are represented in CIM by AdminDomain. A SAN contains one or more Fabrics, which are modelled as AdminDomains. The “containment” of Fabrics to SANs is through the association ContainedDomain. AdminDomain is subclassed from System. This is significant because a SAN and a Fabric can be considered a group of components that operate together as a single system and are managed as such. The relationship of the Fabrics in a SAN could be as redundant fabrics, interconnected (using the same or different transport/protocols) or not connected in any way. Even in the latter case where the Fabrics are disjoint, from an administrative perspective they may still be managed together applying common practices including naming across the Fabrics.

An AdminDomain in CIM is keyed by the property Name with an associated property NameFormat. Typically SANs are identified (“named”) administratively and precise naming conventions are left up to the implementation, which is then responsible for assuring that the names are unique within the discovery of known SANs that populate the same CIM Namespace.

For Fibre Channel Fabrics, the identifier is the Fabric WWN that is based on the principal switch and the NameFormat should indicate that it is a WWN.

7.3.4.1.1.2 Fabrics and Topology

A Fabric in CIM minimally contains a ConnectivityCollection and its component systems. They are associated to the Fabric by the association Component. For the purposes of this discussion, it is assumed one models both.

ConnectivityCollection represents the foundation necessary for routing (and the reason it is defined in the Network model). A ConnectivityCollection groups a set of ProtocolEndpoints together that are able to communicate with each other directly. The ProtocolEndpoint is associated to the ConnectivityCollection by MemberOfCollection. A link is represented by the association ActiveCollection, which associates two ProtocolEndpoints, defined as a connection that is carrying traffic or is configured to carry traffic.

It is important at this point to clarify the relationship (or use) of the ProtocolEndpoint versus the use of FCPort (discussed later). A NetworkPort (from which FCPort is subclassed) is the device that is used to represent the logical aspects of the link and data layers. The ProtocolEndpoint is used to represent the higher network layers for routing. This is best understood when thinking about ethernet and IP, but apply to fibre channel also. When two ProtocolEndpoints are capable of communicating, the association ActiveConnection is used to represent the capability to communicate, and it completes the picture of the topology.

One can ultimately represent multiple ConnectivityCollection (for example, FC, IP (over FC) and IP (FC encapsulated in IP)) for the same fibre channel fabric.

Note that in modelling SANs, Fabrics and ConnectivityCollections, a ConnectivityCollection does not require a Fabric, and a Fabric does not require a SAN. But a SAN requires a Fabric, and a Fabric (for the purposes of this profile) requires a ConnectivityCollection.

The minimum set of requirements for this profile is based on ISO/IEC 14165-414 (FC-GS-4).

7.3.4.1.1.3 Systems and NetworkPorts

As discussed in 7.3.4.1.1.2, a Port is associated to a device to represent the link layer. A NetworkPort is associated to the ProtocolEndpoint by DeviceSAPImplementation and “joins” the System and Device model to the Network model. Instantiation of DeviceSAPImplementation, ProtocolEndpoint and ActiveConnection is not

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

necessary if the transceiver is not installed or the cable connecting the port to another port is not installed since the device is not capable of communicating.

Systems, or in this case ComputerSystem, represent the fabric elements that contain Ports. These are typically Hosts, Switches and Storage Systems. In Fibre Channel, these are called Platforms and Interconnect Elements. The property Dedicated in ComputerSystem allows these fabric elements to be identified. For a host, Dedicated is set to “Not Dedicated”, for a switch, Dedicated is set to “Switch” and for a storage system, Dedicated is set to “Storage”. The Ports on a System are associated by SystemDevice.

Discovery from the viewpoint of the fabric includes the end device, but often times the information available is minimal or not available. In the case of Fibre Channel, this occurs if the platform database is not populated. If this is the case, then discovery cannot tell whether a Fibre Channel Node is contained within the same platform or not. When this occurs, ComputerSystem is not instantiated and the LogicalPortGroup representing the Node and the FCPort are associated to the AdminDomain representing the Fabric.

Additional identification information about ComputerSystem (e.g. DomainID) is placed in OtherIdentifyInfo property.

7.3.4.1.1.4 Zoning

The zoning model is based on ISO/IEC 14165-414 (FC-GS-4). This model represents the management model for defining Zone Sets, Zones and Zone Members and “activation” of a Zone Set for a fabric, as definded below:

Active ZoneSet: the Zone Set currently enforced by the Fabric.

Zone Set Database: the database of the Zone Sets not enforced by the Fabric. Referred to in this document as the Inactive Zone Sets.

Zoning Definitions: a generic term used to indicate both the above concepts.

The zoning model refers to a Zone Set as ZoneSet (7.3.4.1.12.21), a Zone as Zone (7.3.4.1.12.18), ZoneAlias as a NamedAddressCollection and Zone Member as ZoneMembershipSettingData (7.3.4.1.12.20). ZoneSets shall only contain Zones associated by MemberOfCollection. Zones shall only contain ZoneMembershipSettingData associated by ElementSettingData or NamedAddressCollections associated by MemberOfCollection. For more information with regards to NamedAddressCollection, see Enhanced Zoning and Enhanced Zoning Control Subprofile (7.3.4.1.15).

The class ZoneMembershipSettingData has two properties that indicate how the device was identified to be “zoned”. They are ConnectivityMemberType (e.g. PermanentAddress for WWN, NetworkAddress for FCID, etc.) and ConnectivityMemberID which contains the actual device identifier.

The Active Zone Set, defined by an instance of ZoneSet with the Active property set to TRUE, shall only be hosted on the AdminDomain representing the Fabric. The Inactive Zone Sets, defined by an instance of ZoneSet with the Active property set to FALSE, shall be hosted on either the AdminDomain representing the Fabric, as shown in the Figure 53 or the ComputerSystem representing the switch as shown in the Figure 54. The ZoneService and ZoneCapabilities are also associated to the same System (AdminDomain or ComputerSystem) as the Inactive Zone Sets using the association HostedService or ElementCapabilities, respectively.

ZoneService provides the configuration methods to control create Zone Sets, Zones, Zone Aliases and Zone Members, as well as activation of the Zone Set. This service and its methods are described in the Zone Control Subprofile (7.3.4.1.14).

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.4.1.2 Standard dependencies

The Fabric Discovery Profile is based on the following standards:

Table 129 - Fabric Standard Dependencies

Standard	Version	Organization
CIM Specification	2.2	DMTF
CIM Operations over HTTP	1.1	DMTF
CIM Schema	2.8 Preliminary	DMTF

7.3.4.1.3 Profile dependencies

The Fabric Discovery Profile requires the Server Profile (7.3.7).

7.3.4.1.4 CIM Server requirements

7.3.4.1.4.1 Functional Profiles

Table 130 - Required Functional Profiles

Profile Required	Functional Group	Dependency
YES	Basic Read	None
NO	Basic Write	Basic Read
NO	Instance Manipulation	Basic Write
NO	Schema Manipulation	Instance Manipulation
YES	Association Traversal	Basic Read
NO	Query Execution	Basic Read
NO	Qualifier Declaration	Schema Manipulation
YES	Indication	None

7.3.4.1.4.2 Extrinsic methods

The CIM Server shall support extrinsic methods for the Fabric Discovery Profile.

7.3.4.1.4.3 Discovery

The CIM Server shall support SLP discovery as defined in the CIM Operations over HTTP specification.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.4.1.5 Instance diagrams

7.3.4.1.5.1 Fabric Instance

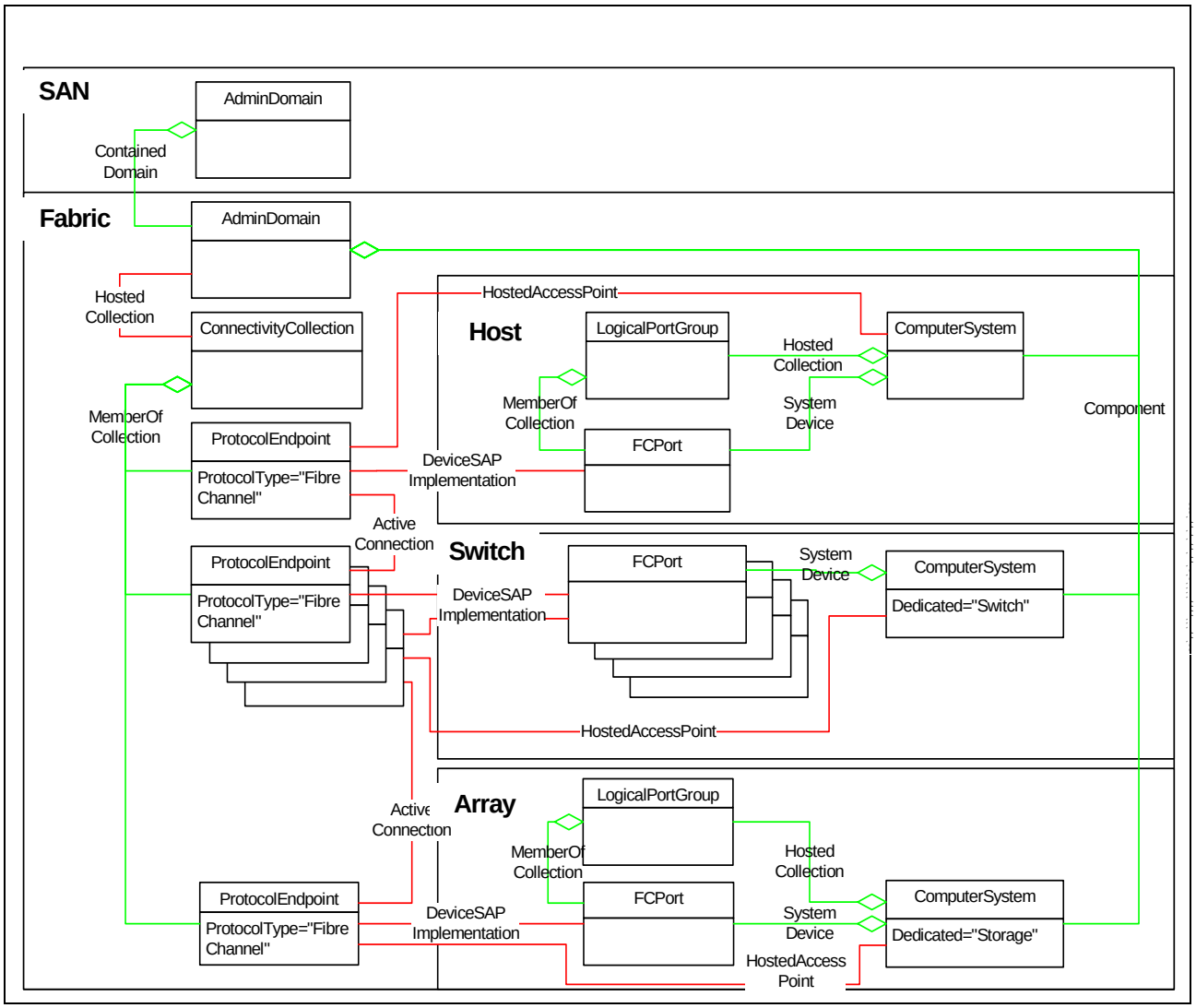


Figure 52 - Fabric Instance Diagram

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.3.4.1.5.2 Zoning

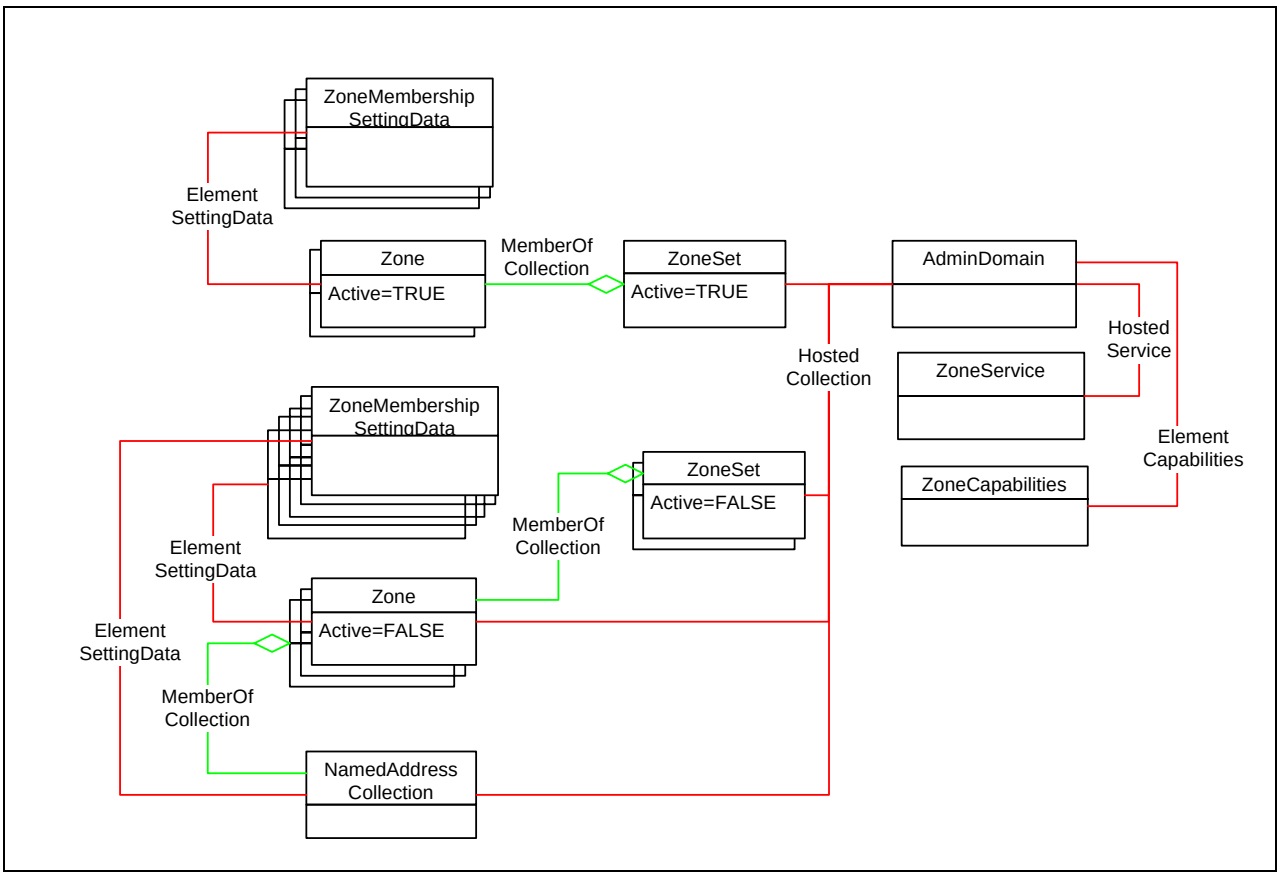


Figure 53 - Zoning Instance Diagram (AdminDomain)

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

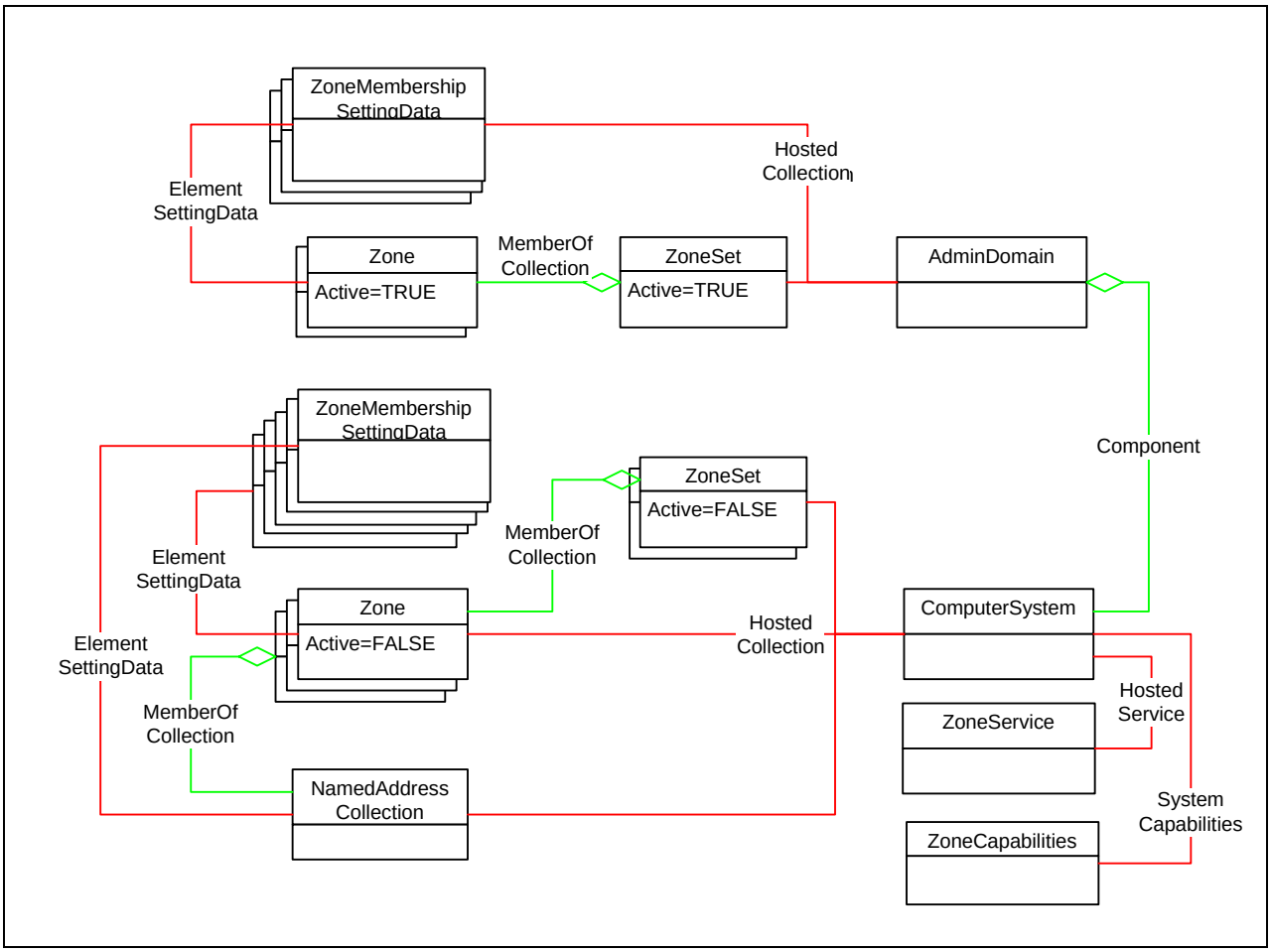


Figure 54 - Zoning Instance Diagram (ComputerSystem)

7.3.4.1.6 Durable names and correlatable IDs of the Profile

7.3.4.1.6.1 Overview

For the Fibre Channel Port, the durable name is the Port WWN. It is found in FCPort.PermanentAddress.

For the Fibre Channel Switch, the durable name is the Switch WWN. It is found in ComputerSystem.Name for ComputerSystems.Dedicated =”Switch”. ComputerSystem.NameFormat is set to “WWN”.

For the Fibre Channel Node, the durable name is the Node WWN. It is found in LogicalPortGroup.Name with NameFormat set to WWN.

For the Fabric Name, the correlatable name is the Fabric WWN that is actually the principal switches WWN. It is found in AdminDomain.Name. AdminDomain.NameFormat is set to “WWN”.

For the Zone Member Identifier, the correlatable name is the Port WWN, the Node WWN or the Domain and Port. These are found in ZoneMembershipSettingData.ConnectivityMemberID with the corresponding ConnectivityMemberType set.

7.3.4.1.6.2 Durable Names Exported

None.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.3.4.1.6.3 Correlatable IDs Supported

None.

7.3.4.1.6.4 Durable names and correlatable Ds

Table 131 - Durable Names Usage

Class	Name Format	Type	Description
AdminDomain.ElementName			

7.3.4.1.6.5 Correlatable IDs used

None.

7.3.4.1.7 Methods

None.

7.3.4.1.8 Client considerations

7.3.4.1.8.1 Fabric identifier

The client needs to consider that the fabric identifier is not durable but is correlatable and may change over time, see 7.2.4, “Correlatable and Durable Names”.

7.3.4.1.8.2 FCPort OperationalStatus

OperationalStatus is the property to indicate status and state for the FCPort. The FCPort instance has one of the following Operational Statuses.

Table 132 - Port OperationalStatus

Operational Status	Description
OK	Port is online
Error	Port has a failure
Stopped	Port is disabled
InService	Port is in Self Test
Unknown	

7.3.4.1.8.3 ComputerSystem OperationalStatus

OperationalStatus is the property to indicate status and state for the ComputerSystem. The ComputerSystem instance has one of the following Operational Statuses and possibly one of the Subsidiary statuses.

Table 133 - OperationalStatus for ComputerSystem

Operational Status	Possible Subsidiary Operational Status	Description
OK		The system has a good status.
OK	Stressed	The system is stressed, for example the temperature is over the limit or there is too much IO in progress.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table 133 - OperationalStatus for ComputerSystem

Operational Status	Possible Subsidiary Operational Status	Description
OK	Predictive Failure	The system will probably fail sometime soon.
Degraded		The system is operational but not at 100 % redundancy. A component has suffered a failure or something is running slow.
Error		An error has occurred causing the system to stop. This error may be recoverable with operator intervention.
Error	Non-recoverable error	A severe error has occurred. Operator intervention is unlikely to fix it.
Error	Supporting entity in error	A modelled element has failed.
No contact		The provider knows about the array but has not talked to it since the last reboot.
Lost communication		The provider used to be able to communicate with the array, but has now lost contact.
Starting		The system is starting up.
Stopping		The system is shutting down.
Stopped		The data path is OK but shut down, the management channel is still working.

7.3.4.1.9 Recipes

7.3.4.1.9.1 Determine the active Zone Set in a SAN

```
// DESCRIPTION
// Traverse from the fabric to all zone sets, looking for
// the active zone set
//
// PREEXISTING CONDITIONS AND ASSUMPTIONS
//
// 1. The fabric of interest (an AdminDomain) has been previously
//   identified and defined in the $Fabric-> variable

$ZoneSets[] = Associators($Fabric->, “CIM_HostedCollection”,
                          “CIM_ZoneSet”, null, null, false, false,
                          null)

for #i in $ZoneSets[] {
    if ($ZoneSet[#i].Active) {
        // <found active ZoneSet>
        // NOTE - there can be only one active ZoneSet in a fabric,
                          though there may be none
        break
    }
```

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
}
```

7.3.4.1.10 Instrumentation requirements

The agent needs to respond to physical fabric changes by adding or removing Logical elements to the AdminDomain. Adding an element to the fabric is straightforward. However, it is not always clear when an element has been removed. The device may have been reset, or temporarily shut down, in which case it would be an element in the fabric with an “unknown” status. The lifetime of “objects that can no longer be discovered” is implementation specific.

If the agent is unable to determine the type of platform discovered (defined in ISO/IEC 14165-414), then the agent shall set the ComputerSystem.Dedicated property to “Unknown”.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.4.1.11 Required CIM Elements

Table 134 - Required CIM Elements

Profile Classes & Associations	Notes
ActiveConnection (7.3.4.1.12.1)	
AdminDomain (7.3.4.1.12.2)	Representing the fabric.
AdminDomain (7.3.4.1.12.2)	Representing the SAN.
Component (7.3.4.1.12.3)	Aggregates Hosts, Arrays and Switches in the AdminDomain that represents the Fabric.
ComputerSystem (7.3.4.1.12.4)	
ConnectivityCollection (7.3.4.1.12.12)	Collects the ProtocolEndpoints.
ContainedDomain (7.3.4.1.12.5)	Associates Fabric to SAN.
DeviceSAPImplementation (7.3.4.1.12.6)	
ElementCapabilities (7.3.4.1.12.7)	Associates ZoneCapabilities to AdminDomain or ComputerSystem.
ElementSettingData (7.3.4.1.12.8)	Associates ZoneMembershipSettingData to the Zone or NamedAddressCollection representing the ZoneAlias.
FCPort (7.3.4.1.12.9)	
HostedAccessPoint (7.3.4.1.12.10)	Associates the ProtocolEndpoint to the ComputerSystem (Switch or Platform).
HostedCollection (7.3.4.1.12.11)	LogicalPortGroup to ComputerSystem.
HostedCollection (7.3.4.1.12.11)	ConnectivityCollection to AdminDomain.
HostedCollection (7.3.4.1.12.11)	AdminDomain or ComputerSystem to ZoneSets, Zones and NamedAddressCollection.
LogicalPortGroup (7.3.4.1.12.13)	Fibre Channel Node.
MemberOfCollection (LogicalPortGroup) (7.3.4.1.12.14)	FCPort to LogicalPortGroup.
ProtocolEndpoint (7.3.4.1.12.16)	ProtocolEndpoint shall be implemented when an ActiveConnection exists. It may be implemented if no ActiveConnection exists.
SystemDevice (7.3.4.1.12.17)	
Zone (7.3.4.1.12.18)	
()	This class is optional.
ZoneMembershipSettingData (7.3.4.1.12.20)	
ZoneMembershipSettingData (7.3.4.1.12.20)	

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table 134 - Required CIM Elements (Continued)

Profile Classes & Associations	Notes
ZoneSet (7.3.4.1.12.21)	
Profile Class and Associated Indications	
Creation/Deletion of FCPort	“SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_FCPort” SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_FCPort”
Creation/Deletion of ComputerSystem	“SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_ComputerSystem” SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_ComputerSystem”
Changes in OperationalStatus of FCPort	“SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_FCPort AND SourceInstance.OperationalStatus <> PreviousInstance.OperationalStatus”
Changes in OperationalStatus of ComputerSystem	“SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_ComputerSystem AND SourceInstance.OperationalStatus <> PreviousInstance.OperationalStatus”

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.3.4.1.12 Required Properties for CIM Elements

7.3.4.1.12.1 ActiveConnection

Table 135 - Required Properties for ActiveConnection

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Antecedent	ref	key	ProtocolEndpoint reference.
Dependent	ref	key	ProtocolEndpoint reference.

7.3.4.1.12.2 AdminDomain

Table 136 - Required Properties for AdminDomain

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
CreationClassName	string	maxlen(256), key	Name of Class.
Name	string	maxlen(256), key, override	For a Fibre Channel Fabric, it should be WWN. For a SAN, it is implementation dependent.
NameFormat	string	maxlen(64)	For a Fibre Channel Fabric, it should be “WWN”.

7.3.4.1.12.3 Component

Table 137 - Required Properties for Component

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
GroupComponent	ref	key	AdminDomain (for Fabric) reference.
PartComponent	ref	key	ComputerSystem reference.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.4.1.12.4 ComputerSystem

Table 138 - Required Properties for ComputerSystem

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
ElementName	string		Switch Symbolic Name. For Platform it is the Platform Label.
CreationClassName	string	maxlen(256), key	Name of Class.
OperationalStatus	uint16	req	
Name	string	maxlen(256), key	For Switches, it is the FC WWN. For Platforms, it is the Platform Name if available.
NameFormat	string	(override “nameformat”)	For Switches, “WWN”. For all others, follow Table 3.
OtherIdentifyingInfo[ ]	string		The DomainID is stored here (in base 10).
IdentifyingDescriptions[ ]	string		“DomainID” is placed in the corresponding index.
Dedicated[ ]	int16		For a Switch, “Switch”. For a Host. “Not Dedicated”. For Arrays, “Storage”. For .... (Map from FC-GS Name to CIM Enumeration”).
OtherDedicatedDescriptions	string		A string describing how or why the system is dedicated when the Dedicated array includes the value 2, \"Other\".

7.3.4.1.12.5 ContainedDomain

Table 139 - Required Properties for ContainedDomain

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
GroupComponent	ref	key	AdminDomain (for SAN) reference.
PartComponent	ref	key	AdminDomain (for Fabric) reference.

7.3.4.1.12.6 DeviceSAPImplementation

Table 140 - Required Properties for DeviceSAPImplementation

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Antecedent	ref	key	FCPort reference.
Dependent	ref	key	ProtocolEndpoint reference.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.4.1.12.7 ElementCapabilities

Table 141 - Required Properties for ElementCapabilities

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
ManagedElement	ref		AdminDomain.
Capabilities	ref		ZoneCapabilities.

7.3.4.1.12.8 ElementSettingData

Table 142 - Required Properties for ElementSettingData

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
ManagedElement	ref	key	Zone or ZoneAlias.
SettingData	ref	key	ZoneMembershipSettingData.

7.3.4.1.12.9 FCPort

Table 143 - Required Properties for FCPort

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
SystemName	string	key	
SystemCreationClassName	string	key	
CreationClassName	string	key	
ElementName	**string**		Port 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.
OperationalStatus	uint16		See Table 132.
DeviceID	string	key, maxlen (64)	Opaque.
Speed	uint64	units ("bits per second")	Speed of zero represents a link not established. 1 Gbit is 1 062 500 000 bit/s 2 Gbit is 2 125 000 000 bit/s 4 Gbit is 4 250 000 000 bit/s) 10 Gbit single channel variants are 10 518 750 000 bit/s 10 Gbit four channel variants are 12 750 000 000 bit/s This is the raw bit rate.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table 143 - Required Properties for FCPort (Continued)

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
PortType	uint16		FC-GS Port.Type The specific mode currently enabled for the Port. The values: \"N\" = Node Port, \"NL\" = Node Port supporting FC arbitrated loop, \"E\" = Expansion Port connecting fabric elements (for example, FC switches), \"F\" = Fabric (element) Port, \"FL\" = Fabric (element) Port supporting FC arbitrated loop and \"B\" = Bridge Port. PortTypes are defined in ISO/ IEC 14165-414. When set to 1 (\"Other\"), the related property OtherPortType contains a string description of the port's type. PortType is defined to force consistent naming of the 'type' property in subclasses and to guarantee unique enum values for all instances of NetworkPort. A range of values, DMTF_Reserved, has been defined that allows subclasses to override and define their specific port types. Vendor Reserved = 16 000..65 535 can be used if the PortType is not one already defined in the above enumerations and a vendor subclass is defined specifying the appropriate “value” and “valuemap”.
OtherNetworkPortType	string		Describes the type of module, when PortType is set to 1 (\"Other\").
LinkTechnology	uint16		For FibreChannel, “FC”.
OtherLinkTechnology	uint16		A string value describing LinkTechnology when it is set to 1, \"Other\".
PermanentAddress	string	maxlen (64)	For FibreChannel, it is the Fibre Channel Port WWN.
NetworkAddresses[ ]	string	maxlen (64), arraytype ("indexed")	For Fibre Channel end device ports, it is the Fibre Channel ID. For Switches, it should be Null.
SupportedCOS	uint16[ ]		FC-GS Class Of Service An array of integers indicating the Classes of Service that are active. Not applicable for switches (e.g. NULL).

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table 143 - Required Properties for FCPort (Continued)

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
SupportedFC4Types	uint16[ ]		FC-GS FC4-TYPE An array of integers indicating the Fibre Channel FC-4 protocols currently running. Not applicable for switches (e.g. NULL).

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.3.4.1.12.10 HostedAccessPoint

Table 144 - HostedAccessPoint

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Antecedent	ref		ComputerSystem.
Dependent	ref		ProtocolEndpoint.

7.3.4.1.12.11 HostedCollection

Table 145 - Required Properties for HostedCollection

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Antecedent	ref	key, min(1), max(1)	ComputerSystem.
Dependent	ref	key, weak	LogicalPortGroup.

7.3.4.1.12.12 ConnectivityCollection

Table 146 - Required Properties for ConnectivityCollection

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
InstanceID	string		
ElementName			Not required, can be the Fabric WWN.

7.3.4.1.12.13 LogicalPortGroup

Table 147 - Required Properties for LogicalPortGroup

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
ElementName	string		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.
InstanceID	string	key	Opaque.
Name			Node WWN.
NameFormat	string		“WWN”

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.3.4.1.12.14 MemberOfCollection (LogicalPortGroup)

Table 148 - Required Properties for MemberOfCollection

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Collection	ref	key	LogicalPortGroup.
Member	ref	key	FCPort.

7.3.4.1.12.15 MemberOfCollection (ConnectivityCollection)

Table 149 - Required Properties for MemberOfCollection

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Collection	ref	key	ConnectivityCollection
ManagedElement	ref	key	ProtocolEndpoint.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.4.1.12.16 ProtocolEndpoint

Table 150 - Required Properties for ProtocolEndpoint

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Name	string	key, maxlen (256)	
CreationClassName	string	key, maxlen (256)	ComputerSystem that HostedAccessPoint is associated to.
SystemCreationClassName	string	key, maxlen (256)	
SystemName	string	key, maxlen (256)	
NameFormat	string	maxlen (256)	NameFormat shall be “WWN”
ProtocolType	string	maxlen (64), valuemap {} values {}	ProtocolType shall be "Fibre Channel"

7.3.4.1.12.17 SystemDevice

Table 151 - Required Properties for SystemDevice

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
GroupComponent	ref	override	ComputerSystem
PartComponent	ref	override	FCPort

7.3.4.1.12.18 Zone

Table 152 - Required Properties for Zone

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
InstanceID	string	key	
ElementName	string		The Zone Name (FC-GS)
ZoneType	uint16 (enum)		Default, or Protocol (FC-GS).
ZoneSubType	uint16 (enum)		FCP, VI, IP (Optional, only required when ZoneType=Protocol)
Active	boolean		This Zone is active.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.3.4.1.12.19 ZoneCapabilities

NOTE The ZoneCapabilities Table 153 and its properties are optional.

Table 153 - Required Properties for ZoneCapabilities

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
InstanceID	string	key	
MaxNumZoneSets	uint32		If Null, it is indeterminate.
MaxNumZones	uint32		If Null ,it is indeterminate.
MaxNumZoneMembers	uint32		If Null, it is indeterminate.
MaxNumZoneAliases	uint32		If Null, it is indeterminate.
MaxNumZonesPerZoneSet	uint32		If Null, it is indeterminate.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.4.1.12.20 ZoneMembershipSettingData

Table 154 - Required Properties for ZoneMembershipSettingData

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
InstanceID	string	key	
ConnectivityMemberType		required	Permanent Address (WWN), Switch Port ID (Domain:Port in base10), Network Address (FCID).
ConnectivityMemberID		required	The value of the WWN, Domain/Port or FCID.

7.3.4.1.12.21 ZoneSet

Table 155 - Required Properties for ZoneSet

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
InstanceID	string	key	
ElementName	string	required	The ZoneSet name.
Active	boolean	required	Indicates that this ZoneSet is active and cannot be changed.

7.3.4.1.13 Optional Subprofiles

Table 156 - Optional Profiles or Subprofiles

Name	Notes
Zone Control Subprofile (7.3.4.1.14)	
Enhanced Zoning and Enhanced Zoning Control Subprofile (7.3.4.1.15)	
FDMI Subprofile (7.3.4.1.16)	

7.3.4.1.14 Zone Control Subprofile

7.3.4.1.14.1 Description

The zoning model includes extrinsic methods for creating Zone Sets, Zones and Zone Members and adding Zones to Zone Sets and Zone Members to Zones. Additionally, SMI-S defines intrinisic methods for the removing of Zone Members from Zones and Zone Aliases, Zones from Zone Sets, and deleting Zone Members, Zones and Zone Sets.

When an Inactive ZoneSet is “Activated”, new instances representing the Active Zone Set and Active Zones are generated from the Inactive Zone Set definition (where a switch may prune the referenced Zone Set collapsing aliases, removes empty zones, etc.).

When a new Zone Set is “Activated”, the instances representing the previous active Zone Set no longer exists.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

In the case where the Inactive Zone Sets are hosted on a switch, the client cannot know which Inactive Zone Set was used to define the current Active Zone Set. Also, if two Inactive Zone Sets with the same name are hosted on two different switches, the definitions maybe completely different.

7.3.4.1.14.2 Standard dependencies

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

7.3.4.1.14.3 Profile dependencies

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

7.3.4.1.14.4 CIM Server requirements

7.3.4.1.14.4.1 Functional Profiles

Table 157 - Required Functional Profiles

Profile Required	Functional Group	Dependency
YES	Basic Read	None
NO	Basic Write	Basic Read
YES	Instance Manipulation	Basic Write
NO	Schema Manipulation	Instance Manipulation
YES	Association Traversal	Basic Read
NO	Query Execution	Basic Read
NO	Qualifier Declaration	Schema Manipulation
YES	Indication	None

7.3.4.1.14.4.2 Extrinsic methods

The CIM Server shall support extrinsic methods for the Fabric Discovery Profile.

7.3.4.1.14.4.3 Discovery

The CIM Server shall support SLP discovery as defined in the CIM Operations over HTTP specification.

7.3.4.1.14.5 Instance Diagrams

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

7.3.4.1.14.6 Durable names and correlatable IDs

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

7.3.4.1.14.7 Extrinsic Zoning methods

7.3.4.1.14.7.1 CreateZoneSet

The method creates a ZoneSet and associates it to the AdminDomain that the ZoneService is hosted.

```
CreateZoneSet (
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
                string ZoneSetName,
                [Out] ref CIM_ZoneSet);
```

7.3.4.1.14.7.2 CreateZone

The method creates a Zone and associates it to AdminDomain that the ZoneService is hosted.

```
CreateZone (
                string ZoneName,
                Uint16 ZoneType,
                Uint16 ZoneSubType,
                [Out] ref Zone);
```

7.3.4.1.14.7.3 CreateZoneMembershipSettingData

CreateZoneMembershipSettingData creates a ZoneMembershipSettingData and adds it to the specified Zone or NamedAddressCollection. The ConnectivityMemberID is dependent upon the ConnectivityMemberType.

For Fibre Channel, the ConnectivityMemberType of “PermanentAddress”, the ConnectivityMemberID is the NxPort WWN. For ConnectivityMemberType of “NetworkAddress”, the ConnectivityMemberID is the NXPort Address ID. For ConnectivityMemberType of “SwitchPortID”, the ConnectivityMemberID is “Domain:PortNumber”.

```
CreateZoneMembershipSettingData (
                Uint16 ConnectivityMemberType,
                string ConnectivityMemberID,
                ref SystemSpecificCollection,
                [Out] ref ZoneMembershipSettingData);
```

7.3.4.1.14.7.4 AddZone

Adds to the ZoneSet the specified Zone. Adding a Zone to a ZoneSet, extends the zone enforcement definition of the ZoneSet to include the members of that Zone. If adding the Zone is successful, the Zone should be associated to the ZoneSet by MemberOfCollection.

```
AddZone (
                [In] CIM_ZoneSet ref ZoneSet,
                [In] CIM_Zone ref Zone,
                [Out] CIM_MemberOfCollection ref MemberOfCollection);
```

7.3.4.1.14.7.5 AddZoneMembershipSettingData

Adds to the Zone or NamedAddessCollection the specified ZoneMembershipSettingData.

```
AddZoneMembershipSettingData (
                [In] CIM_SystemSpecificCollection ref
                                SystemSpecificCollection,
                [In] CIM_ZoneMembershipSettingData ref
                                ZoneMembershipSettingData,
                [Out] CIM_MemberOfCollection ref MemberOfCollection);
```

7.3.4.1.14.7.6 ActivateZoneSet

```
Uint32 ActivateZoneSet (
                [In] CIM_ZoneSet ref ZoneSet,
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
                    [In] boolean Activate )
```

7.3.4.1.14.7.7 SessionControl

SessionControl enables an application to request a lock of the fabric to begin zoning configuration changes.

This method allows a client to request or release a lock on the fabric for zoning configuration changes. As described in FC-GS, in the context of Enhanced Zoning Management, management actions to a Zone Server (e.g. write access to the Zoning Database) shall occur only inside a GS session. Clients executing zoning management operations shall use fabric sessions cooperatively if the SMI-S agent supports it. (If the value of SessionStatus is 4 (“Not Applicable”) then no cooperative session usage is possible).

Before a client executes zoning management operations (intrinsic or extrinsic methods), the client shall request a new session and wait for the request to be granted. To request a new session, first wait until the property “SessionStatus” of the fabric’s CIM_ZoneService is 3 (“Ended”) and the property “RequestedSessionStatus” is 5 “No Change”. Then call SessionControl with RequestedSessionStatus = 2 (“Started”). Once zoning management operations are completed, the client shall release the session to enable the provider to propagate changes to the fabric and to allow other clients to perform management operations. To end a session and commit the changes, call SessionControl with RequestedSessionStatus = 3 (“Ended”). To abort a sequence of zoning management operations without updating the fabric, call SessionControl with RequestedSessionStatus = 4 (“Terminated”).

SMIS agents shall block on calls to SessionControl until the request is fulfilled. For example, an error may occur while committing changes to a fabric, i.e. after a call to SessionControl with RequestedSessionStatus = 3 (“Ended”). The method cannot return until the session has ended, so that a CIM error can be returned if a problem occurs. While the method is in progress, another client may read the value of the RequestedSessionStatus property and see the value set by the method in progress at the same time. Once the request is fulfilled, the RequestedSessionStatus property is set to value 5 “No Change”, regardless of the value in the setInstance operation.

A SMIS agent may raise an error if these client cooperation rules are not followed. For the purposes of a SMIS agent, a series of requests from the same authenticated entity are considered to be from a single client. An agent may verify that such a series corresponds to the sequence described above and raise the error CIM_ERR_FAILED at any time if the sequence is violated.

```
Uint32 SessionControl (
            [In,
             Heffalump {“2”, “3”, “4”},
             Values {“Started”, “Ended”, “Terminated”}]
            Uint16 RequestedSessionStatus;};
```

7.3.4.1.14.8 Intrinsic Zoning Methods

7.3.4.1.14.8.1 Removing a zone from a zone set

As seen in the instance diagram, a zone is a member of a zone set if there is a “CIM_MemberOfCollection” association from the zone set to the zone. To remove a zone from a zone set, delete the instance of the association “CIM_MemberOfCollection” using the intrinsic operation deleteInstance.

7.3.4.1.14.8.2 Removing a zone alias from a zone

A zone alias is a member of a zone if there is a “CIM_MemberOfCollection” association from the zone to the zone alias. To remove a zone alias from a zone set, delete the instance of the association “CIM_MemberOfCollection” using the intrinsic operation deleteInstance.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.4.1.14.8.3 Removing a zone member from a zone or zone alias

Zone members are represented by CIM_ZoneMembershipSettingData instances. No instance of CIM_ZoneMembershipSettingData exists unless it is associated to a zone or zone alias by a CIM_ElementSettingData association. However, an instance of CIM_ZoneMembershipSettingData may be associated to more than one zone or zone alias.

Removing a zone member from a zone or zone alias is equivalent to deleting the instance of the CIM_ElementSettingData association. Delete the instance using the intrinsic operation deleteInstance. Clients are allowed to delete all the members of a zone or zone alias this way, leaving the zone or zone alias empty.

If this is the last instance of a CIM_ElementSettingData association for a particular CIM_ZoneMembershipSettingData, do not delete the instance of CIM_ZoneMembershipSettingData; it is the provider's responsibility to clean up these structures.

7.3.4.1.14.8.4 Deleting a zone member

Zone members are represented by CIM_ZoneMembershipSettingData instances associated to zones or zone aliases via CIM_ElementSettingData associations. To delete a zone member (and remove it from any zones or zone aliases from which it is a member) use the CIM operation deleteInstance to delete the instance of CIM_ZoneMembershipSettingData.

Do not delete the corresponding instances of the CIM_ElementSettingData; it is the provider's responsibility to clean up these structures.

Clients are allowed to delete the last member in a zone alias or zone, leaving the zone or zone alias empty.

7.3.4.1.14.8.5 Deleting a zone, zone alias, or zone set

Use the intrinsic operation deleteInstance to delete a zone, zone alias or zone set. Client are allowed to delete zones or zone aliases that are members of collections (zones or zone sets). Clients are allowed to delete the last member of a zone or zone set, leaving the collection empty.

A zone set or zone cannot be deleted if it is currently active (the error would be CIM_ERR_FAILED). Some implementations may prohibit deleting zonesets, zones or zone aliases that still have members (the error would be CIM_ERR_FAILED). When a zone, zone alias or zone set is deleted, the client does not have to delete the corresponding instances of CIM_MemberOfCollection or CIM_HostedCollection; it is the provider's responsibility to clean up these structures.

7.3.4.1.14.9 Client Considerations

Many agent implementations do not allow Zone, a ZoneAlias or a Zone Set to be defined empty. Since the methods defined in SMI-S do not support creating a Zone Set with a Zone and a Zone with a Zone Member, the SessionControl method should be used to build a Zone Definition that is interoperable. This is done by calling ZoneSession() to “Start” defining or updating the Zone Definition. The client then calls the appropriate methods as necessary to build the desired Zone Definition. For example, calling CreateZoneSet() to create a new Zone Set, CreateZone() to create a new Zone, AddZoneToZoneSet() to add the newly created Zone to the newly created Zone Set, and CreateZoneMembershipSettingData() to create and add a new Zone Member to the newly created Zone. Upon completion of the new zoning definition, ZoneControl is called again to “End” the session. The changes to the Zone Definition would then be applied to the Zone Set Database. This set of calls would create a Zone Definition where the Zone and ZoneSet are not empty and would be interoperable across all agent implementations.

7.3.4.1.14.10 Recipes

7.3.4.1.14.10.1 Create or delete zones Common Functions

```
// DESCRIPTION
//
// Common functions used by the recipes below.
```

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
//
// startSession: attempt to start fabric session if required;
//   returns false if attempt fails; returns true if attempt succeeds
//   or if session control is unnecessary
//
// endSession: finalize fabric session if required; returns false
//   if attempt fails; returns true if attempt succeeds or if session
//   control is unnecessary
//
// PREEXISTING CONDITIONS AND ASSUMPTIONS
//
// None

sub boolean startSession ($ZoneService->)
{
    $ZoneService = GetInstance($ZoneService->, false, false, false,
                          null)

    // session statuses
    #Ended = 3
    #NotApplicable = 4

    // requested session statuses
    #Started = 2
    #NoChange = 5

    if ($ZoneService.SessionState == #NotApplicable)
         return true // no session control implemented by this agent

    if ($ZoneService.SessionState != #Ended)
         return false // fabric session is in use by another client or
                          agent

    if ($ZoneService.RequestedSessionState != #NoChange)
         return false // another client has already requested session

    %InArguments[“RequestedSessionState”] = #Started

    #status = InvokeMethod($ZoneService->, “SessionControl”,
                          %InArguments, %OutArguments)
    if (#status != 0) // e.g. “Failed”
        return false

    $ZoneService = GetInstance($ZoneService->, false, false, false,
                          null)
    if ($ZoneService.SessionState != #Started)
        return false
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
    return true
}

sub boolean endSession ($ZoneService->) {
    $ZoneService = GetInstance($ZoneService->, false, false, false,
                               null)

    // session statuses
    #Started = 2
    #NotApplicable = 4

    // requested session statuses
    #End = 3

    if ($ZoneService.SessionStatus == #NotApplicable){
        return true      // no need for session control

    if ($ZoneService.SessionStatus != #Started)
        return false     // no session started by this client

    %InArguments[“RequestedSessionState”] = #End
    #status = InvokeMethod($ZoneService, “SessionControl”,
                           %InArguments, %OutArguments)
    if (#status != 0)   // e.g. “Failed”
        return false

    // Do not wait, or even check, for SessionState to have value
                           “Ended” as
    // a) InvokeMethod will block till done (or failed) anyway
    // b) Before the check can be made, session may already be started
    //    by another client

    return true
}
```

7.3.4.1.14.10.2 Add new or existing Zone Member to Existing Zone

```
// DESCRIPTION
// Add new or existing Zone Member to Existing Zone
//
// Assume the client has already invoked some logic to determine which
// System (fabric or switch) will host the zone database and zone
// service to be used.  Request and obtain a fabric session from the
// zone service.  Use an extrinsic method to attempt to create a new
// instance of ZoneMembershipSettingData, associated to a zone.  If
// the creation fails because an instance already exists for the
// desired zone member id, simply create an association between the
// pre-existing ZoneMembershipSettingData instance and the zone
// instance.  Then close the fabric session.
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
//
// PREEXISTING CONDITIONS AND ASSUMPTIONS
//
// 1. The System hosting the zone database (ComputerSystem or
//    AdminDomain) has been previously identified and defined in the
//    $System-> variable
//
// 2. The zone member type is defined in the #ConnectivityMemberType
                          variable
//
// 3. The zone member id of the new zone member is defined in the
//    #ConnectivityMemberID variable
//
// 4. An existing zone is defined in the $Zone-> variable
//
// FUNCTIONS

// 1. Get the Zone Service and start the session

$ZoneServices->[] = AssociatorNames($System->, “CIM_HostedService”,
                                    “CIM_ZoneService”, null, null)

// Assumption 1 (above) guarantees there is a zone service for this
// System, Fabric Profile mandates there is no more than one zone
// service for this System
$ZoneService-> = $ZoneService->[0]

// Start the session
if (!&startSession($ZoneService->)) {
    return
}

// 2. Create or locate a ZoneMembershipSettingData
%InArguments[“ConnectivityMemberType”] = #ConnectivityMemberType
%InArguments[“ConnectivityMemberID”] = #ConnectivityMemberID
%InArguments[“SystemSpecificCollection”] = $Zone->
#status = InvokeMethod($ZoneService->,
                          “CreateZoneMembershipSettingData”,
                      %InArguments[], %OutArguments[])

// 3. Add to zone if not created as a member of the zone
// NOTE: ZoneMember output argument is set even if return status is 8
                          (Already_Exists)
$ZoneMember-> = %OutArguments[“ZoneMembershipSettingData”]
if (#status == 8) {
    %InArguments2[“SystemSpecificCollection”] = $Zone
    %InArguments2[“ZoneMembershipSettingData”] = $ZoneMember->
    InvokeMethod($ZoneService->, “AddZoneMembershipSettingData”,
```

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
                    %InArguments2[], %OutArguments[])
}
else if (#status != 0)
    // ERROR!

// 4. End session successfully
&endSession($ZoneService->)
```

7.3.4.1.14.10.3 Create new Zone, add new/existing Zone Member, and add to existing ZoneSet

```
// DESCRIPTION
// Create new Zone, add new/existing Zone Member, and add to existing
                            ZoneSet
//
// Assume the client has already invoked some logic to determine which
// System (fabric or switch) will host the zone database and zone
// service to be used.  Request and obtain a fabric session from the
// zone service.  Create a new Zone using an extrinsic method.  The
// session may not be ended if any zone is empty, so add a zone member
// to the new zone.  The session also may not be ended unless every
// zone is a member of at least one zone set, so add the new zone to
// an existing zone set.  Then close the fabric session.
//
//
// PREEXISTING CONDITIONS AND ASSUMPTIONS
//
// 1. The System hosting the zone database (ComputerSystem or
//    AdminDomain) has been previously identified and defined in the
//    $System-> variable
//
// 2. The name for a new zone is defined in the #ZoneName variable
//
// 3. The type for the new zone is defined in the #ZoneType variable
//
// 4. The sub type for the new zone is defined in the #ZoneSubType
//    variable
//
// 5. The zone member type is defined in the #ConnectivityMemberType
                            variable
//
// 6. The zone member id of the new zone member is defined in the
//    #ConnectivityMemberID variable
//
// 7. An existing zoneSet is defined in the $ZoneSet-> variable
//
// FUNCTIONS

// 1. Get the Zone Service and start the session
$ZoneServices->[] = AssociatorNames($System->, “CIM_HostedService”,
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
                                        “CIM_ZoneService”, null, null)

// Assumption 1 (above) guarantees there is a zone service for this
// System, Fabric Profile mandates there is no more than one zone
// service for this System
$ZoneService-> = $ZoneServices->[0]

    if (!&startSession($ZoneService->)) {
    return
}

// 2. Create a zone
%InArguments[“ZoneName”] = #ZoneName
%InArguments[“ZoneType”] = #ZoneType
%InArguments[“ZoneSubType”] = #ZoneSubType
InvokeMethod($ZoneService->, “CreateZone”, %InArguments[],
                            %OutArguments[])
$Zone-> = $OutArguments[“Zone”]

// 3. Create or locate a ZoneMembershipSettingData
%InArguments[“ConnectivityMemberType”] = #ConnectivityMemberType
%InArguments[“ConnectivityMemberID”] = #ConnectivityMemberID
%InArguments[“SystemSpecificCollection”] = $Zone->
#status = InvokeMethod($ZoneService->,
                            “CreateZoneMembershipSettingData”,
                        %InArguments[], %OutArguments[])

// 4. Add to zone if not created as a member of the zone
$ZoneMember-> = %OutArguments[“ZoneMembershipSettingData”]

if (#status == 8) {
    %InArguments2[“SystemSpecificCollection”] = $Zone->
    %InArguments2[“ZoneMembershipSettingData”] = $ZoneMember->
    InvokeMethod($ZoneService->, “AddZoneMembershipSettingData”,
                %InArguments2[], %OutArguments[])
}
else if (#status != 0)
    // ERROR!

// 5. Add the new zone to the existing zone set
%InArguments[“ZoneSet”] = $ZoneSet->
%InArguments[“Zone”] = $Zone->
#status = InvokeMethod($ZoneService->, “AddZone”, %InArguments[],
                            %OutArguments[])
if (#status != 0)
    // ERROR!

// 6. End Session
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
&endSession($ZoneService->)
```

7.3.4.1.14.10.4 Create new ZoneSet and add existing Zone

```
// DESCRIPTION
// Create new ZoneSet and add existing Zone
//
// Assume the client has already invoked some logic to determine which
// System (fabric or switch) will host the zone database and zone
// service to be used.  Request and obtain a fabric session from the
// zone service.  Create a new ZoneSet with a given name, using an
// extrinsic method.  The session may not be ended if any ZoneSet is
// empty, so add an existing zone to the ZoneSet. Then close the
// fabric session.
//
// PREEXISTING CONDITIONS AND ASSUMPTIONS
//
// 1. The System hosting the zone database (ComputerSystem or
//    AdminDomain) has been previously identified and defined in the
//    $System-> variable
//
// 2. The name for the new zone set is defined in the #ZoneSetName
//    variable
//
// 3. An existing zone is defined in the $Zone-> variable
//
// FUNCTIONS

// 1. Get the Zone Service and start the session

$ZoneServices->[] = AssociatorNames($System->, “CIM_HostedService”,
                                    “CIM_ZoneService”, null, null)
// Assumption 1 (above) guarantees there is a zone service for this
// System, Fabric Profile mandates there is no more than one zone
// service for this System
$ZoneService-> = $ZoneServices->[0]

if (!&startSession($ZoneService->)
   return
}

// 2. Create a zone set
%InArguments[“ZoneSetName”] = #ZoneSetName
#status = InvokeMethod($ZoneService->, “CreateZoneSet”, %InArguments[],
                           %OutArguments[])
if (#status != 0)
    // ERROR!
```

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
$ZoneSet-> = %OutArguments[“ZoneSet”]

// 3. Add the existing zone to the new zone set
%InArguments[“ZoneSet”] = $ZoneSet->
%InArguments[“Zone”] = $Zone->
#status = InvokeMethod($ZoneService->, “AddZone”, %InArguments[],
                           %OutArguments[])
if (#status != 0)
    // ERROR!

// 4. End Session
&endSession($ZoneService->)
```

7.3.4.1.14.10.5 Delete zone

```
// DESCRIPTION
// Delete Zone
//
// Try to use intrinsic delete operation to delete a Zone instance.
// Before any operations can be imposed on the zoning service, a
// session is requested and obtained from the zone service.  If the
// deletion fails, this may be because the zone is active, or because
// it is not empty.  In the latter case, remove all members from the
// zone by deleting the ElementSettingData association instances, and
// try the deletion again.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
// 1.  The object name of the zone to be deleted is defined in the
//     $Zone-> variable
// 2.  The object name of the zone service object for the System
//     hosting the zone database is defined in the $ZoneService->
//     variable


if(!&startSession($ZoneService->))
    return

try {
    DeleteInstance($Zone->)
}
catch(CIM_ERR_FAILED) {
    // Verify that Zone is not active
    $Zone = GetInstance($Zone->, false, false, false, null)
    if ($Zone.Active) {
        // tell client of its logic problem
        throw CIM_ERR_FAILED
    }
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
    // Failure may be caused because zone has members
    // Try to delete all zone memberships (not zone members themselves)
    $ZoneElements->[] = ReferenceNames($Zone->,
                            “CIM_ElementSettingData”, null)
    for #i in $ZoneElements->[] {
        DeleteInstance($ZoneElements[#i])
    }

    // Try again
    DeleteInstance($Zone->)
}

&endSession($ZoneService->)
```

7.3.4.1.14.10.6 Delete ZoneSet

```
// DESCRIPTION
// Delete Zone Set
//
// Try to use intrinsic delete operation to delete a ZoneSet
// instance. Before any operations can be imposed on the zoning
// service, a session is requested and obtained from the zone service.
// The session is released when the operations are complete.  If the
// deletion fails, this may be because the zone set is active, or
// because it is not empty.  In the latter case, remove all zones from
// the zone set by deleting the MemberOfCollection association
// instances, and try the deletion again.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
// 1.  The object name of the zone set to be deleted is defined in the
//     $ZoneSet-> variable
// 2.  The object name of the zone service object for the system
//     hosting the zone database is defined in the $ZoneService->
//     variable

if (!&startSession($ZoneService->))
    return
}

try {
    DeleteInstance($ZoneSet->)
}
catch(CIM_ERR_FAILED) {
    $ZoneSet = GetInstance($ZoneSet->, false, false, false, null)
    if ($ZoneSet.Active) {
```

--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
            // tell client of logic problem
            throw CIM_ERR_FAILED
        }

        // Failure may be because zoneset is not empty
        $ZoneMemberships->[] = ReferenceNames($ZoneSet->,
                               “CIM_MemberOfCollection”, null)
        for #i in $ZoneMemberships->[] {
            DeleteInstance($ZoneMemberships->[$i])
        }

        // Try again
        DeleteInstance($ZoneSet->)
    }

    &endSession($ZoneService->)
```

7.3.4.1.14.10.7 Create ZoneMember

```
    // DESCRIPTION
    // Create a zone member based on the parameters collected by the
    // CIM Client. Before any operations can be imposed on the zoning
                               service,
    // a session is requested and obtained from the zone service.  The
    // session is released when the operations are complete.
    //
    // PRE-EXISTING CONDITIONS AND ASSUMPTION
    // 1.The fabric of interest, the AdminDomain, has been previously
    //   identified and defined in the $Fabric-> variable
    // 2.The zone member is defined in the #ZoneMemberType variable
    // 3.The zone member id of the new zone member is defined in the
    //   #ZoneMemberID  variable
    // 4.The object name of the zone member to be deleted is defined in
    //   the $ZoneMember-> variable
    // 5.Assume that there is only one zone service per fabric

     $ZoneServices->[] = AssociatorNames(
         $Fabric->,
         “CIM_HostedService”,
         “CIM_ZoneService”,
         null,
         null)
    $ZoneService-> = nameof $ZoneService
    if(!&startSession($ZoneService->))
           return

    %InArguments[“ZoneMemberType”] = $ZoneMemberType
    %InArguments[“ZonememberId”] = $ZoneMemberId
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
%InArguments[“SystemSpecificCollection”] = $Fabric
InvokeMethod(
     $ZoneService->,
     “CreateZoneAlias”,
     %InArguments[],
     %OutArguments[])
$ZoneMember-> = %OutArguments[“ZoneMember”]
&endSession($ZoneService->)
```

7.3.4.1.14.10.8 Delete ZoneMember

```
// DESCRIPTION
// Delete a zone member, removing it from any zones and aliases of
// which it is a member.
//
// Use the intrinsic delete operation to delete a
// ZoneMembershipSettingData instance. Before any operations can be
// imposed on the zoning service, a session is requested and obtained
// from the zone service.  The session is released when the operations
// are complete.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
// 1.  The object name of the ZoneMembershipSettingData to be deleted is
                           defined in the
//     $ZoneMember-> variable
// 2.  The object name of the zone service object for the system
//     hosting the zone database is defined in the $ZoneService->
//     variable

if(!&startSession($ZoneService->))
       return

DeleteInstance($ZoneMember->)
&endSession($ZoneService->)
```

7.3.4.1.14.11 Instrumentation Requirements

The agent shall support the use case defined in the Client Considerations (7.3.4.1.15.8).

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.4.1.14.12 Required CIM Elements

Table 158 - Required CIM Elements

Profile Classes & Associations	Notes
HostedService (7.3.4.1.14.14)	Associates ZoneService to the AdminDomain or ComputerSystem.
ZoneService (7.3.4.1.14.15)	
Associated Indications	

7.3.4.1.14.13 Required Properties for CIM Elements

7.3.4.1.14.14 HostedService

Table 159 - Required Properties for HostedService

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Antecedent	ref	key, min(1), max(1)	AdminDomain or ComputerSystem.
Dependent	ref	key, weak	ZoneService.

7.3.4.1.14.15 ZoneService

Table 160 - Required Properties for ZoneService

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
SystemCreationClassName	string	propagated, key	
CreationClassName	string	key	
SystemName	string	propagated, key	AdminDomain.
Name	string	key	
OperationalStatus	uint16 (enum)		
SessionStatus	uint16 (enum)		
RequestedSessionStatus			
CreateZone()	uint16		
DeleteZone()	uint16		
CreateZoneSet()	uint16		
DeleteZoneSet()	uint16		
ActivateZoneSet()	uint16		

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table 160 - Required Properties for ZoneService (Continued)

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
DeactivateZoneSet()	uint16		

7.3.4.1.14.16 Optional Subprofiles

Table 161 - Optional Profiles or Subprofiles

Name	Notes
None	

7.3.4.1.15 Enhanced Zoning and Enhanced Zoning Control Subprofile

7.3.4.1.15.1 Description

This profiel describes the additional zoning functions for enhanced zoning. Note that Sessions are normally part of enhanced zoning, but are included in the base fabric profile to address the various types of zoning operations in a single object model. In this subprofile, only ZoneAlias is added.

7.3.4.1.15.2 Standard dependencies

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

7.3.4.1.15.3 Profile dependencies

Support for the Zone Control Subprofile (7.3.4.1.14) is required by the Enhanced Zoning and Enhanced Zoning Control subprofile.

7.3.4.1.15.4 CIM Server Requirements

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

7.3.4.1.15.5 Instance diagrams

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

7.3.4.1.15.6 Durable names and correlatable IDs

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

7.3.4.1.15.7 Methods

7.3.4.1.15.7.1 CreateZoneAlias

The method creates a ZoneAlias and associates it to AdminDomain that the ZoneService is Hosted on.

```
CreateZoneAlias (
            string ZoneAliasName,
            [Out] ref ZoneAlias);
```

7.3.4.1.15.7.2 AddZoneAlias

Adds to the Zone the specified ZoneAlias.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
AddZoneAlias (
            [In] CIM_Zone ref Zone,
            [In] CIM_ZoneAlias ref ZoneAlias,
[Out] CIM_MemberOfCollection ref MemberOfCollection);
```

7.3.4.1.15.8 Client Considerations

None for this specification.

7.3.4.1.15.9 Recipes

7.3.4.1.15.9.1 Create a ZoneAlias

```
// DESCRIPTION
// Create zone alias and add new/existing zone member based on
// the parameters collected by the  CIM Client.
// Before any operations can be imposed on the zoning
// service, a session is requested and obtained from the zone
// service. Create a new ZoneAlias. The session may not be ended if
// the ZoneAlias is empty, so add a zone member to the new ZoneAlias.
// The session is released when the operations are
// completed.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
// 1.The system of interest,either the fabric (AdminDomain)
//   or the switch (ComputerSystem), has been
//   previously identified and defined in the
//   $System-> variable
// 2.The name of the new zone alias is defined in the
//   #ZoneAliasName variable
// 3.   The zone member type is defined in the #ConnectivityMemberType
//   variable
// 4.   The zone member Id of the new zone member is defined in the
//   #ConnectiivityMemberID variable


// 1. Get the ZoneService and start a session
$ZoneServices->[] = AssociatorNames(
     $System->,
     “CIM_HostedService”,
     “CIM_ZoneService”, null, null)

// Assumption 1 above guarantees there is a zone service for this
// system. The fabric and switch profiles that there is no more than
// one ZoneService for this system
$ZoneService-> = $ZoneServices[0]

if(!&startSession($ZoneService->))
{
     return
```

--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
}

// 2. Create the ZoneAlias
%InArguments[“CollectionAlias”] = #ZoneAliasName
#status = InvokeMethod(
     $ZoneService->,
     “CreateZoneAlias”,
     %InArguments[],
     %OutArguments[])

$ZoneAlias-> = %OutArguments[“ZoneAlias”]
if(#status != 0)
     // ERROR!

// 3. Create or locate a ZoneMembershipSettingData
%InArguments[“ConnectivityMemberType”] = #ConnectivityMemberType
%InArguments[“ConnectivityMemberID”] = #ConnectivityMemberID
%InArguments[“SystemSpecificCollection”] = $ZoneAlias->
#status = InvokeMethod($ZoneService->,
                          “CreateZoneMembershipSettingData”,
                       %InArguments[], %OutArguments[])

// 4. Add to zone alias if not created as a member of the zone alias
//    Zone member reference is set accordingly in the output arguments.

$ZoneMember-> = %OutArguments[“ZoneMembershipSettingData”]

if (#status == 8) {
    // ZoneMembershipSettingData already exists
    %InArguments2[“SystemSpecificCollection”] = $ZoneAlias->
    %InArguments2[“ZoneMembershipSettingData”] = $ZoneMember->
    InvokeMethod($ZoneService->, “AddZoneMembershipSettingData”,
                 %InArguments2[], %OutArguments[])
}
else if (#status != 0)
    // ERROR!


// 5. End the session gracefully
&endSession($ZoneService->)
```

7.3.4.1.15.9.2 Delete a ZoneAlias

```
// DESCRIPTION
// Delete a zone alias.
// Before any operations can be imposed on the zoning service, a
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
// session is requested and obtained from the zone service.
// The session is released when the operations are completed.
//
// if the deletion fails, it may be because the Zone Alias is not empty.
// In this case, remove all members from the alias by deleting the
// ElementSettingData associations, and try the deletion again.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
// 1.The system of interest,either the fabric (AdminDomain)
//   or the switch (ComputerSystem), has been
//   previously identified and defined in the
//   $System-> variable
// 2.The object name of the zone alias to be deleted is
//   defined in the $ZoneAlias-> variable

// 1. Get the zone service and start a session
$ZoneServices->[] = AssociatorNames(
     $System->,
     “CIM_HostedService”,
     “CIM_ZoneService”,
     null,
     null)

// Assumption 1 above guarantees there is a zone service for this
// system. the fabric and switch profiles that there is no more than
// one ZoneService for this system
$ZoneService-> = $ZoneServices[0]

if(!&startSession($ZoneService->))
{
     return
}

// 2. Attempt to delete the alias
try{
     DeleteInstance($ZoneAlias->)
}catch(CIM_ERR_FAILED){
     // Try to remove any zone members in the alias
     // via the ElementSettingData association
     $ZoneMembers->[] = referenceNames($ZoneAlias->,
            “CIM_ElementSettingData”,
            null)
     for #j in $ZoneMembers->[] {
        DeleteInstance(ZoneMembers[#j])
     }
     // Try again
     DeleteInstance($ZoneAlias->)
```

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
}
// 3. End Session
&endSession($ZoneService->)
```

7.3.4.1.15.10 Instrumentation requirements

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.4.1.15.11 Required CIM Elements

Table 162 - Required CIM Elements

Profile Classes & Associations	Notes
HostedCollection	NamedAddressCollection hosted on System.
MemberOfCollection	Associates ZoneMembershipSettingData to NamedAddressCollection.
NamedAddressCollection	The Zone Alias.
ZoneService	
Packages and Subprofiles	
Zone Control Subprofile (7.3.4.1.14)	
Associated Indications	

7.3.4.1.15.12 Required Properties for CIM Elements

7.3.4.1.15.12.1 HostedCollection

Table 163 - Required Properties for HostedCollection

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Antecedent	ref	key, min(1), max(1)	AdminDomain or ComputerSystem.
Dependent	ref	key, weak	NamedAddressCollection.

7.3.4.1.15.12.2 MemberOfCollection

Table 164 - Required Properties of MemberOfCollection

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Collection	ref	key	NamedAddressCollection.
Member	ref	key	ZoneMembershipSettingData.

7.3.4.1.15.12.3 NamedAddressCollection

Table 165 - Required Properties for NamedAddressCollection

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
InstanceID	string	key	REQUIRED.
CollectionAlias	string	required	Zone Alias Name FCSW ZoneAlias.Name, REQUIRED.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.4.1.15.13 ZoneService

The Service responsible for defining the zone enforcement for the fabric. The ZoneService is Hosted on an AdminDomain and defines the containment and scope of the zoning entities.

NOTE The following property list includes only those properties that shall be added to the pre-existing ZoneService instance (required by the Zone Control subprofile).

Table 166 - Required Properties for ZoneService

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
CreateZoneAlias()	uint16		
DeleteZoneAlias()	uint16		

7.3.4.1.15.13.1 Optional Subprofiles

Table 167 - Optional Profiles or Subprofiles

Name	Notes
None.	

7.3.4.1.16 FDMI Subprofile

7.3.4.1.16.1 Description

The Fabric-Device Management Interface (FDMI) enables the management of devices such as HBAs through the Fabric. The FDMI complements data in the Fabric Profile.

This profile only addresses HBA type devices. The HBA Management Interface defined by FDMI is a subset of interface defined by the Fibre Channel HBA API specification, as exposed by the FC HBA Profile (7.3.5.1).

7.3.4.1.16.2 Standard dependencies

See 7.3.5.1, “FC HBA Profile”.

7.3.4.1.16.3 Profile dependencies

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

7.3.4.1.16.4 CIM Server requirements

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

7.3.4.1.16.5 Instance diagrams

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

7.3.4.1.16.6 Durable names and correlatable IDs

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.4.1.16.7 Methods

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

7.3.4.1.16.8 Client considerations

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

7.3.4.1.16.9 Recipes

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

7.3.4.1.16.10 Instrumentation requirements

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.4.1.16.11 Required CIM Elements

Table 168 - Required CIM Elements

Profile Classes & Associations	Notes
ElementSoftwareIdentity (7.3.4.1.16.12.2)	Associates PortController to SoftwareIdentity
ControlledBy (7.3.4.1.16.12.1)	
FCPort (7.3.4.1.16.12.3)	
LogicalPortGroup (7.3.4.1.16.12.4)	
MemberOfCollection (7.3.4.1.16.12.5)	
PortController (7.3.4.1.16.12.6)	The HBA.
ProtocolControllerForPort (7.3.4.1.16.12.7)	OPTIONAL.
SCSIProtocolController (7.3.4.1.16.12.8)	OPTIONAL.
SystemDevice (7.3.4.1.12.17)	Associates ComputerSystem and FCPort or SCSIProtocolController.
Packages	
Physical Package “package” (7.3.2.2)	
Software Subprofile (7.3.3.7)	
Associated Indications	

7.3.4.1.16.12 Required Properties for CIM Elements

7.3.4.1.16.12.1 ControlledBy

Table 169 - Required Properties for ControlledBy

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Antecedent	ref	key, override	SoftwareIdentity
Dependent	ref	key, override	PortController
InstanceID	string	required	
Version	string	required	
Manufacturer	string	required	
Classification	string	required	

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.4.1.16.12.2 ElementSoftwareIdentity

Table 170 - Required Properties for ElementSoftwareIdentity

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Antecedent	ref	key, override	SoftwareIdentity.
Dependent	ref	key, override	PortController.
InstanceID	string	required	
Version	string	required	
Manufacturer	string	required	
Classification	string	required	

7.3.4.1.16.12.3 FCPort

Table 171 - Required Properties for FCPort

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
SystemName	string	key	
SystemCreationClassName	string	key	
CreationClassName	string	key	
ElementName	**string**		Port Symbolic Name
OperationalStatus	uint16		
DeviceID	string	key, maxlen (64)	Opaque
Speed	uint64	units ("bits per second")	Speed of zero represents a link not established. 1 Gbit is 1 062 500 000 bit/s 2 Gbit is 2 125 000 000 bit/s 4 Gbit is 4 250 000 000 bit/s) 10 Gbit single channel variants are 10 518 750 000 bit/s 10 Gbit four channel variants are 12 750 000 000 bit/s This is the raw bit rate.
MaxSpeed	uint64		Port Supported Speed from HBA API.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table 171 - Required Properties for FCPort (Continued)

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
PortType	uint16		FC-GS Port.Type The specific mode currently enabled for the Port. The values: \"N\" = Node Port, \"NL\" = Node Port supporting FC arbitrated loop, \"E\" = Expansion Port connecting fabric elements (for example, FC switches), \"F\" = Fabric (element) Port, \"FL\" = Fabric (element) Port supporting FC arbitrated loop and \"B\" = Bridge Port. PortTypes are defined in ISO/ IEC 14165-414. When set to 1 (\"Other\"), the related property OtherPortType contains a string description of the port's type. PortType is defined to force consistent naming of the 'type' property in subclasses and to guarantee unique enum values for all instances of NetworkPort. A range of values, DMTF_Reserved, has been defined that allows subclasses to override and define their specific port types. Vendor Reserved = 16 000..65 535 can be used if the PortType is not one already defined in the above enumerations, and a vendor subclass is defined specifying the appropriate “value” and “valuemap”.
LinkTechnology	uint16		For FibreChannel, “FC”.
PermanentAddress	string	maxlen (64)	For FibreChannel, it is the Fibre Channel Port WWN.
NetworkAddresses[ ]	string	maxlen (64), arraytype ("indexed")	For Fibre Channel end device ports, it is the Fibre Channel ID. For Switches, it should be Null.
SupportedMaximumTransmissionUnit	uint16		

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.3.4.1.16.12.4 LogicalPortGroup

Table 172 - Required Properties for LogicalPortGroup

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
ElementName	string		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.
InstanceID	string	key	Opaque.
Name			Node WWN.
NameFormat	string		“WWN”.

7.3.4.1.16.12.5 MemberOfCollection

Table 173 - Required Properties of MemberOfCollection

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Collection	ref	key	The Collection that aggregates members.
Member	ref	key	The aggregated member of the Collection.

7.3.4.1.16.12.6 PortController

Table 174 - Required Properties for PortController

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
SystemCreationClassName	string	maxlen(256), key	The scoping System's CreationClassName.
SystemName	string	maxlen(256), key	The scoping System's Name.
CreationClassName	string	maxlen(256), key	The name of the concrete subclass.
DeviceID	string	maxlen(64), key	Opaque.
ProtocolSupported	uint16		The protocol used by the Controller to access 'controlled' Devices.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.4.1.16.12.7 ProtocolControllerForPort

Table 175 - Required Properties of ProtocolControllerForPort

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Dependent	ref		The Port.
Antecedent	ref		The protocol controller.

7.3.4.1.16.12.8 SCSIProtocolController

Table 176 - Required Properties for SCSIProtocolController

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
SystemCreationClassName	string	maxlen(256), key	The scoping System's CreationClassName.
SystemName	string	maxlen(256), key	The scoping System's Name.
CreationClassName	string	maxlen(256), key	The name of the concrete subclass.
DeviceID	string	maxlen(64), key	Opaque.
MaxUnitsControlled	uint32		Maximum number of directly addressable entities supported by this Controller. A value of 0 should be used if the number is unknown or unlimited.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.4.2 Switch Profile

7.3.4.2.1 Description

The switch profile models the physical and logical aspects of a Fibre Channel fabric interconnect element. The ComputerSystem class constitutes the core of the switch model. It is identified as a switch using the property Dedicated set to “switch”.

If a switch is modular, for instance if the switch is comprised of multiple blades on a backplane, LogicalModule can optionally be used to model each sub-module, and as an aggregation point for the switch ports.

FCPort describes the logical aspects of the port link and the data layers. PhysicalPackage models the physical aspects of a port. An instance of the FCPortStatistics class is expected for each instance of the FCPort class. FCPortStatistics expose real time port health and traffic information.

7.3.4.2.2 Standard Dependencies

The Switch Profile is based on the following standards:

Table 177 - Switch Standard Dependencies

Standard	Version	Organization
CIM Specification	2.2	DMTF
CIM Operations over HTTP	1.1	DMTF
CIM Schema	2.8 Preliminary	DMTF

7.3.4.2.3 Profile Dependencies

The Switch Profile requires the Server Profile (7.3.7).

7.3.4.2.4 CIM Server Requirements

7.3.4.2.4.1 Functional Profiles

Table 178 - Required Functional Profiles

Profile Required	Functional Group	Dependency
YES	Basic Read	None
NO	Basic Write	Basic Read
NO	Instance Manipulation	Basic Write
NO	Schema Manipulation	Instance Manipulation
YES	Association Traversal	Basic Read
NO	Query Execution	Basic Read
NO	Qualifier Declaration	Schema Manipulation
YES	Indication	None

7.3.4.2.4.2 Extrinsic Methods

The CIM Server shall support extrinsic methods for the Server profile.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.4.2.4.3 Discovery

The CIM Server shall support SLP discovery as defined in the CIM Operations over HTTP specification.

7.3.4.2.5 Instance Diagram

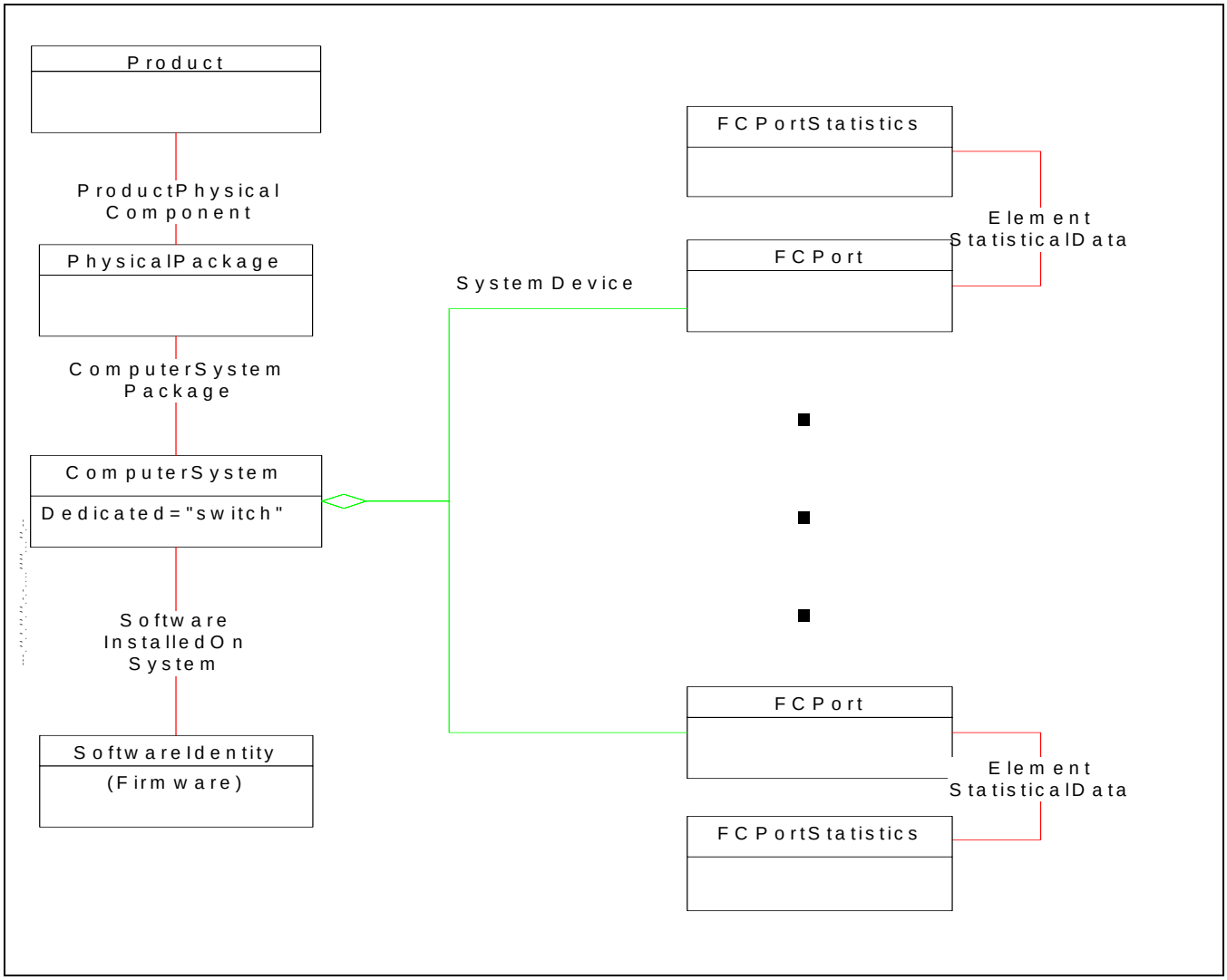


Figure 55 - Switch Instance Diagram

7.3.4.2.6 Durable Names and Correlatable IDs of the Profile

7.3.4.2.6.1 Durable Names Exported

See parent profile.

7.3.4.2.6.2 Correlatable IDs Used

See parent profile.

7.3.4.2.7 Methods

See parent profile.

7.3.4.2.8 Client Considerations

See parent profile

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.3.4.2.9 Recipes

See parent profile

7.3.4.2.10 Instrumentation Requirements

The information about the device that is supposed to be managed by the provider running on host (proxy agent) is implementation specific.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.4.2.11 Required CIM Elements

Table 179 - Required CIM Elements

Profile Classes & Associations	Notes
ComputerSystem (7.3.4.2.12.1)	
ElementStatisticalData (7.3.4.2.12.2)	
FCPort (7.3.4.2.12.3)	
FCPortRateStatistics (7.3.4.2.12.4)	
FCPortStatistics (7.3.4.2.12.5)	
SystemDevice (7.3.4.2.12.6)	
Packages	
Physical Package “package” (7.3.2.2)	
Software Package	
Associated Indications	
Creation/Deletion of ComputerSystem	SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_ComputerSystem SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_ComputerSystem
Change in status of ComputerSystem	SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_ComputerSystem AND SourceInstance.OperationalStatus <> PreviousInstance.OperationalStatus
Change in status of FCPort	SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_FCPort AND SourceInstance.OperationalStatus <> PreviousInstance.OperationalStatus

7.3.4.2.12 Required Properties for CIM Elements

7.3.4.2.12.1 ComputerSystem

Table 180 - Required Properties for ComputerSystem

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
ElementName	string		User Friendly name
OperationalStatus	uint16		
CreationClassName	string	maxlen(256), key	Name of Class
Name	string	maxlen(256), key	For Switches, it is the FC WWN.
NameFormat	string	override	“WWN”.
OtherIdentifyingInfo[ ]	string		The DomainID is stored here in decimal format.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table 180 - Required Properties for ComputerSystem (Continued)

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
IdentifyingDescription[ ]	string		“DomainID” is placed in the corresponding index.
Dedicated[ ]	int16		“Switch”.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.4.2.12.2 ElementStatisticalData

Table 181 - Required Properties for ElementStatisticalData

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
ManagedElement	ref		The reference to the FCPort
Stats	ref		The reference to the FCPortStatistics.

7.3.4.2.12.3 FCPort

Table 182 - Required Properties for FCPort

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
SystemName	string	key	
SystemCreationClassName	string	key	
CreationClassName	string	key	
ElementName	**string**		Port Symbolic Name
OperationalStatus	uint16		
DeviceID	string	key, maxlen (64)	Opaque
Speed	uint64	units ("bits per second")	Speed of zero represents a link not established 1 Gbit is 1 062 500 000 bit/s 2 Gbit is 2 125 000 000 bit/s 4 Gbit is 4 250 000 000 bit/s) 10 Gbit single channel variants are 10 518 750 000 bit/s 10 Gbit four channel variants are 12 750 000 000 bit/s This is the raw bit rate.
MaxSpeed	uint64		The max speed of the Port in Bits per Second using the same algorithm as Speed.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table 182 - Required Properties for FCPort (Continued)

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
PortType	uint16	override	FC-GS Port.Type The specific mode currently enabled for the Port. The values: \"N\" = Node Port, \"NL\" = Node Port supporting FC arbitrated loop, \"E\" = Expansion Port connecting fabric elements (for example, FC switches), \"F\" = Fabric (element) Port, \"FL\" = Fabric (element) Port supporting FC arbitrated loop, and \"B\" = Bridge Port. PortTypes are defined in the 14165-414. When set to 1 (\"Other\"), the related property OtherPortType contains a string description of the port's type. PortType is defined to force consistent naming of the 'type' property in subclasses and to guarantee unique enum values for all instances of NetworkPort. A range of values, DMTF_Reserved, has been defined that allows subclasses to override and define their specific port types. Vendor Reserved = 16000..65535 can be used if the PortType is not one already defined in the above enumerations and a vendor subclass is defined specifying the appropriate “value” and “valuemap”.
OtherNetworkPortType	string		Describes the type of module, when PortType is set to 1 (\"Other\").
PortNumber	uint16		NetworkPorts are often numbered relative to either a logical modules or a network element.
LinkTechnology	uint16		For FibreChannel, “FC”.
PermanentAddress	string	maxlen (64)	For FibreChannel, it is the Fibre Channel Port WWN.
NetworkAddresses[ ]	string	maxlen (64), arraytype ("indexed")	For Fibre Channel end device ports, it is the Fibre Channel ID. For Switches, it should be Null.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.4.2.12.4 FCPortRateStatistics

Table 183 - Required Properties for FCPortRateStatistics

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
InstanceID	unit16	key	
SampleInterval	datetime		This property is recommended
StatisticTime	datetime		This property is recommended
TxFrameRate	uint64		This property is recommended
RxFrameRate	uint64		This property is recommended
MaxTxFrameRate	uint64		This property is recommended
MaxRxFrameRate	uint64		This property is recommended
TxRate	uint64		This property is recommended
RxRate	uint64		This property is recommended
PeakTxRate	uint64		This property is recommended
PeakRxRate	uint64		This property is recommended

7.3.4.2.12.5 FCPortStatistics

Table 184 - Required Properties for FCPortStatistics

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
InstanceID	unit16	key	
StatisticTime	datetime		This property is recommended
ResetSelectedStats()			This property is recommended
BytesTransmitted	uint64		
BytesReceived	uint64		
PacketsTransmitted	uint64		
PacketsReceived	uint64		
LIPCount	uint64		This property is recommended
NOSCount	uint64		This property is recommended
ErrorFrames	uint64		This property is recommended
DumpedFrames	uint64		This property is recommended
LinkFailures	uint64		
LossOfSyncCounter	uint64		This property is recommended
LossOfSignalCounter	uint64		This property is recommended

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table 184 - Required Properties for FCPortStatistics (Continued)

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
PrimitiveSeqProtocolErrCount	uint64		
CRCErrors	uint64		
InvalidTransmissionWords	uint64		This property is recommended
FramesTooShort	uint64		This property is recommended
FramesTooLong	uint64		This property is recommended
AddressErrors	uint64		This property is recommended
BufferCreditNotProvided	uint64		This property is recommended
DelimiterErrors	uint64		This property is recommended
EncodingDisparityErrors	uint64		This property is recommended
LinkResetsReceived	uint64		This property is recommended
LinkResetsTransmitted	uint64		This property is recommended
MulticastFramesReceived	uint64		This property is recommended
MulticastFramesTransmitted	uint64		This property is recommended
FBSYFrames			This property is recommended
PBSYFrames			This property is recommended
FRJTFrames			This property is recommended
PRJTFrames			This property is recommended
RXClass1Frames			This property is recommended
TXClass1Frames			This property is recommended
Class1FBSY			This property is recommended
Class1PBSY			This property is recommended
Class1FRJT			This property is recommended
Class1PRJT			This property is recommended
RXClass2Frames			This property is recommended
TXClass2Frames			This property is recommended
Class2FBSY			This property is recommended
Class2PBSY			This property is recommended
Class2FRJT			This property is recommended
Class2PRJT			This property is recommended
RXClass3Frames			This property is recommended

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table 184 - Required Properties for FCPortStatistics (Continued)

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
TXClass3Frames			This property is recommended
Class3FramesDiscarded			This property is recommended
RXBroadcastFrames			This property is recommended
TXBroadcastFrames			This property is recommended
RxOLS			This property is recommended
TxOLS			This property is recommended
InvalidOrderedSets			This property is recommended

7.3.4.2.12.6 SystemDevice

Table 185 - Required Properties for SystemDevice

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
GroupComponent	ref	override	System Reference
PartComponent	ref	override	LogicalDevice Reference

7.3.4.2.13 Optional Subprofiles

Table 186 - Optional Profiles or Subprofiles

Name	Notes
Blades Subprofile (7.3.4.2.14)	

7.3.4.2.14 Blades Subprofile

7.3.4.2.14.1 Description

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

7.3.4.2.14.2 Standard Dependencies

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

7.3.4.2.14.3 Profile Dependencies

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

7.3.4.2.14.4 CIM Server Requirements

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.4.2.14.5 Instance Diagram

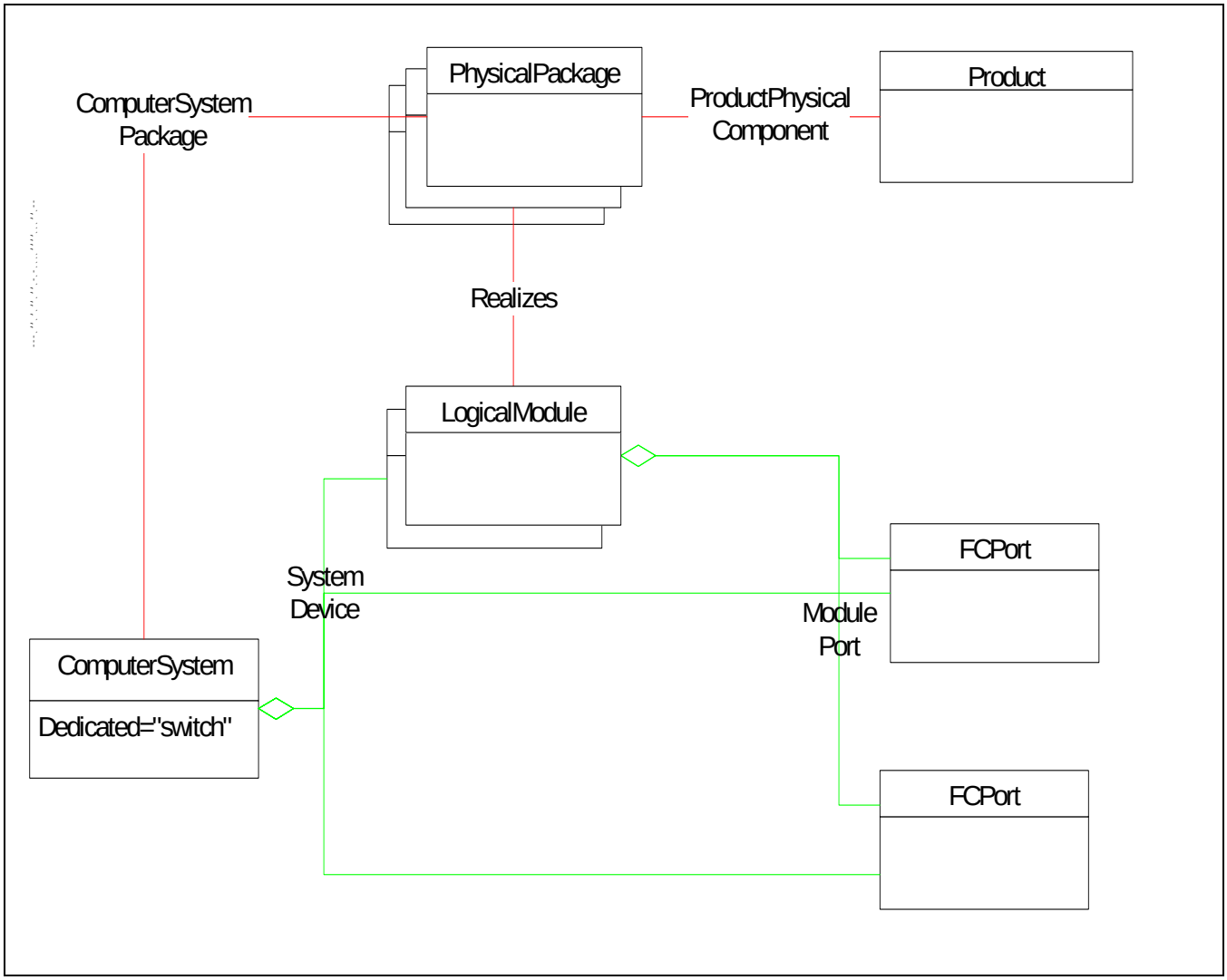


Figure 56 - Switch Blade Instance Diagram

7.3.4.2.14.6 Durable Names and Correlatable IDs

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

7.3.4.2.14.7 Methods

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

7.3.4.2.14.8 Client Considerations

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

7.3.4.2.14.9 Recipes

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.4.2.14.10 Instrumentation Requirements

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.3.4.2.14.11 Required CIM Elements

Table 187 - Required CIM Elements

Profile Classes & Associations	Notes
LogicalModule (7.3.4.2.14.12.1)	
ModulePort (7.3.4.2.14.12.2)	
Realizes	Associates LogicalModule to PhysicalPackage
SystemDevice (7.3.4.2.14.12.3)	
Packages	
Physical Package “package” (7.3.2.2)	
Associated Indications	
Creation/Deletion of LogicalModule.	“SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_LogicalModule” “SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_LogicalModule” These indicaitons are RECOMENDED.
Change in status of LogicalModule	“SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_LogicalModule AND SourceInstance.OperationalStatus <> PreviousInstance.OperationalStatus” This indicaiton is RECOMENDED.

7.3.4.2.14.12 Required Properties for CIM Elements

7.3.4.2.14.12.1 LogicalModule

Table 188 - Required Properties for LogicalModule

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
SystemName	string	key	
SystemCreationClassName	string	key	
CreationClassName	string	key	
ElementName	string		
OperationalStatus	uint16		
DeviceID	string	key, maxlen (64)	
ModuleNumber	uint16		

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.4.2.14.12.2 ModulePort

Table 189 - Required Properties for ModulePort

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
GroupComponent	ref	key	LogicalModule
PartComponent	ref	key	

7.3.4.2.14.12.3 SystemDevice

Table 190 - Required Properties for SystemDevice

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
GroupComponent	ref	override	System Reference
PartComponent	ref	override	LogicalDevice Reference

7.3.4.2.14.13 Optional Subprofiles

Table 191 - Optional Profiles or Subprofiles

Name	Notes
None	

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.3.5 Hosts

7.3.5.1 FC HBA Profile

7.3.5.1.1 Description

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.

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 (typically Card) through the Realizes association. If the HBA has logical operations that apply to the HBA and not to an individual port, then the PortController can be instantiated. The PortController is associated to the ComputerSystem through the SystemDevice association and associated to the ports through the ControlledBy association.

7.3.5.1.2 Standard dependencies

The FC HBA profile is based on the following standards:

Table 192 - HBA Standard Dependencies

Standard	Version	Organization
CIM Specification	2.2	DMTF
CIM Operations over HTTP	1.1	DMTF
CIM Schema	2.8	DMTF

7.3.5.1.3 Profile Dependencies

The FC HBA profile requires the Server Profile (7.3.7).

7.3.5.1.4 CIM Server Requirements

7.3.5.1.4.1 Functional Profiles

Table 193 - Required Functional Profiles

Profile Required	Functional Group	Dependency
YES	Basic Read	None
NO	Basic Write	Basic Read
NO	Instance Manipulation	Basic Write
NO	Schema Manipulation	Instance Manipulation
YES	Association Traversal	Basic Read
NO	Query Execution	Basic Read
NO	Qualifier Declaration	Schema Manipulation
YES	Indication	None

7.3.5.1.4.2 Extrinsic methods

The CIM Server shall support extrinsic methods for the Server profile.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.5.1.4.3 Discovery

The CIM Server shall support SLP discovery as defined in the CIM Operations over HTTP specification.

7.3.5.1.5 Instance diagrams

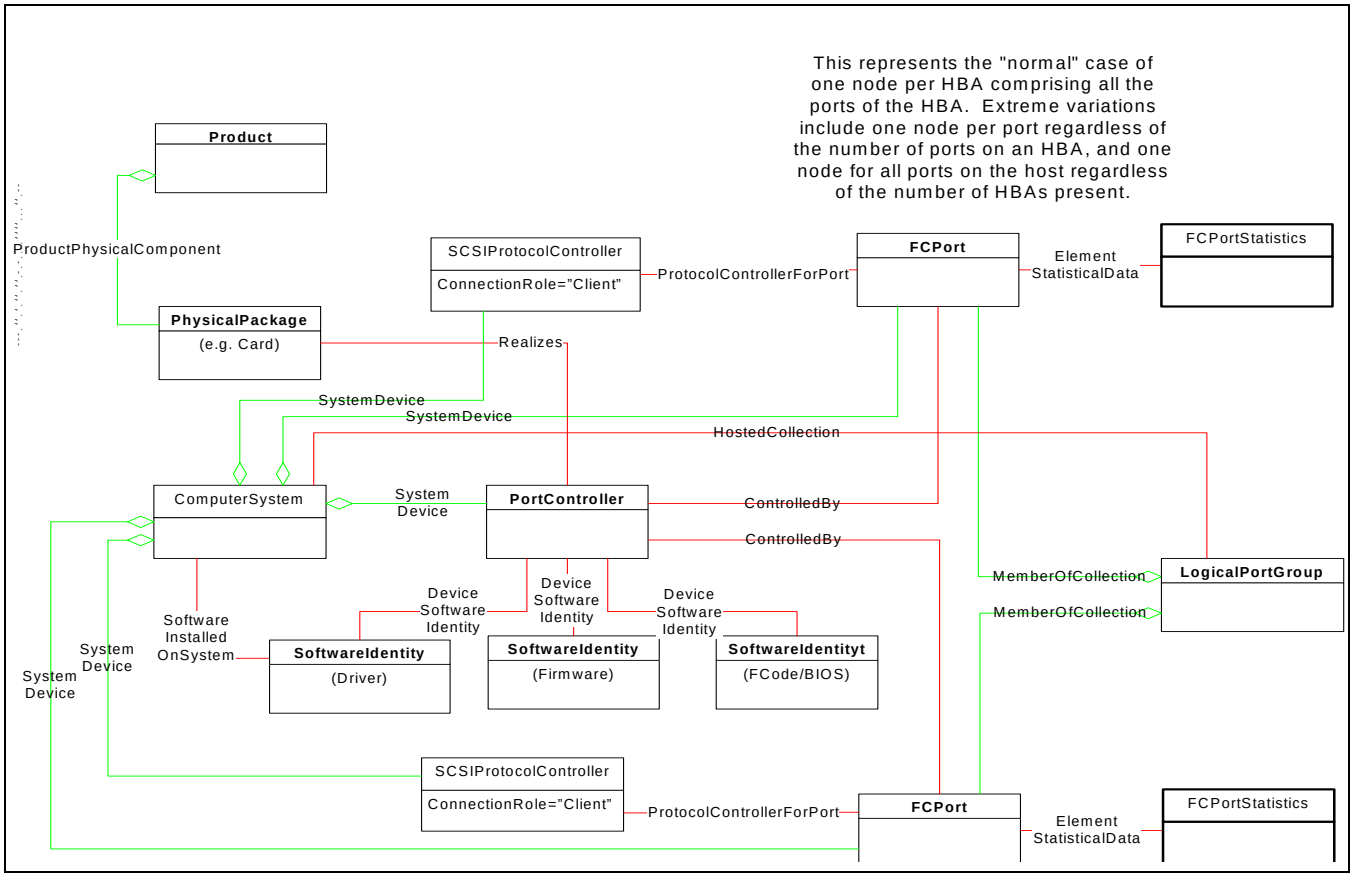


Figure 57 - FC HBA Instance Diagram

7.3.5.1.6 Durable names and correlatable IDs of the Profile

7.3.5.1.6.1 Durable names exported

For the Fibre Channel Port, the durable name is the Port WWN in FCPort.PermanentAddress.

7.3.5.1.6.2 Correlatable IDs used

There are no correlatable IDs defined for this profile.

7.3.5.1.7 Methods

There are no methods defined for this profile.

7.3.5.1.8 Client considerations

7.3.5.1.8.1 Multiple agents

The client does need to consider that there could be multiple SMI-S agents providing instances unrelated to what maybe provided on the Host system and may be unrelated to other SMI-S agents on the host.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.3.5.1.9 Recipes

7.3.5.1.9.1 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(
             $HBA->[#i],// ObjectName
             "CIM_Realizes",// AssocClass
             "CIM_PhysicalPackage",// ResultClass
             "Antecedent",// ResultRole
             "Dependent",// Role
             false,       // IncludeQualifiers
             false,       // IncludeClassOrigin
             {"Manufacturer", "Model"})// PropertyList
          // Exactly one PhysicalPackage MUST be returned
          if ($PhysicalPackage[] == null || $PhysicalPackage[].length == 0)
                        {
             <ERROR! Improper Physical Package information!>
          }
          // NOTE: The Product properties of interest are all Key qualified
          // properties, thus the instance name rather than the instance
                        itself
          // is retrieved.
          //
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
$Product->[] = AssociatorNames(
   $PhysicalPackage[0],// ObjectName
   “CIM_ProductPhysicalComponent”,// AssocClass
   “CIM_Product”,// ResultClass
   “GroupComponent”,// ResultRole
   “PartComponent”)// Role
// Exactly one PhysicalPackage MUST be returned
if ($Product->[] == null || $Product->[].length == 0) {
   <ERROR! Improper Product information!>
}
// Step 3. Determine the software (e.g. firmware, driver(s), BIOS,
// FCode) installed on the HBA.
//
#PropList = {“VersionString”, “Manufacturer”, “Classifications”}
$Software[] = Associators(
   $HBA->[#i],// ObjectName
   “CIM_ElementSoftwareIdentity”,// AssocClass
   “CIM_SoftwareIdentity”,// ResultClass
   “Antecedent”,// ResultRole
   “Dependent”,// Role
   false,       // IncludeQualifiers
   false,       // IncludeClassOrigin
   #PropList)// PropertyList
if ($Software[] != null && $Software[].length > 0) {
   for (#j in $Software[]) {
      // Retrieve relevant property instance data
             // These properties are not used in the recipe,
             // this just demonstrates how to locate this
             // information
      #VersionString = $Software[#j].VersionString
      #Manufacturer = $Software[#j].Manufacturer
      #Classifications[] = $Software[#j].Classifications
   }
}
// Step 4. Locate the Fibre Channel ports on the HBA and determine
// each port’s speed and WWN.
#PropList = {“Speed”, “PermanentAddress”}
$Ports[] = Associators(
   $HBA->[#i],// ObjectName
   “CIM_ControlledBy”,// AssocClass
   “CIM_FCPort”,// ResultClass
   “Dependent”,// ResultRole
   “Antecedent”,// Role
   false,    // IncludeQualifiers
   false,    // IncludeClassOrigin
   #PropList)// PropertyList
if ($Ports[] != null && $Ports[].length > 0) {
   for (#j in $Ports[]) {
      // Retrieve relevant Port instance data
      #Speed = $Ports[#j].Speed
      #PermanentAddress[] = $Ports[#j].PermanentAddress
```

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
                    // 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
                 }
              }
        }
```

7.3.5.1.10 Instrumentation requirements

There are no instrumentation requirements defined for this profile.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.5.1.11 Required CIM Elements

Table 194 - Required CIM Elements

Profile Classes & Associations	Notes
ComputerSystem (7.3.5.1.12.1)	
ControlledBy (7.3.5.1.12.2)	Optional.
ElementSoftwareIdentity (7.3.5.1.12.3)	
ElementStatisticalData (7.3.5.1.12.4)	Associates FCPort and FCPortStatistics.
FCPort (7.3.5.1.12.5)	
FCPortStatistics (7.3.5.1.12.6)	
HostedCollection (7.3.5.1.12.7)	Between ComputerSystem and LogicalPortGroup.
InstalledSoftwareIdentity (7.3.5.1.12.8)	Optional ; between ComputerSystem and the driver SoftwareIdentity instance.
LogicalPortGroup (7.3.5.1.12.9)	
MemberOfCollection (7.3.5.1.12.10)	Between FCPort and LogicalPortGroup.
PhysicalPackage (7.3.5.1.12.11)	
PortController (7.3.5.1.12.12)	Optional: the logical parts of the HBA.
Product (7.3.5.1.12.13)	
ProductPhysicalComponent (7.3.5.1.12.14)	
ProtocolControllerForPort (7.3.5.1.12.15)	Optional; mandatory for FCPorts.
Realizes (7.3.5.1.12.16)	
SCSIProtocolController (7.3.5.1.12.17)	Only required if the port supports SCSI (FCP).
SystemDevice (7.3.5.1.12.18)	Associates ComputerSystem and FCPort, PortController or SCSIProtocolController.
SoftwareIdentity (7.3.5.1.12.19)	Optional; represents driver, firmware or FCode/BIOS.
Packages	
none	
Associated Indications	
Creation of an FCPort	SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_FCPort.
Change in status of FCPort	SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_FCPort AND SourceInstance.OperationalStatus <> PreviousInstance.OperationalStatus.
Removal of an FCPort	SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_FCPort.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.5.1.12 Required properties for CIM Elements

7.3.5.1.12.1 ComputerSystem

Table 195 - Required Properties for ComputerSystem

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
CreationClassName	string	maxlen(256), key	Name of Class.
Name	string	maxlen(256), key	The name of the host, based on NameFormat.
NameFormat	string	required	In the FC HBA Profile, valid NameFormats are “IPAddressV4”, “IPAddressV6”, or “DNSName” .

7.3.5.1.12.2 ControlledBy

Table 196 - Required Properties for ControlledBy

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Antecedent	ref	key, override	The PortController.
Dependent	ref	key, override	The controlled FCPort.

7.3.5.1.12.3 ElementSoftwareIdentity

Table 197 - Required Properties for ElementSoftwareIdentity

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Antecedent	ref SoftwareIdentity	key	The SoftwareElement.
Dependent	ref ManagedElement	key	The LogicalDevice that requires or uses the software. In the FC HBA Profile, this refers to the PortController (or FCPort).

7.3.5.1.12.4 ElementStatisticalData

Table 198 - Required Properties for ElementStatisticalData

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
ManagedElement	ref		For the FC HBA Profile, this is the FCPort.
Stats	ref		For the FC HBA Profile, this is the FCPortStatistics.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.5.1.12.5 FCPort

Table 199 - Required Properties for FCPort

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
ElementName	**string**		Port Symbolic Name.
OperationalStatus[ ]	uint16		
DeviceID	string	key, maxlen (64)	Opaque.
Speed	uint64	units ("bits per second")	Speed of zero represents a link not established. 1 Gbit is 1 062 500 000 bit/s 2 Gbit is 2 125 000 000 bit/s 4 Gbit is 4 250 000 000 bit/s) 10 Gbit single channel variants are 10 518 750 000 bit/s 10 Gbit four channel variants are 12 750 000 000 bit/s This is the raw bit rate.
MaxSpeed	uint64		Port Supported Speed from HBA API.
PortType	uint16	override	“Unknown = 0, “Other” = 1, “N” = 10, “NL” = 11, “F/NL” = 12, “Nx” = 13, “E” = 14, “F” = 15, “FL” = 16, “B” = 17, “G” = 18.
OtherNetworkPortType	string		Describes the type of module, when PortType is set to 1 (\"Other\").
LinkTechnology	uint16		For FibreChannel, “FC”.
PermanentAddress	string	maxlen (64)	For FibreChannel, it is the Fibre Channel Port WWN.
NetworkAddresses[ ]	string	maxlen (64), arraytype ("indexed")	For Fibre Channel end device ports, it is the Fibre Channel ID. For Switches, it should be Null. This property is OPTIONAL.
ActiveMaximumTransmissionUnit	uint64		The active or negotiated maximum transmission unit (MTU) that can be supported. This property is OPTIONAL.
SupportedCOS	uint16[ ]		Port Supported Class of Service. This property is OPTIONAL.
ActiveFC4Types	uint16[ ]		

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.5.1.12.6 FCPortStatistics

Table 200 - Required Properties for FCPortStatistics

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
InstanceID	string	key, maxlen(64)	
ElementName	string		Suggestion: use FCPort's ElementName.
BytesTransmitted	uint64		From NetworkPortStatistics Superclass. Maps to HBA_PortStatistics,TxWords. Multiply word count by 4.
BytesReceived	uint64		From NetworkPortStatistics Superclass. Maps to HBA_PortStatistics.RxWords. Multiply word count by 4.
PacketsTransmitted	uint64		From NetworkPortStatistics Superclass. Maps to HBA_PortStatistics.TxFrames.
PacketsReceived	uint64		From NetworkPortStatistics Superclass. Maps to HBA_PortStatistics.RxFrames.
LIPCount	uint64		
NOSCount	uint64		
ErrorFrames	uint64		
DumpedFrames	uint64		
LinkFailures	uint64		Maps to HBA_PortStatistics.LinkFailureCount.
LossOfSyncCounter	uint64		Maps to HBA_PortStatistics.LossOfSynchCount.
LossOfSignalCounter	uint64		Maps to HBA_PortStatistics.LossOfSignalCount.
PrimitiveSeqProtocolErrCount	uint64		
CRCErrors	uint64		Maps to HBA_PortStatistics.InvalidCRCCount.
InvalidTransmissionWords	uint64		Maps to HBA_PortStatistics.InvalidTxWordCount.
StatisticTime	datetime		Optional; time last measurement was taken.

7.3.5.1.12.7 HostedCollection

Table 201 - Required Properties for HostedCollection

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Antecedent	ref	key, min(1), max(1)	ComputerSystem.
Dependent	ref	key, weak	LogicalPortGroup.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.5.1.12.8 InstalledSoftwareIdentity

Table 202 - Required Properties for InstalledSoftwareIdentity

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
System	ref	key	ComputerSystem.
SoftwareIdentity	ref	key	SoftwareIdentity.

7.3.5.1.12.9 LogicalPortGroup

Table 203 - Required Properties for LogicalPortGroup

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
ElementName	string		Node Symbolic Name.
InstanceID	string	key	Opaque.
Name			Node WWN.
NameFormat	string		“WWN”.

7.3.5.1.12.10 MemberOfCollection

Table 204 - Required Properties of MemberOfCollection

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Collection	ref	key	The Collection that aggregates members.
Member	ref	key	The aggregated member of the Collection.

7.3.5.1.12.11 PhysicalPackage

In this subprofile, the PhysicalPackage class represents the card (or possibly motherboard) containing the HBA.

PhysicalPackage is subclassed from PhysicalElement

Table 205 - Required Properties for PhysicalPackage

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
ElementName	string		User Friendly name. This property is OPTIONAL.
Name	string	maxlen(1024)	This property is OPTIONAL.
Tag	string	maxlen(256), key	An arbitrary string that uniquely identifies the Physical Element.
CreationClassName	string	maxlen(256). key	The name of the concrete subclass.
Manufacturer	string	maxlen(256)	

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table 205 - Required Properties for PhysicalPackage (Continued)

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Model	string	maxlen(64)	
SerialNumber	string	maxlen(64)	This property is OPTIONAL.
Version	string	maxlen(64)	This property is OPTIONAL.
Partnumber	string	maxlen(256)	This property is OPTIONAL.

7.3.5.1.12.12 PortController

Table 206 - Required Properties for PortController

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
SystemCreationClassName	string	maxlen(256), key	The scoping System's CreationClassName.
SystemName	string	maxlen(256), key	The scoping System's Name.
CreationClassName	string	maxlen(256), key	The name of the concrete subclass.
DeviceID	string	maxlen(64,key	Opaque.

7.3.5.1.12.13 Product

The Product represents the product attributes of the HBA itself.

Table 207 - Required Properties for Product

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
ElementName	string		User Friendly name. Suggested use is Vendor, Version and product name.
Name	string	key, maxlen(256)	Commonly used Product name.
IdentifyingNumber	string	key, maxlen(64)	Product identification such as a serial number on software, a die number on a hardware chip or (for non-commercial Products) a project number.
Vendor	string	key, maxlen(256)	The name of the Product's supplier or entity selling the Product (the manufacturer, reseller, OEM, etc.). Corresponds to the Vendor property in the Product object in the DMTF Solution Exchange Standard.
Version	string	key, maxlen(64)	Product version information.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.5.1.12.14 ProductPhysicalComponent

Indicates that the referenced PhysicalElement is acquired as part of a Product.

This association is used in SMI-S to associate a Product with its PhysicalPackage.

ProductPhysicalComponent is subclassed from Component.

Table 208 - Required Properties for ProductPhysicalComponent

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
GroupComponent	ref	override	The Product.
PartComponent	ref	override	The PhysicalElement that is a part of the Product.

7.3.5.1.12.15 ProtocolControllerForPort

Table 209 - Required Properties of ProtocolControllerForPort

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Dependent	ref		The Port ; in this profile, the cardinality shall be 1.
Antecedent	ref		The protocol controller; in this profile, the cardinality shall be 1.

7.3.5.1.12.16 Realizes

IN this profile, REalizes associates PortController and PhysicalPackage instances.

Realizes is subclassed from Dependency.

Table 210 - Required Properties for Realizes

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
PhysicalElement	ref	override	The physical component that implements the Device.
LogicalDevice	ref	override	The LogicalDevice

--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.3.5.1.12.17 SCSIProtocolController

Table 211 - Required Properties for SCSIProtocolController

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
SystemCreationClassName	string	maxlen(256), key	The scoping System's CreationClassName.
SystemName	string	maxlen(256), key	The scoping System's Name.
CreationClassName	string	maxlen(256), key	The name of the concrete subclass.
DeviceID	string	maxlen(64), key	Opaque.

7.3.5.1.12.18 SystemDevice

Table 212 - Required Properties for SystemDevice

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
GroupComponent	ref	override	System Reference.
PartComponent	ref	override	LogicalDevice Reference.

7.3.5.1.12.19 SoftwareIdentity

The SoftwareIdentity is used to model either software or firmware.

SoftwareIdentity is subclassed from LogicalElement.

Table 213 - Required Properties for SoftwareIdentity

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
InstanceID	string	key	The name used to identify this SoftwareIdentity.
VersionString	string		Software Version should be in the form <Major>.<Minor>.<Revision> or <Major>.<Minor><letter><revision>.
Manufacturer	string		Manufacturer of this software.
Classifications[ ]	uint16	required	“Driver”, “Firmware” or “FCode/BIOS”.
BuildNumber	uint16		OPTIONAL; the internal identifier for this compilation of software, if available.
RevisionNumber	uint16		OPTIONAL; this is the numeric representation of the revision number in the VersionString.
MajorVersion	uint16		OPTIONAL; this is the numeric representation of the Major number in the VersionString.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table 213 - Required Properties for SoftwareIdentity (Continued)

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
MinorVersion	uint16		OPTIONAL; this is the numeric representation of the Minor number in the VersionString.

7.3.5.1.13 Optional Subprofiles

Table 214 - Optional Profiles or Subprofiles

Name	Notes
None.	

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.3.6 Storage

7.3.6.1 Array Profile

7.3.6.1.1 Description

The Array model profile describes external RAID arrays and disk storage systems. The key classes are:

- Computer Systems that represent the array as a whole;
- Storage Volumes that are equivalent to SCSI Logical Units visible to consumers;
- StoragePools that are the allocatable/available space on the array;
- Fibre Channel ports through which the LUNs are made available.

The basic array profile provides a high level read-only ‘view’ of an array. The various subprofiles indicated in Figure 59 extend this description and also enable configuration of the array. Refer to 7.3.6.1.13, “Optional Subprofiles” for more information on these optional extensions. This profile also includes the mandatory Physical Package package, which describes the physical layout of the array and includes product identification information.

7.3.6.1.2 Standard Dependencies

The Array profile is based on the following standards:

Table 215 - Array Standard Dependencies

Standard	Version	Organization
CIM Specification	2.2	DMTF
CIM Operations over HTTP	1.1	DMTF
CIM Schema	2.8 Preliminary	DMTF

7.3.6.1.3 Profile Dependencies

The Array profile requires the Server Profile (7.3.7).

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.6.1.4 CIM Server Requirements

7.3.6.1.4.1 Functional Profiles

Table 216 - Required Functional Profiles

Profile Required	Functional Group	Dependency
YES	Basic Read	None
NO	Basic Write	Basic Read
NO	Instance Manipulation	Basic Write
NO	Schema Manipulation	Instance Manipulation
YES	Association Traversal	Basic Read
NO	Query Execution	Basic Read
NO	Qualifier Declaration	Schema Manipulation
YES	Indication	None

7.3.6.1.4.2 Extrinsic Methods

Although there are some extrinsic methods defined within classes in this profile, they are not needed for this Profile. However, sub profiles do require the use of Extrinsic methods.

7.3.6.1.4.3 Discovery

The CIM Server shall support SLP discovery as defined in the CIM Operations over HTTP specification.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.6.1.5 Instance Diagrams.

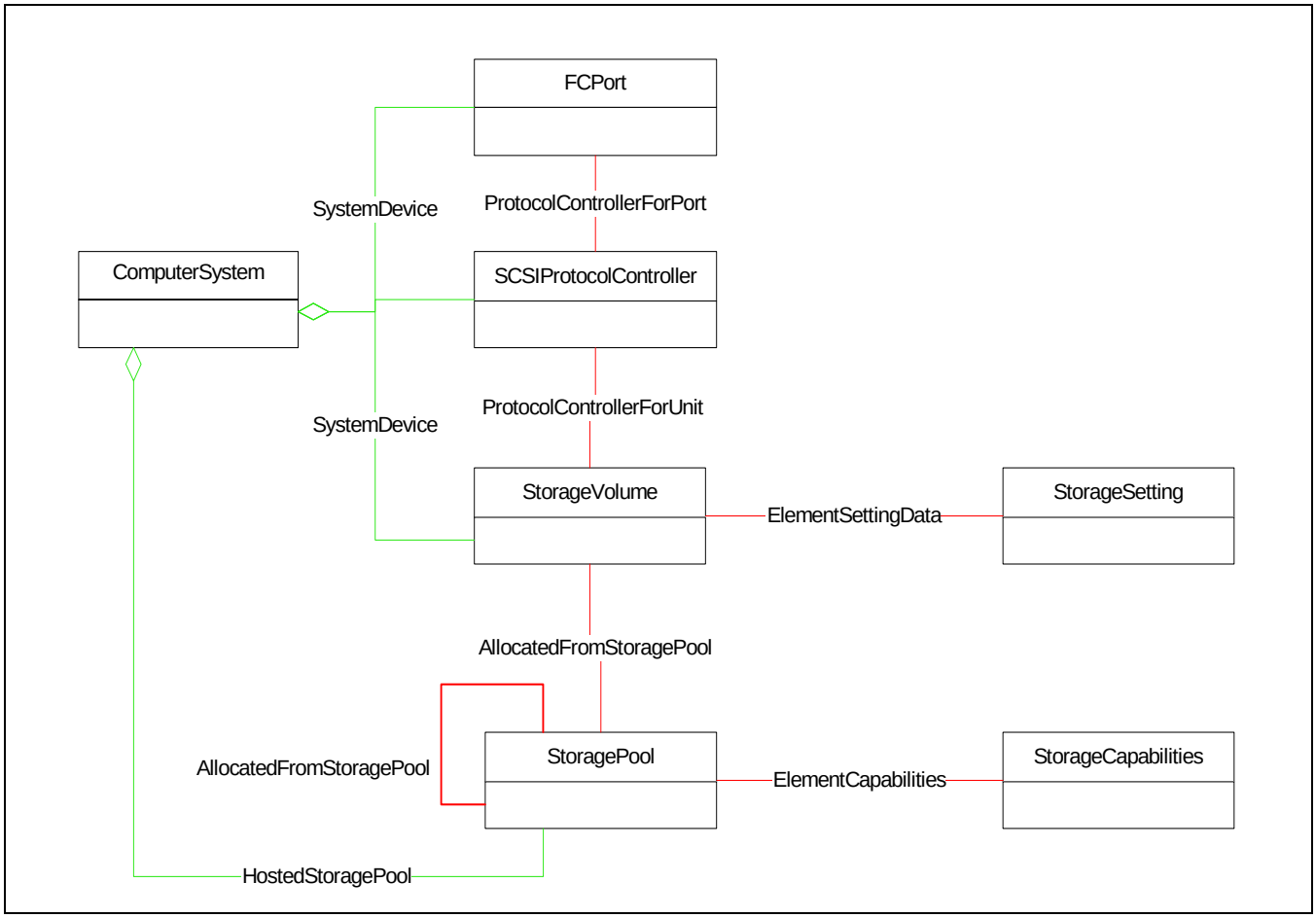


Figure 58 - Array Profile Instance Diagram

The main function of a disk array is to host storage and provide it to consumers for use. This is modeled in CIM using the concepts of 'Storage Pool' and 'Storage Volume'

Storage Pools

A StoragePool is an abstract notion of a blob of consumable storage space. A pool has certain ‘StorageCapabilities’, which indicate the range of 'Quality of Service' requirements that can be applied to objects created from the pool. In this top-level profile, StorageCapabilities are informational only. Refer to 7.3.3.10, “Pool Manipulation Capabilities and Settings Subprofile” for details on the use of these objects.

Storage pools are scoped relative to the ComputerSystem (indicated by the HostedStoragePool association). Objects created from a pool have the same scope.

Child objects (e.g. StorageVolumes or StoragePools) created from a StoragePool are linked back to the parent pool using an AllocatedFromStoragePool association.

There are two properties on StoragePool that describe the size of the ‘underlying’ storage. TotalManagedStorage describes the total raw storage in the pool and RemainingManagedStorage describes the storage currently remaining in the pool. RemainingManagedStorage plus AllocatedFromStoragePool.SpaceConsumed from all of the StorageVolumes allocated from the pool shall equal TotalManagedStorage.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Primordial Pool

The Primordial Pool is a specific instance of StoragePool. At least one shall always exists on the array to represent the unallocated storage on the storage device. The size of this Pool shall be equal to the total size of the unallocated, raw (unformatted or unprepared) storage. The Primordial property shall be true for Primordial Pools.

The use of the Primordial Pool is to determine the amount of unallocated space left on the array.

Storage Volumes

Storage Volumes are configured pieces of storage that shall be exposed from a system through an external interface. In the class hierarchy they are a sub class of a StorageExtent. In SCSI terms, they are Logical Units.

Storage Volumes are created from Storage Pools using the Storage Configuration Service (see 7.3.3.11, “LUN Creation Subprofile”).

In this profile, a StorageSetting is informational only. Refer to 7.3.3.11, “LUN Creation Subprofile” for details on StorageSettings

7.3.6.1.6 Durable Names and Correlatable IDs of the Profile

7.3.6.1.6.1 Durable Names Exported

For StorageVolume, the durable name is the Name property. The format of this property is available in NameFormat. The valid formats are described in Section ‘Find the Durable Name for Volumes (7.3.6.1.6.2)

For Fibre Channel port, the durable name is the Port WWN. It is found in FCPort.PermanentAddress.

For the array itself (the computer system), the Name property contains a durable name. The format of this name is defined by the NameFormat property.

7.3.6.1.6.2 Find the Durable Name for Volumes

Different implementation use different approaches to uniquely identify SCSI units (Logical Devices). The agent is encouraged to try these standard interfaces in this order to find a durable volume name. The best name is put in the StorageVolume Name field. The NameFormat attribute of LogicalDevice (and subclasses) identifies how the name field is generated. A client that uses the same name format will help assure a consistent model.

“Inquiry-VPD page 83 data” is documented in the SCSI Primary Commands specification. It allows a device to report a list of identifiers in a variety of formats. Identifier type 3 as specified in ANSI X3.320-1994 is used for device identification. The ANSI Name Address Authority (NAA) specifies the format. When “association” is set to 0, the ID represents the logical device rather than a single port. NAA specifies that high order 4 bits define the format used in the rest of the identifier. Other NAA values and identifiers types may be used in older implementations. If the volume does not report VPD page 83, VPD page 80 is a serial number; this value shall be merged with vendor and model strings from standard inquiry to generate a unique ID. Some vendors store a serial number in the vendor-specific data in the standard inquiry data. The last option is a Fibre Channel WWN that may map 1-1 to a device in JBOD configurations.

Refer to Table 3, “Standardized Name Formats,” for a complete list of the name formats recognized by SMI-S.

7.3.6.1.6.3 Correlatable IDs Used

None.

7.3.6.1.7 Methods

None.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.6.1.8 Client Considerations

7.3.6.1.8.1 Connectivity Considerations

The description and figures describe an implementation with Fibre Channel connectivity. In order to support other transports in a future revision of the specification, clients should use LogicalPort rather than FCPort in CIM Operations. LogicalPort.PermanentAddress continues to serve as a correlatable ID for ports, but it is not necessarily an FC WWN.

7.3.6.1.8.2 Find Port Information

Ports shall be aggregated from ComputerSystems using SystemDevice. In an array with multiple storage processors, ports shall be aggregated from the component ComputerSystem; this aggregation allows a client to see which ports are associated with a particular processor and to understand possible single points of failure.

If the Array has Fibre Channel connectivity, the FCPort shall include ProtocolControllerForPort associations to SCSIProtocolController to indicate the SCSI device for the port. SCSIProtocolController shall include ProtocolControllerForUnit associations to exposed StorageVolumes.

SCSIProtocolControllers can serve as initiators (for example, a port in an HBA) or as targets (ports in devices). A RAID array model may include both; they can be differentiated in two ways:

- SCSIProtocolController.ConnectionRole can be ‘Client’ (initiator) or ‘Server’ (target).
- The ProtocolControllerAccessesUnit association indicates a initiator to LU relationship.
- The ProtocolControllerForUnit association indicates a target/LU relationship.

7.3.6.1.8.3 Find System Status

The ‘OperationalStatus’ property is available on most objects in the model and is used to indicate it’s status. For the whole array, the ComputerSystem instance shall have one of the following Main Operational Status values and possibly one of the Subsidiary status values. The main OperationalStatus shall be the first element in the array.

Table 217 - OperationalStatus for ComputerSystem

Main Operational Status	Possible Subsidiary Operational Status	Description
OK		The system has a good status
OK	Stressed	The system is stressed, for example the temperature is over limit or there is too much IO in progress
OK	Predictive Failure	The system will probably fail sometime soon
Degraded		The system is operational but not at 100% redundancy. A component has suffered a failure or something is running slow
Error		An error has occurred causing the system to stop. This error may be recoverable with operator intervention.
Error	Non-recoverable error	A severe error has occurred. Operator intervention is unlikely to fix it

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table 217 - OperationalStatus for ComputerSystem (Continued)

Main Operational Status	Possible Subsidiary Operational Status	Description
Error	Supporting entity in error	A modeled element has failed
No contact		The provider knows about the array but has not talked to it since last reboot
Lost communication		The provider used to be able to communicate with the array, but has now lost contact.
Starting		The array is starting up
Stopping		The array is shutting down.
Stopped		The data path is OK but shut down, the management channel is still working.

A client has the option of obtaining Asynchronous notification of changes in status through CIM_InstModification. More details on indications are in 7.2.5, “Events – CIM indications”.

7.3.6.1.8.4 Find Volume Status

The status of a volume can be determined by looking at the values in the OperationalStatus and ExtentStatus properties. Table 218 describes their possible states. ExtentStatus provides further clarification of the main OperationalStatus.

The status described in Table 218 shall be supported for StorageVolume.OperationalStatus and StorageVolume.ExtentStatus. ExtentStatus provides a further clarification of the main OperationalStatus. The main OperationalStatus shall be the fist element in the array.

Table 218 - OperationalStatus for StorageVolume

Operational Status	ExtentStatus	Description
OK		The volume has good status
Degraded		The volume is operating in a degraded mode. There may have been a loss of redundancy.
Degraded	Spare In Use	The spare has been put into use. No more spares are available.
Degraded	Rebuild	A spare is in use and a rebuild is in process.
Error		The volume is not functioning
Error	Broken	The volume is not functioning but there is no confirmed data loss
Error	Data Lost	The volume has broken and there is data loss
Starting		The volume is in process of initialization
Dormant		The volume is offline

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.6.1.8.5 Find Port Status

The status of a FCPort is determined by the value of the OperationalStatus property. Table 219 shows the allowed values for this property and their meanings. The table defines the possible states that shall be supported for FCPort.OperationalStatus. The main OperationalStatus shall be the first element in the array.

Table 219 - Port State/Status

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

7.3.6.1.9 Recipes

7.3.6.1.9.1 Overview

The following recipes show examples of how a client navigates the model and determine information from the basic array profile. A compliant SMI-S server implementation shall support them. For details on the Pseudo code syntax please refer to 7.2.7, “Recipe Conventions”.

7.3.6.1.9.2 Summarize the Pools on an array.

```
// DESCRIPTION
// This recipe works out the following:
//   The overall size of the array, by summarizing the
//      TotalManagedSpace for the primordial pools.
//   The consumed space on the array. This is worked out by summing
//      the space consumed by volumes.
//   The Unallocated space on the array is the TotalManagedSpace -
//      consumed space.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTIONS
// 1.The object name for the device, CIM_ComputerSystem, of interest
//    has been previously identified and defined in the $Array->
//   variable
// 2.The class definitions have been retrieved via a previous
//   EnumerateClasses call so it is possible to set includeQualifiers
//   to false on the associators call for higher performance.

// first find the pools...
$Pools[] = Associators(
     $Array->,
     “CIM_HostedStoragePool”,
     “CIM_StoragePool”,
     “GroupComponent”,
     “PartComponent”,
     false,
     false,
```

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
     $Properties[] {“TotalManagedSpace”,”Primordial”});

// Then cycle through them and add up the managed space from the
                            primordial pools.
#managedSpace = 0
#allocSpace = 0

for #i in $Pools->[]
{
     if ($Pools->[#i]->Primordial)
        #managedSpace = managedSpace + $Pools->[#i].TotalManagedSpace
     $Allocs[] = references(
        $Pools->[#i],
        “CIM_AllocatedFromStoragePool”,
        “Antecedent”,
        false,
        false,
        null)

     for #j in $Allocs
     {
        if ($Allocs[#j].Antecedent ISA “CIM_StorageVolume”)
           #allocSpace = #allocSpace + $Allocs[#j].SpaceConsumed
     }
}



// managedSpace is the total managed space and #allocSpace is the total
                            allocated space.
// With these variables the client can determine the storage space
                            reserved by pools and used
// in the allocation of volumes in respect to the total storage possible
                            in this Array
```

7.3.6.1.9.3 List volume information

```
// DESCRIPTION
// This recipe will determine and access the name and volume size used
// for each volume in the array.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTIONS
// 1.The object name for the device, CIM_ComputerSystem, of interest
//   has been previously identified and defined in the $Array->
//   variable
// 2.The class definition have been retrieved so it is possible to
//   set includeQualifiers to false on the associators call.
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
// first find the vols ....
$Vols[] = Associators(
     $Array->,
     “CIM_SystemDevice”,
     “CIM_StorageVolume”,
     “GroupComponent”,
     “PartComponent”,
     false,
     false,
     $Properties[] {“ElementName”, “Name”, “NameFormat”,
        “BlockSize”, “NumberOfBlocks”});

// Then cycle through and use Volume information
for #i in $Vols[]
{
     #VolumeDescription $Vols[#i].ElementName
     #VolumeName = $Vols[#i].Name
     #NameFormat = $Vols[#i].NameFormat
     #VolumeSize = $Vols[#i].BlockSize * $Vols[#i].NumberOfBlocks
}
```

7.3.6.1.9.4 List LUN information

```
// DESCRIPTION
// For each volume, get the LUN & FC Port information.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTIONS
// 1.Have a list of Volumes from the previous recipe.
// 2.Know the class definition so can set includeQualifiers false on
//   the associators call.

// FUNCTION: findUnit
// return the association instance from $pcfu which has #Pc as the
// Antecedent
sub CIMInstance findUnit (int #Pc, CIMInstance$pcfu[] )
{
     for #i in $pcfu
     {
        if (compare (#i.antecedent, #pc->))
           return #i
     }
     Error(“can’t find relevent associator”)
}


// MAIN
#first find the vols ...
for #i in $Vols[]
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
{
     $ProCont[] = Associators(
        $#i->,
        “ProtocolControllerForUnit”,
        “CIM_ProtocolController”,
        “Dependent”,
        “Antecedent”);
     $pcfu[] = References(
        $#i->,
        “ProtocolControllerForUnit”,
        “CIM_ProtocolController”,
        “Dependent”,
        false,
        false);
     for #y in $ProCont
     {
        $Ports[] = AssociatorsNames(
           $#y->,
           “CIM_ProtocolControllerForPort”,
           “CIM_LogicalPort”,
           “Antecedent”,
           “Dependent”);
        #luninfo = findUnit($y,$pcfu[])

        // The following variables now contain the information
        // “ LUN“ = #luninfo.DeviceNumber
        // “Volume“ = #i.ElementName
     }
}
```

7.3.6.1.10 Instrumentation Requirements

There are no instrumentation requirements defined for this profile.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.6.1.11 Required CIM Elements

Table 220 - Required CIM Elements

Profile Classes & Associations	Notes
AllocatedFromStoragePool (7.3.6.1.12.1)	
ElementCapabilities (7.3.6.1.12.2)	
ElementSettingData (7.3.6.1.12.3)	
ComputerSystem (7.3.6.1.12.4)	
FCPort (7.3.6.1.12.5)	
HostedStoragePool (7.3.6.1.12.6)	
ProtocolControllerForPort (7.3.6.1.12.7)	OPTIONAL; mandatory for FCPorts.
ProtocolControllerForUnit (7.3.6.1.12.8)	
SCSIProtocolController (7.3.6.1.12.9)	
StorageCapabilities (7.3.6.1.12.10)	
StoragePool (7.3.6.1.12.11)	
StorageSetting (7.3.6.1.12.12)	
StorageVolume (7.3.6.1.12.13)	
SystemDevice (7.3.6.1.12.14)	
Packages	
Physical Package “package” (7.3.2.2)	
Software Package (7.3.2.3)	
Associated Indications	
Creation/Deletion of an Array	SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_ComputerSystem SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_ComputerSystem
Change in operational status of a Volume	SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_StorageVolume AND SourceInstance.OperationalStatus <> PreviousInstance.OperationalStatus
Change in operational status of an FCPort	SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_FCPort AND SourceInstance.OperationalStatus <> PreviousInstance.OperationalStatus
Change in operational status of an Array	SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_ComputerSystem AND SourceInstance.OperationalStatus <> PreviousInstance.OperationalStatus

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.6.1.12 Required Properties for CIM Elements

7.3.6.1.12.1 AllocatedFromStoragePool

Table 221 - Required Properties for AllocatedFromStoragePool

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Antecedent	ref		The StoragePool.
Dependent	ref		The sub pool or volume.
SpaceConsumed	uint64		Space Consumed from this Pool (in bytes).

7.3.6.1.12.2 ElementCapabilities

Table 222 - Required Properties for ElementCapabilities

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Capabilities	ref		The StorageCapabilities instance.
ManagedElement	ref		The object to which the capabilities apply.

7.3.6.1.12.3 ElementSettingData

Table 223 - Required Properties for ElementSettingData

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
SettingData	ref		The StorageSetting instance.
ManagedElement	ref		The StorageVolume to which the storagesetting applies.

7.3.6.1.12.4 ComputerSystem

Table 224 - Required Properties for ComputerSystem

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
ElementName	string		User Friendly name.
OperationalStatus	uint16[ ]		Status of array.
CreationClassName	string	required, key	
Name	string	key	The identifier for the Array (e.g. IP address).
NameFormat	string		The format of the Name property shall be one of: IP, WWN, Other.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table 224 - Required Properties for ComputerSystem (Continued)

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Dedicated[ ]	int16	“blockserver”, “storage”	For this profile Dedicated will always include these two values.
PrimaryOwnerContact	string		Optional
PrimaryOwnerName	string		Optional

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.3.6.1.12.5 FCPort

Table 225 - Required Properties for FCPort

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
SystemCreationClassName	string	key	
SystemName	string	key	
CreationClass	string	key	
ElementName	string		User friendly name/caption for port.
OperationalStatus[ ]	uint16		Status of device
DeviceID	string	key	Opaque
PortType	uint16		FC-GS Port.Type The specific mode currently enabled for the Port. The values: \"N\" = Node Port, \"NL\" = Node Port supporting FC arbitrated loop, \"E\" = Expansion Port connecting fabric elements (for example, FC switches), \"F\" = Fabric (element) Port, \"FL\" = Fabric (element) Port supporting FC arbitrated loop, and \"B\" = Bridge Port. PortTypes are defined in the ISO/IEC 14165-414. When set to 1 (\"Other\"), the related property OtherPortType contains a string description of the port's type. PortType is defined to force consistent naming of the 'type' property in subclasses and to guarantee unique enum values for all instances of NetworkPort. A range of values, DMTF_Reserved, has been defined that allows subclasses to override and define their specific port types. Vendor Reserved = 16 000..65 535 can be used if the PortType is not one already defined in the above enumerations and a vendor subclass is defined specifying the appropriate “value” and “valuemap”.
UsageRestriction	uint16		Front-end only.
PermanentAddress	string		The WWN of the port.
NetworkAddresses[ ]	string		The Fibre Channel address of the port. This property is OPTIONAL.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table 225 - Required Properties for FCPort (Continued)

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Speed	uint64		Speed of zero represents a link not established. 1 Gbit is 1 062 500 000 bit/s 2 Gbit is 2 125 000 000 bit/s 4 Gbit is 4 250 000 000 bit/s) 10 Gbit single channel variants are 10 518 750 000 bit/s 10 Gbit four channel variants are 12 750 000 000 bit/s This is the raw bit rate. This property is OPTIONAL.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.3.6.1.12.6 HostedStoragePool

Table 226 - Required Properties from HostedStoragePool

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
PartComponent	ref		The storage pool.
GroupComponent	ref		The scoping system.

7.3.6.1.12.7 ProtocolControllerForPort

Table 227 - Required Properties from ProtocolControllerForPort

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Antecedent	ref		The SCSIProtocolController for this port.
Dependent	ref		The port.
AccessPriority	unit16		The priority of access through this port for this ProtocolController. This property is OPTIONAL.

7.3.6.1.12.8 ProtocolControllerForUnit

Table 228 - Required Properties from ProtocolControllerForUnit

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Antecedent	ref		The protocol controller.
Dependent	ref		The exposed logical unit.
DeviceNumber	unit16		The Logical Unit number for this Volume through this controller. Formatted as unseparated uppercase hexadecimal digits, with no leading “0x”.

7.3.6.1.12.9 SCSIProtocolController

Table 229 - Required Properties for SCSIProtocolController

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
SystemCreationClassName	string	key	
SystemName	string	key	
CreationClass	string	key	
ElementName	**string**		User friendly name/caption for port. This property is OPTIONAL.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table 229 - Required Properties for SCSIProtocolController (Continued)

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
OperationalStatus[ ]	uint16		Status of device. This property is OPTIONAL.
DeviceID	string	key	Opaque
MaxUnitsControlled	uint32		Maximum number of units controlled by this controller. This property is OPTIONAL.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.6.1.12.10 StorageCapabilities

Table 230 - Required Properties from StorageCapabilities

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
InstanceID	string	key	Opaque identifier.
ElementName	string		User friendly name/ caption.
ElementType	uint16		Type of element this capability applies to.
NoSinglePointOfFailure	boolean		
NoSinglePointOfFailureDefault	boolean		
DataRedundancyMin	uint16		
DataRedundancyMax	uint16		
DeltaReservationDefault	uint16		
DeltaReservationMin	uint16		
DeltaReservationMax	uint16		
DataRedundancyDefault	uint16		
PackageRedundancyMin	uint16		
PackageRedundancyMax	uint16		
PackageRedundancyDefault	unit16		

7.3.6.1.12.11 StoragePool

Table 231 - Required Properties for StoragePool

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
ElementName	string		User Friendly name.
InstanceID	string	key	Opaque identifier.
PoolID	string	required	A unique name in the context of the System that identifies this pool.
TotalManagedSpace	uint64		
RemainingManagedSpace	unit64		
Primordial	boolean		Defaults to false, true for the primordial pools.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.6.1.12.12 StorageSetting

Table 232 - Required Properties from StorageSetting

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
InstanceID	string	key	Opaque identifier
ElementName	string	required	User friendly name/caption
DataRedundancyMin	uint16		
DataRedundancyMax	uint16		
DataRedundancyGoal	uint16		
DeltaReservationGoal	uint16		
DeltaReservationMin	uint16		
DeltaReservationMax	uint16		
PackageRedundancyMin	uint16		
PackageRedundancyMax	uint16		
PackageRedundancyGoal	unit16		

7.3.6.1.12.13 StorageVolume

Table 233 - Required Properties for StorageVolume

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
SystemCreationClassName	string	key	
SystemName	string	key	
CreationClass	string	key	
ElementName	string		User Friendly name.
Name	string		VPD Page 83 ID.
NameFormat	uint16		Format of Name property.
ExtentStatus	uint16[ ]		Status of volume (Rebuild,spare in use etc).
OperationalStatus	unit16[ ]		Current general status of volume.
DeviceID	string	key	Opaque ID.
BlockSize	uint64		Block size of Volume.
NumberOfBlocks	uint64		NUmber of Blocks (not size of volume is BlockSize* NumberOFBlocks).
IsBasedOnUnderlyingRedundancy	boolean		

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table 233 - Required Properties for StorageVolume (Continued)

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
NoSinglePointOfFailure	boolean		Current value of StorageSetting.
DataRedundancy	uint16		Current value of StorageSetting.
PackageRedundancy	uint16		Current value of StorageSetting.
DeltaReservation	uint16		Current value of StorageSetting.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.6.1.12.14 SystemDevice

Table 234 - Required Properties for SystemDevice

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
GroupComponent	ref		System Reference.
PartComponent	ref		LogicalDevice Reference.

7.3.6.1.13 Optional Subprofiles

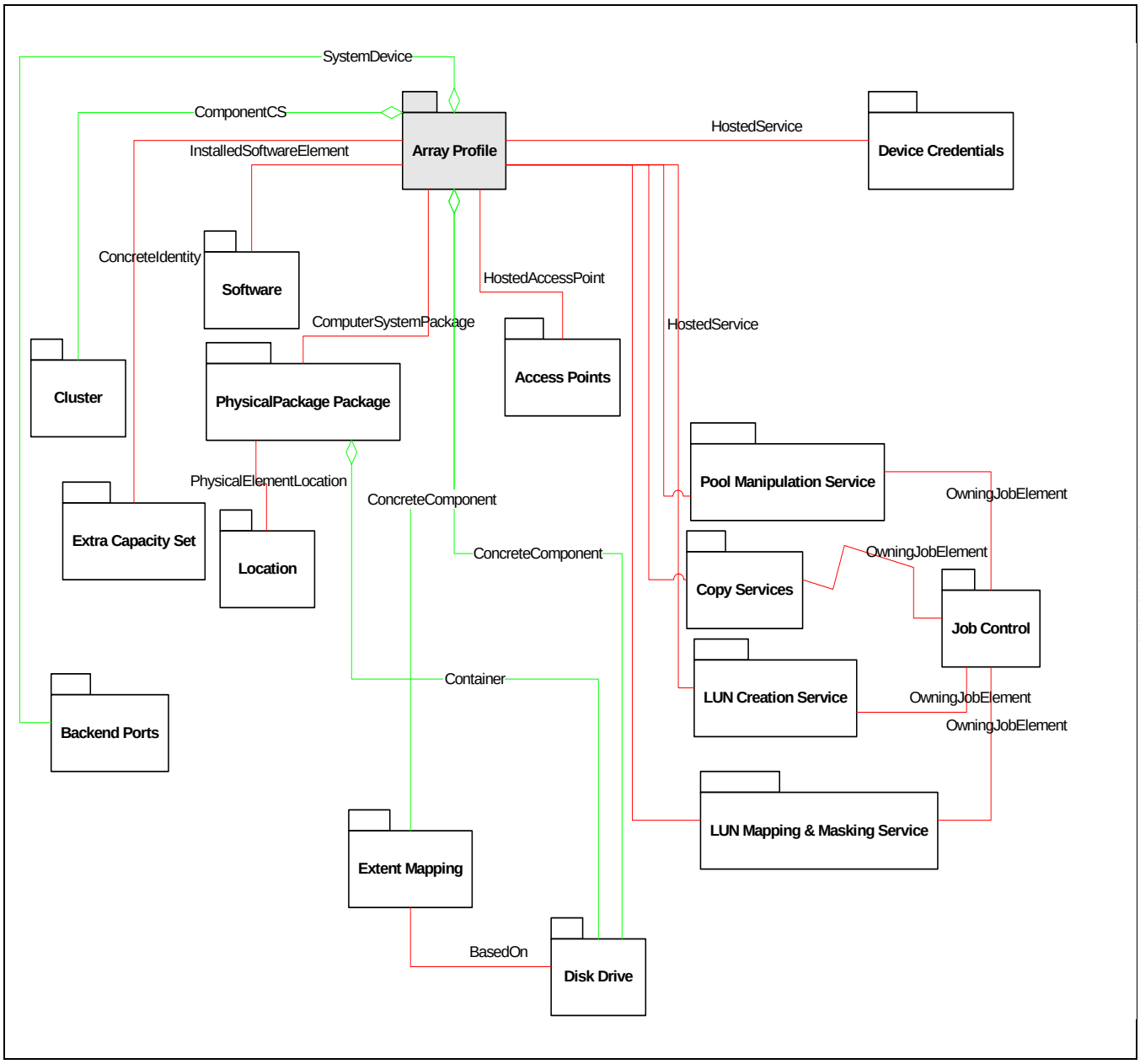


Figure 59 - Array Packages Diagram

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table 235 - Optional Profiles or Subprofiles

Name	Notes
Access Points subprofile (7.3.3.2)	
Cluster Subprofile (7.3.3.3)	
Disk Drive subprofile (7.3.3.4)	
Extent Mapping subprofile (7.3.3.5)	
Location subprofile (7.3.3.6)	
Software Subprofile (7.3.3.7)	
Copy Services Subprofile (7.3.3.8)	
Job Control Subprofile (7.3.3.9)	
Pool Manipulation Capabilities and Settings Subprofile (7.3.3.10)	
LUN Creation (7.3.3.11)	
Device Credentials subprofile (7.3.3.12)	
LUN Masking and Mapping (7.3.3.14)	
Sparing Subprofile (B.2.1)	
Backend Ports subprofile (7.3.3.13)	

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.6.2 In-Band Virtualization Profile

7.3.6.2.1 Description

An in-band virtualization system uses storage provided by array controllers to create a seamless pool of storage. The virtualization system in turn allocates volumes from the pool for host systems to use. The system sits between two fabrics. The first fabric contains the array systems used by the virtualization system. The second fabric connects the virtualization system to the hosts' systems.

The basic Virtualization System profile provides a read-only view of the system. The various subprofiles indicated in Figure 60 extend this description and also enable configuration. Refer to 7.3.6.2.13, "Optional Subprofiles" for more information on these optional extensions. This profile also includes the mandatory Physical Package package, which describes the physical layout of the system and includes product identification information.

7.3.6.2.2 Standard dependencies

The In-band Virtualizer profile is based on the following standards:

Table 236 - In-Band Virtualizer Standard Dependencies

Standard	Version	Organization
CIM Specification	2.2	DMTF
CIM Operations over HTTP	1.1	DMTF
CIM Schema	2.8 Preliminary	DMTF

7.3.6.2.3 Profile dependencies

The In-band Virtualizer profile requires the Server Profile (7.3.7).

7.3.6.2.4 CIM Server Requirements

7.3.6.2.4.1 Functional Profiles

Table 237 - Required Functional Profiles

Profile Required	Functional Group	Dependency
YES	Basic Read	None
NO	Basic Write	Basic Read
NO	Instance Manipulation	Basic Write
NO	Schema Manipulation	Instance Manipulation
YES	Association Traversal	Basic Read
NO	Query Execution	Basic Read
NO	Qualifier Declaration	Schema Manipulation
YES	Indication	None

7.3.6.2.4.2 Extrinsic methods

Support for extrinsic methods is optional for the InBandVirtualizer profile.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.6.2.4.3 Discovery

The CIM Server shall support SLP discovery as defined in the CIM Operations over HTTP specification.

7.3.6.2.5 Instance diagrams

7.3.6.2.5.1 Overview

The modelling in this document is split into various sections that describe how to model particular elements of an In-Band Virtualization System. The diagrams used in this document are 'Instance' diagrams implying the actual classes that you implement rather than the class hierarchy diagrams often used to show CIM models. This is felt to be easier to understand. Please refer to the CIM Schema for information on class inheritance information and full information on the properties and methods used.

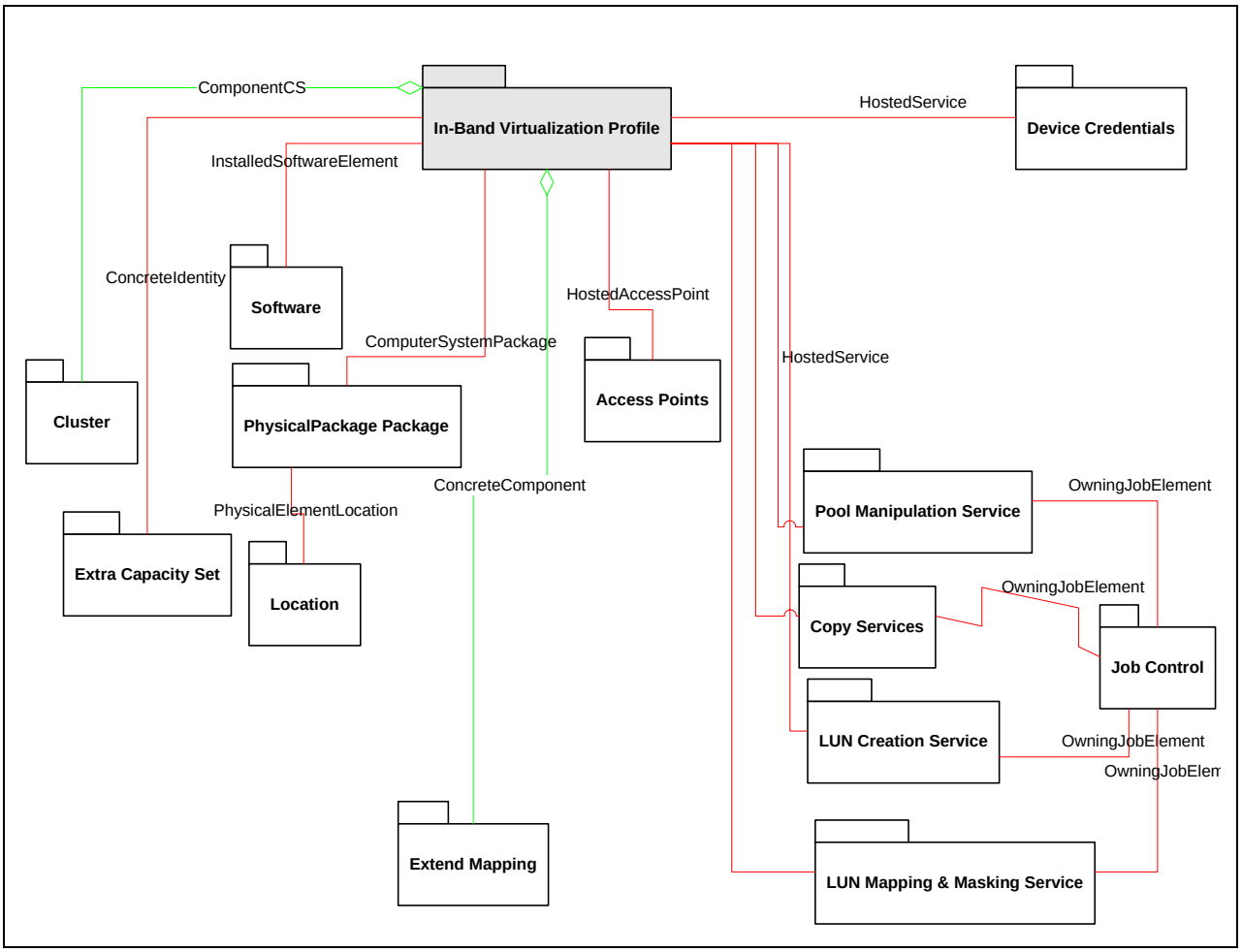


Figure 60 - In Band Virtualization Overview Diagram

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.3.6.2.5.2 In Band Virtualization System

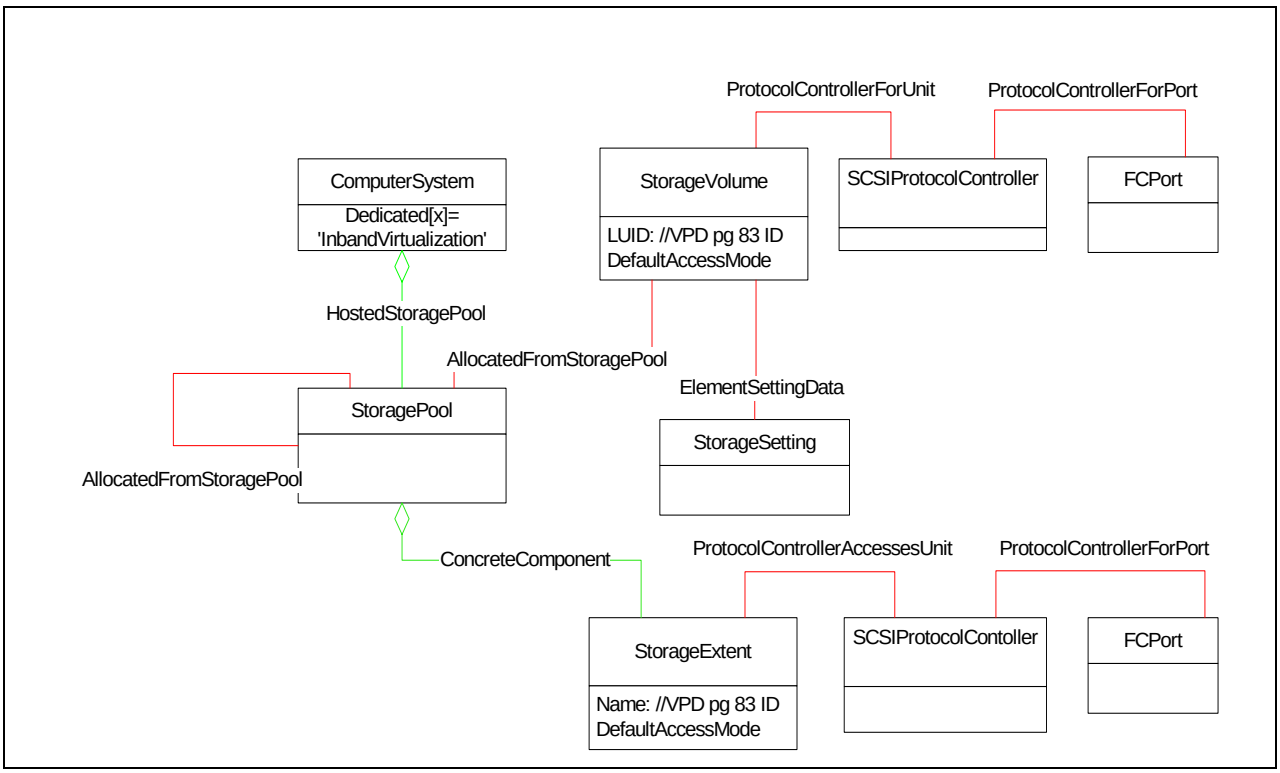


Figure 61 - In Band Virtualization System Instance

The Virtualization system is modelled using the ComputerSystem class with the “Dedicated” properties set to ‘BlockServer’ and “InBandVirtualization”. The model allows the system to be a cluster or contain redundant components, but the components act as a single system. The ComputerSystem class and common Cluster Subprofile model this.

The StoragePool classes in the center of the diagram represents the mapping from array storage to Volumes for host access. The pool is hosted on the ComputerSystem and services to control it are host on the same controller. The StorageExtent at the bottom of the screen represents the storage from external arrays used by the mapping. These StorageExtents are connected to the pool using the ConcreteComponent association.

StorageVolumes at the upper right are the volumes created from the StoragePool and are accessible from hosts. The associations to the SCSIProtocolController and to the FCPort indicate ports the volume is mapped to. The StorageVolumes are described by the StorageSetting class connected by the ElementSettingData association.

7.3.6.2.5.2.1 Controller software

Information on the installed controller software is represented by the optional Software subprofile. This is linked to the controller using an InstalledSoftwareElement association.

7.3.6.2.5.2.2 Device management access

Most devices now have a web GUI to allow device specific configuration. This is modelled using the common subprofile “Access Point”.

7.3.6.2.5.2.3 Physical modelling

The physical aspects of the metadata controller are represented by the Common Package “Physical Package” and the optional subprofile “Location”. See these common sections for more details.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.3.6.2.5.2.4 Services

The system hosts services used to control the configuration of the system's resources . These services are optional and modelled by "LUN Creation", "Copy Services" and "Job Control" subprofiles.

7.3.6.2.6 Durable names and correlatable IDs of the Profile

7.3.6.2.6.1 Durable names exported

For StorageVolume, the durable names used are the names of the volumes. The format of this property is available in NameFormat. The valid formats are described in 7.3.6.1.6.2, "Find the Durable Name for Volumes".

For Fibre Channel port, the durable name is the Port WWN. It is found in FCPort.PermanentAddress.

For the system itself (the computer system), the Name property contains a durable name. The format of this name is defined by the NameFormat property.

7.3.6.2.6.2 Correlatable IDs used

None.

7.3.6.2.7 Methods

The methods needed by this model are part of the common subprofiles for the services and are described there.

7.3.6.2.8 Client considerations

The description and figures describe an implementation with Fibre Channel connectivity. In order to support other transports in a future revision of the specification, clients should use LogicalPort rather than FCPort in CIM Operations. LogicalPort.PermanentAddress continues to serve as a correlatable ID for ports, but it is not necessarily an FC WWN.

7.3.6.2.9 Recipes

There are no recipes defined for this profile.

7.3.6.2.10 Instrumentation requirements

None.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.6.2.11 Required CIM Elements

Table 238 - Required CIM Elements

Profile Classes & Associations	Notes
AllocatedFromStoragePool (7.3.6.2.12.1)	
ConcreteComponent (7.3.6.2.12.3)	
ComputerSystem (7.3.6.2.12.2)	
ElementCapabilities (7.3.6.2.12.4)	
ElementSettingData (7.3.6.2.12.5)	
FCPort (7.3.6.2.12.6)	
HostedStoragePool (7.3.6.2.12.7)	
ProtocolControllerAccessesUnit (7.3.6.2.12.8)	
ProtocolControllerForPort (7.3.6.2.12.9)	OPTIONAL; mandatory for FCPorts.
ProtocolControllerForUnit (7.3.6.2.12.10)	
SCSIProtocolController (7.3.6.2.12.11)	
StorageCapabilities (7.3.6.2.12.12)	
StoragePool (7.3.6.2.12.14)	
StoragePool (7.3.6.2.12.14)	
StorageSetting (7.3.6.2.12.15)	
StorageVolume (7.3.6.2.12.16)	
SystemDevice (7.3.6.2.12.17)	(port)
SystemDevice (7.3.6.2.12.17)	(volume)
StorageExtent (7.3.6.2.12.13)	
Packages	
Physical Package “package” (7.3.2.2).	
Associated Indications	
Creation/Deletion of a Storage Pool	SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_StoragePool SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_StoragePool.
Creation/Deletion of a Storage Volume	SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_StorageVolume SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_StorageVolume.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table 238 - Required CIM Elements (Continued)

Profile Classes & Associations	Notes
Creation/Deletion of an FCPort	SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_FCPort SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_FCPort.
Creation/Deletion of a Virtualizer	SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_ComputerSystem SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_ComputerSystem.
Change in the status of a Storage Volume	SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_StorageVolume AND SourceInstance.Operationalstatus <> PreviousInstance.Operationalstatus.
Change in the status of an FC Port	SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_FCPort AND SourceInstance.Operationalstatus <> PreviousInstance.Operationalstatus.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.6.2.12 Required properties for CIM elements

7.3.6.2.12.1 AllocatedFromStoragePool

Table 239 - Required Properties for AllocatedFromStoragePool

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Antecedent	ref	override	The reference to the StoragePool.
Dependent	ref	override	The reference to the logical element that is the subsidiary element.
SpaceConsumed	uint64		Space Consumed from this Pool (in megabytes).

7.3.6.2.12.2 ComputerSystem

Table 240 - Required properties for ComputerSystem

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
CreationClassName	string	key	
ElementName	string		User Friendly name.
OperationalStatus	uint16		Status of array.
Name	string	key	The identifier for the Array (e.g., IP address or FC world-wide name).
NameFormat	string		The format of the Name property.
Dedicated[ ]	int16	“blockserver”, “metadatacontroller”	The use of this ComputerSystem.
PrimaryOwnerContact	string		Optional.
PrimaryOwnerName	string		Optional.

7.3.6.2.12.3 ConcreteComponent

Table 241 - Required properties for ConcreteComponent

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
GroupComponent	ref	key, override	
PartComponent	ref	key, override	

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.6.2.12.4 ElementCapabilities

Table 242 - Required properties for ElementCapabilities

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
ManagedElement	ref	key, override	
Capabilities	ref	key, override	

7.3.6.2.12.5 ElementSettingData

Table 243 - Required properties for ElementSettingData

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
ManagedElement	ref	key	The ManagedElement.
SettingData	ref	key	The Setting Data object associated with the ManagedElement.
IsDefault	uint16		An enumerated integer indicating that the referenced setting is a default setting for the element, or that this information is unknown."),||ValueMap {"0", "1", "2"}, ||Values {"Unknown", "Is Default", "Is Not Default"}.
IsCurrent	uint16		An enumerated integer indicating that the referenced setting is currently being used in the operation of the element, or that this information is unknown."),||ValueMap {"0", "1", "2"}, ||Values {"Unknown", "Is Current", "Is Not Current"}.

7.3.6.2.12.6 FCPort

Table 244 - Required properties for FCPort

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
SystemCreationClassName	string	key	
SystemName	string	key	
CreationClassName	string	key	
ElementName	**string**		User friendly name/caption for port.
OperationalStatus[ ]	uint16		Status of device.
DeviceID	string	key	Opaque.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table 244 - Required properties for FCPort (Continued)

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
PortType	uint16		FC-GS Port.Type The specific mode currently enabled for the Port. The values: \"N\" = Node Port, \"NL\" = Node Port supporting FC arbitrated loop, \"E\" = Expansion Port connecting fabric elements (for example, FC switches), \"F\" = Fabric (element) Port, \"FL\" = Fabric (element) Port supporting FC arbitrated loop and \"B\" = Bridge Port. PortTypes are defined in ISO/IEC 14165-414. When set to 1 (\"Other\"), the related property OtherPortType contains a string description of the port's type. PortType is defined to force consistent naming of the 'type' property in subclasses and to guarantee unique enum values for all instances of NetworkPort. A range of values, DMTF_Reserved, has been defined that allows subclasses to override and define their specific port types. Vendor Reserved = 16 000..65 535 can be used if the PortType is not one already defined in the above enumerations and a vendor subclass is defined specifying the appropriate “value” and “valuemap”. This property is OPTIONAL.
PermanentAddress	string		The WWN of the port.
NetworkAddresses[ ]	string		The Fibre Channel address of the port. This property is OPTIONAL.
Speed	uint64		Speed of zero represents a link not established. 1 Gbit is 1 062 500 000 bit/s 2 Gbit is 2 125 000 000 bit/s 4 Gbit is 4 250 000 000 bit/s) 10 Gbit single channel variants are 10 518 750 000 bit/s 10 Gbit four channel variants are 12 750 000 000 bit/s This is the raw bit rate. This property is OPTIONAL.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.6.2.12.7 HostedStoragePool

Table 245 - Required properties from HostedStoragePool

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
PartComponent	ref		The storage pool.
GroupComponent	ref		The scoping system.

7.3.6.2.12.8 ProtocolControllerAccessesUnit

Table 246 - Required properties for ProtocolControllerAccessesUnit

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
NegotiatedSpeed	unit64		
NegotiatedDataWidth	unit32		
Dependent	ref	override	LogicalDevice Reference.
AccessState	unit16		
TimeOfDeviceReset	datetime		
NumberOfHardResets	unit32		
NumberOfSoftResets	unit32		
Antecedent	ref	override	SCSIProtocolController Reference.
SCSITimeouts	unit32		
SCSIRetries	unit32		
InitiatorId	unit32		
TargetId	uint32		
TargetLUN	unit64		
SCSIReservation	unit16		
SCSISignal	unit16		
MaxQueueDepth	unit32		
QueueDepthLimit	unit32		

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.6.2.12.9 ProtocolControllerForPort

Table 247 - Required Properties from ProtocolControllerForPort

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Antecedent	ref		The SCSIProtocolController for this port.
Dependent	ref		The port.
AccessPriority	unit16		The priority of access through this port for this ProtocolController. This property is OPTIONAL.

7.3.6.2.12.10 ProtocolControllerForUnit

Table 248 - Required Properties from ProtocolControllerForUnit

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Antecedent	ref		The protocol controller
Dependent	ref		The exposed logical unit.
DeviceNumber	unit16		The Logical Unit number for this Volume through this controller. Formatted as unseparated uppercase hexadecimal digits, with no leading “0x”.

7.3.6.2.12.11 SCSIProtocolController

Table 249 - Required properties for SCSIProtocolController

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
ElementName	**string**		User friendly name/caption for port. This property is OPTIONAL.
OperationalStatus[ ]	uint16		Status of device. This property is OPTIONAL.
DeviceID	string	key	Opaque
MaxUnitsControlled	uint32		Maximum number of units controlled by this controller This property is OPTIONAL.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.6.2.12.12 StorageCapabilities

Table 250 - Required properties from StorageCapabilities

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
InstanceID	unit16	key	
ElementName	string		User friendly name/caption.
ElementType	uint16		Type of element this capability applies to.
NoSinglePointOfFailure	boolean		
NoSinglePointOfFailureDefault	boolean		
DataRedundancyMin	uint16		
DataRedundancyMax	uint16		
DataRedundancyDefault	uint16		
DeltaReservationMin	uint16		
DeltaReservationMax	uint16		
DeltaReservationDefault	uint16		
PackageRedundancyMin	uint16		
PackageRedundancyMax	uint16		
PackageRedundancyDefault	unit16		

7.3.6.2.12.13 StorageExtent

Table 251 - Required properties for StorageExtent

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Name	string	required	
SystemCreationClassName	string	maxlen(256), key	The scoping System's CreationClassName.
SystemName	string	maxlen(256), key	The scoping System's Name.
CreationClassName	string	maxlen(256), key	The name of the concrete subclass.
DeviceID	string	maxlen(64), key	unique identifying information.
BlockSize	uint64		
NumberOfBlocks	uint64		
ConsumableBlocks	uint64		
OperationalStatus	unit16[ ]		
ExtentStatus	uint16[ ]		

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.3.6.2.12.14 StoragePool

Table 252 - Required Properties for StoragePool

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
ElementName	string		User Friendly name.
InstanceID	string	key	Opaque identifier.
PoolID	string	required, maxlen(256)	A unique name in the context of the System that identifies this pool.
TotalManagedSpace	uint64		
RemianingManagedSpace	unit64		
Primordial	boolean		defaults to false, true for the primordial pools.

7.3.6.2.12.15 StorageSetting

Table 253 - Required properties from StorageSetting

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
InstanceID	uint16	key	
ElementName	string	required	User friendly name/caption.
DataRedundancyMin	uint16		
DataRedundancyMax	uint16		
DataRedundancyGoal	uint16		
DeltaReservationMin	uint16		
DeltaReservationMax	uint16		
DeltaReservationGoal	uint16		
PackageRedundancyMin	uint16		
PackageRedundancyMax	uint16		
PackageRedundancyGoal	unit16		

7.3.6.2.12.16 StorageVolume

Table 254 - Required properties for StorageVolume

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
SystemCreationClassName	string	key	
SystemName	string	key	

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table 254 - Required properties for StorageVolume (Continued)

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
CreationClassName	string	key	
InstanceID	uint16	key	
ElementName	string	required	User Friendly name.
Name	string		VPD Page 83 ID.
NameFormat	uint16		Format of Name property.
ExtentStatus	uint16[ ]		Status of volume (Rebuild,spare in use, etc.)
OperationalStatus	unit[ ]		Current general status of volume.
DeviceID	string		Opaque ID
BlockSize	uint64		Block size of Volume.
NumberOfBlocks	uint64		NUmber of Blocks (not size of volume is BlockSize* NumberOFBlocks).
IsBasedOnUnderlyingRedundancy	boolean		
NoSinglePointOfFailure	boolean		Current value of StorageSetting.
DataRedundancy	uint16		Current value of StorageSetting.
PackageRedundancy	uint16		Current value of StorageSetting.
DeltaReservation	uint16		Current value of StorageSetting.

--`,,```,,,,````-`-,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.3.6.2.12.17 SystemDevice

Table 255 - Required properties for SystemDevice

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
GroupComponent	ref	override	System Reference.
PartComponent	ref	override	LogicalDevice Reference.

7.3.6.2.13 Optional Subprofiles

Table 256 - Optional Profiles or Subprofiles

Name	Notes
Cluster Subprofile (7.3.3.3)	
Access Points subprofile (7.3.3.2)	
Software Subprofile (7.3.3.7)	
Location subprofile (7.3.3.6)	
Pool Manipulation Capabilities and Settings Subprofile (7.3.3.10)	
Job Control Subprofile (7.3.3.9)	
Copy Services Subprofile (7.3.3.8)	
LUN Masking and Mapping (7.3.3.14)	
LUN Creation Subprofile (7.3.3.11)	
Device Credentials subprofile (7.3.3.12)	
Extent Mapping subprofile (7.3.3.5)	

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.6.3 Storage Library Profile

7.3.6.3.1 Description

The schema for a StorageLibrary provides the classes and associations necessary to represent various forms of removable media libraries. This profile is based upon the CIM 2.7 model and defines the subset of classes that supply the necessary information for robotic storage libraries.

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. Detailed descriptions of classes are from the CIM 2.8 **preliminary** schema. Only the classes unique to storage libraries are described by this Profile. Other classes that are common to multiple Profiles may be found elsewhere in this standard.

The relevant objects for a storage library should be instantiated in the name space of the provider (or agent) for a storage library resource. Whenever an instance of a class for a resource may exist in multiple name spaces a *Durable Name* is defined to aid clients in correlating the objects across name spaces. For storage libraries, durable names are defined for the following resources:

- FCPort;
- ChangerDevice;
- TapeDrive;
- StorageLibrary;
- MediaAccessDevice.

The durable names are defined in 7.3.6.3.6, “Durable names and correlatable IDs of the profile”. All other objects do not require durable names and have instances within a single name space.

7.3.6.3.2 Standard dependencies

The storage library profile is based on the following standards:

Table 257 - Storage Library Standard Dependencies

Standard	Version	Organization
CIM Specification	2.2	DMTF
CIM Operations over HTTP	1.1	DMTF
CIM Schema	2.8 Preliminary	DMTF

7.3.6.3.3 Profile dependencies

The storage library profile requires the Server Profile (7.3.7).

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.6.3.4 CIM Server requirements

7.3.6.3.4.1 Functional Profiles

Table 258 - Required Functional Profiles

Profile Required	Functional Group	Dependency
YES	Basic Read	None.
NO	Basic Write	Basic Read.
NO	Instance Manipulation	Basic Write.
NO	Schema Manipulation	Instance Manipulation.
YES	Association Traversal	Basic Read.
NO	Query Execution	Basic Read.
NO	Qualifier Declaration	Schema Manipulation.
YES	Indication	None.

7.3.6.3.4.2 Extrinsic methods

The CIM Server shall support extrinsic methods for the Server profile.

7.3.6.3.4.3 Discovery

The CIM Server shall support SLP discovery as defined in the CIM Operations over HTTP specification.

7.3.6.3.5 Instance diagrams

7.3.6.3.5.1 Overview

The following instance diagrams represent five related views of the top-level storage library profile:

a) System Level;

b) MediaAccessDevice and its physical and logical relationships;

c) ChangerDevice and its connections to SoftwareIdentity, ProtocolController, and StorageMediaLocation;

d) StorageMediaLocation and its relationship to PhysicalMedia and other physical classes;

e) StorageMediaLocation and its required Realizes relationships.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.6.3.5.2 System Level View

Figure 62 shows the required top-level components for a StorageLibrary system. Note that *all* LogicalDevice subclasses shall be associated with StorageLibrary via System Device.

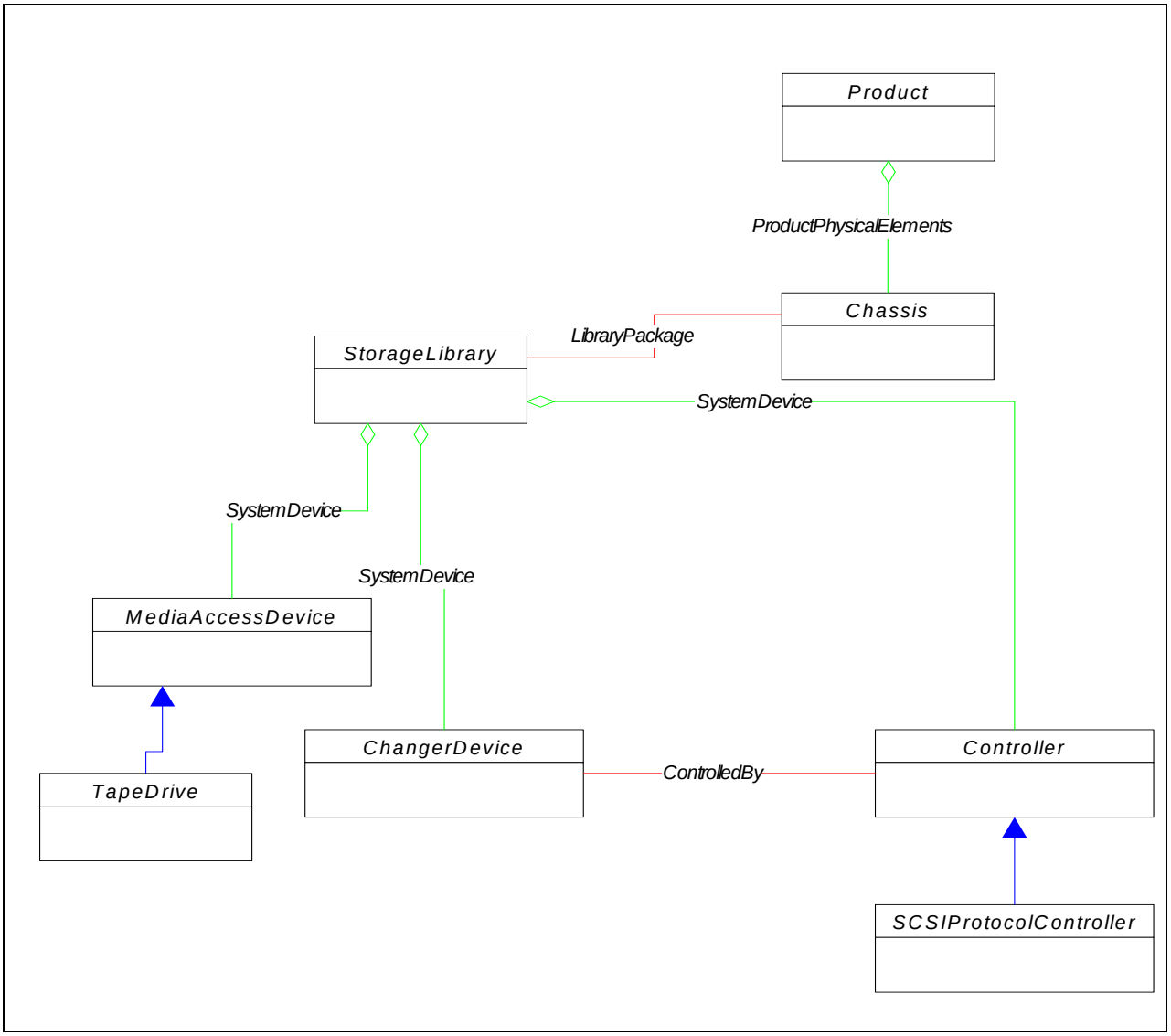


Figure 62 - StorageLibrary-centric Instance Diagram

7.3.6.3.5.3 MediaAccessDevice-centric View

Figure 63 shows the required classes related to MediaAccessDevice. Though not shown in this figure, recall that both MediaAccessDevice and ProtocolController are connected to a StorageLibrary instance through the SystemDevice association. Note that in some libraries, notably small autoloaders, external hosts access a library's ChangerDevice through the ProtocolController of a MediaAccessDevice. For such libraries, an additional ProtocolControllerForUnit association should be instantiated between the MediaAccessDevice's ProtocolController and the affected ChangerDevice. ProtocolControllerForUnit is a many-to-many

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

association, so a single ProtocolController can be connected to multiple LogicalDevices if this accurately represents a library's configuration.

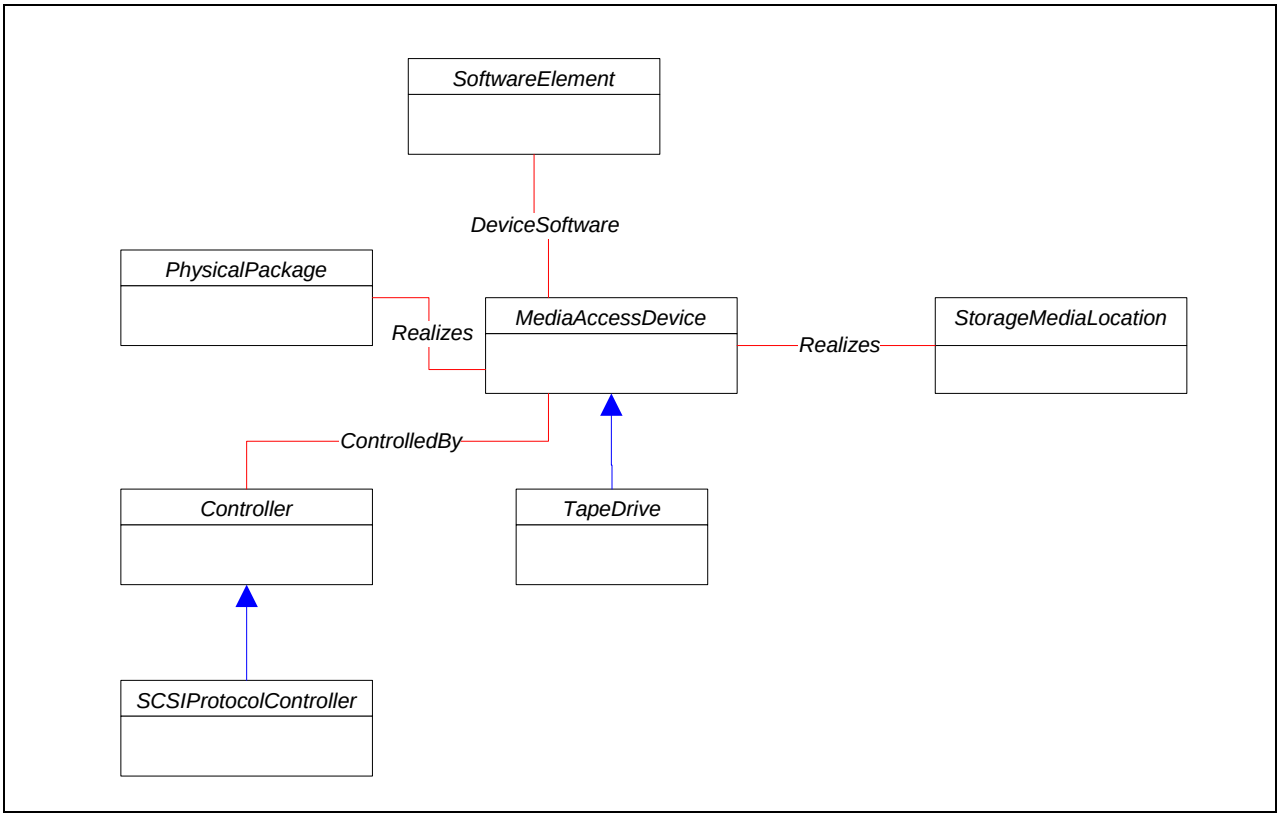


Figure 63 - MediaAccessDevice-centric Instance Diagram

7.3.6.3.5.4 ChangerDevice-centric View

Figure 64 shows the required classes related to ChangerDevice.

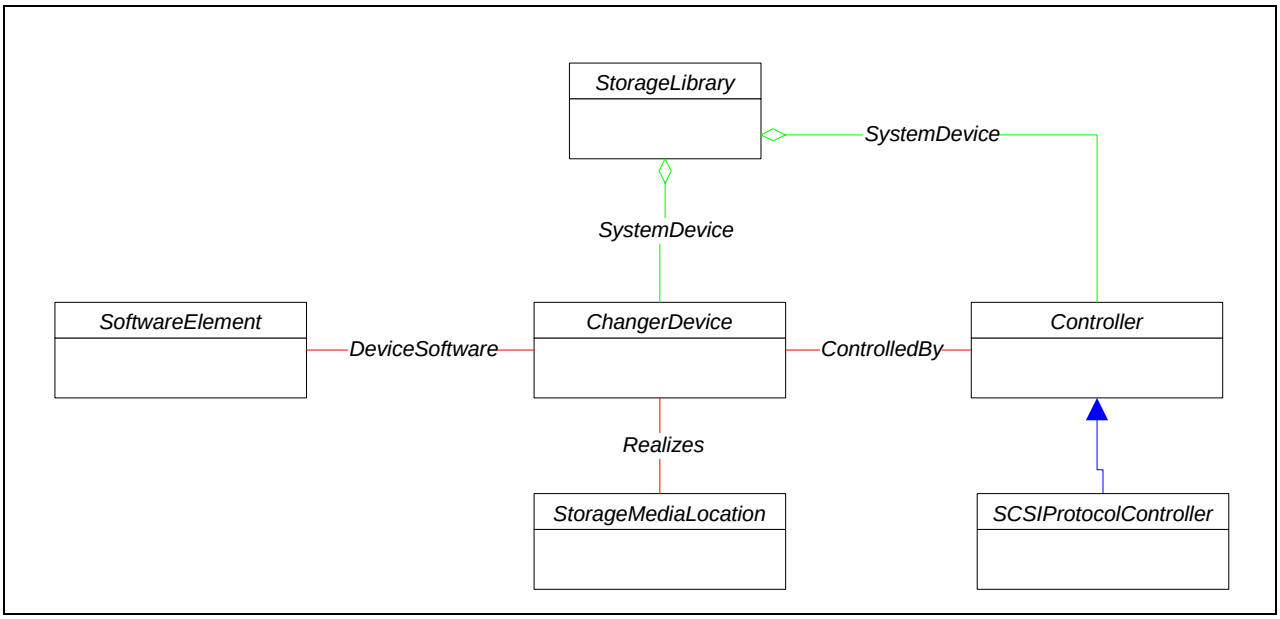


Figure 64 - ChangerDevice-centric Instance Diagram

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.6.3.5.5 Physical view

Figure 65 shows important physical components of a storage library and how they relate. With regard to StorageMediaLocation and Magazine, one of two implementation alternatives shall be selected.

a) Instantiate multiple Magazines associated to Chassis via Container, then instantiate StorageMediaLocations that are contained (again via Container) within each Magazine.

b) Instantiate multiple StorageMediaLocations directly associated to Chassis via Container, without the use of Magazines. Other optional classes, such as Panel, can also be used to group StorageMediaLocations, but this is not required.

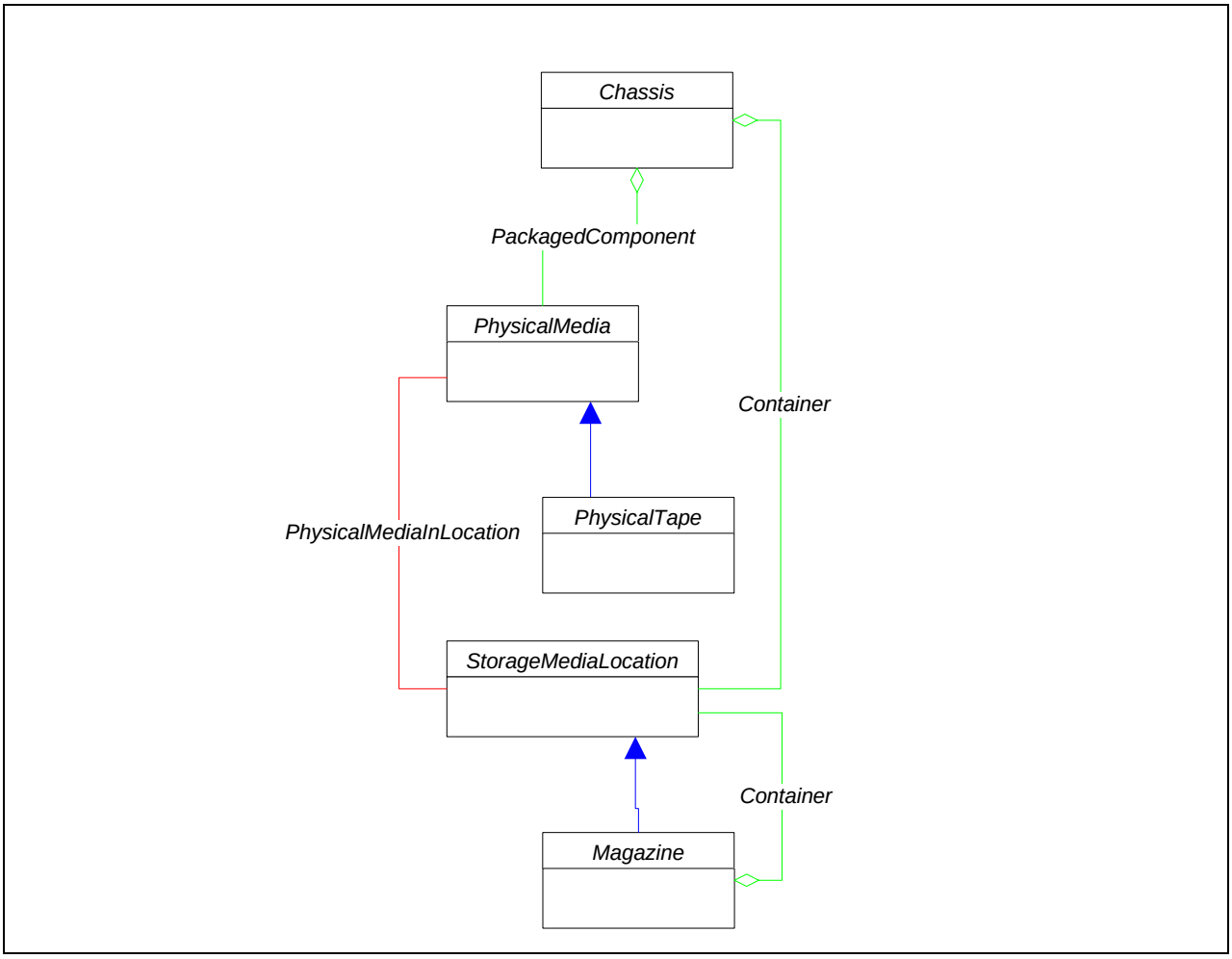


Figure 65 - Physical View Instance Diagram

7.3.6.3.5.6 StorageMediaLocation Instance Diagram

Figure 66 shows relationships between various LogicalDevices (i.e., MediaAccessDevices, LimitedAccessPort and ChangerDevice) and StorageMediaLocation. For each LogicalDevice that can hold media, at least one StorageMediaLocation shall be associated via Realizes. The figure also shows how

--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

PhysicalMedia is conceptually placed “inside” a LogicalDevice by associating PhysicalMedia with a StorageMediaLocation that Realizes a LogicalDevice.

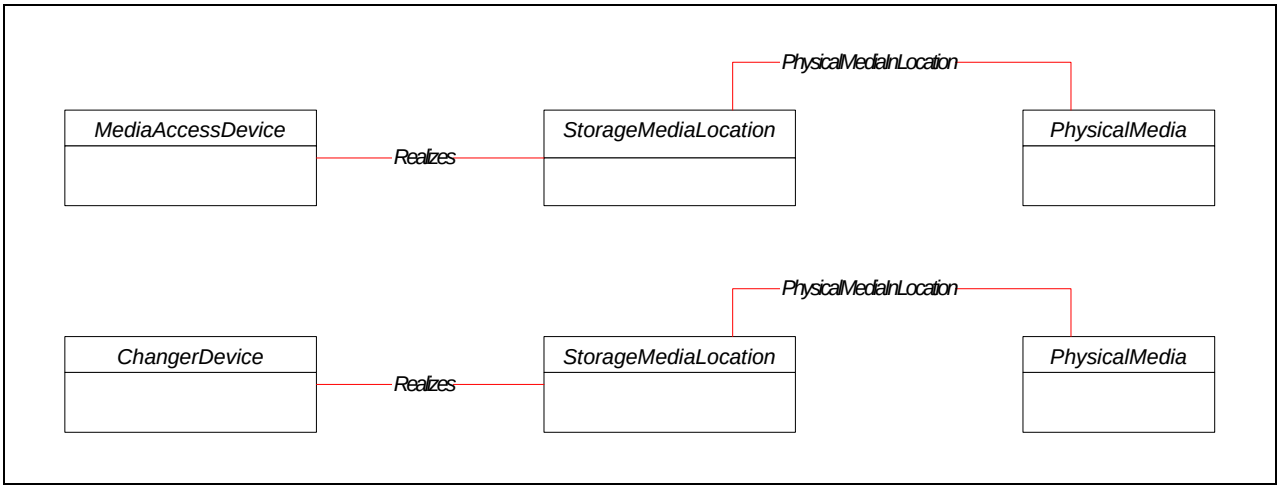


Figure 66 - StorageMediaLocation Instance Diagram

7.3.6.3.6 Durable names and correlatable IDs of the profile

7.3.6.3.6.1 Durable names exported

No durable names are exported by this profile.

7.3.6.3.6.2 Correlatable IDs used

Different implementations use different approaches to uniquely identify the SCSI units pertinent to Storage Media Libraries (i.e., Changer Devices and Media Access Devices). The agent should utilize the same Durable Name techniques described for volumes in 7.3.6.1.6, “Durable Names and Correlatable IDs of the Profile”. The chosen name is stored in the Name attribute of the logical device with the corresponding setting for the NameFormat attribute. Allowable name formats and device pairings for the storage library profile are:

- FCPort: FCPort.PermanentAddress = Fibre Channel Port World-Wide Name. NameFormat should be set to “WWN”;
- ChangerDevice.DeviceID = Vendor+Product+Serial Number+(optional instance number). Vendor, Model and Serial number should be taken from the ChangerDevice’s associated StorageLibrary, Product, and/or Chassis. An option instance number may be added to uniquely denote more than one ChangerDevice “inside” a StorageLibrary;
- MediaAccessDevice (or TapeDrive).DeviceID = Vendor+Product+Serial number for the MediaAccessDevice;
- StorageLibrary.Name = Vendor+Product+Serial number for the StorageLibrary and/or its associated Product and Chassis. NameFormat should be set to “Vendor+Product+Serial”.

Please refer to Table 3 for additional information.

7.3.6.3.7 Methods

No methods have been defined for this profile.

7.3.6.3.8 Client considerations

See 7.3.6.3.9, “Recipes”.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.6.3.9 Recipes

7.3.6.3.9.1 Overview

While no pseudo-code-based recipes have been written for this profile, this subclause provides some helpful information for writing management applications and suggests techniques for addressing common use cases.

7.3.6.3.9.2 Discover a Storage Media Library

Discovery of Storage Media Libraries is achieved by looking up instances of StorageLibrary. StorageLibrary is subclassed from System and has a corresponding Name and NameFormat property as described above under 7.3.6.3.6, “Durable Names and Correlatable IDs of the Profile”. Specifically, NameFormat shall be set to “VendorModelSerial” and the Name shall be of the form Vendor + Product + Serial.

7.3.6.3.9.3 Determine Library Physical Media capacity

The physical media capacity of a library is the number of physical media objects that may be stored in the currently installed configuration of a Storage Media Library. This capacity may be determined by enumerating the StorageMediaLocation instances that are associated with each of the library’s Chassis objects.

In implementations that choose to include the Capacity subprofile, minimum and maximum slot capacities for a Storage Library are modelled in the ConfigurationCapacity described in B.3.5, “Library Capacity Subprofile”. Since this use case relies on an optional part of the profile, it may not be supported by each agent implementation.

7.3.6.3.9.4 Determine Physical Media Inventory

To determine the physical media inventory of a StorageLibrary, clients should discover the Chassis instance associated with a particular StorageLibrary (via the LibraryPackage association) and enumerate the PhyscialMedia instances associated with the Chassis through the PackagedComponent association.

7.3.6.3.9.5 Discover Tape Library Control Type

The control mechanism to a library is either:

- SCSI Media Changer Commands directed to the library’s changer device;
- Library control commands directed to a Library Control service.

If a library does not have a ProtocolController instance associated via ProtocolControllerForUnit to the ChangerDevice then the client should conclude that an alternate mechanism for controlling the library is required. This mechanism is allowed to vary but will, ideally, be represented by an instance of Service.

7.3.6.3.9.6 Determine Library Drive capacity

The current drive capacity of a library may be determined by enumerating the MediaAccessDevice instances through the SystemDevice association of the library.

When the optional Capacity subprofile is implemented, the number of drives discovered should be within the range indicated by the minimum and maximum capacity attribute found on the library Chassis’ ElementCapacity **association with** ConfigurationCapacity for tape drives. This bounds check is not available if the Capacity subprofile is not implemented.

7.3.6.3.9.7 Determine Drive Data Path technology

Clients can discover the data path protocol of each drive within a storage library by enumerating MediaAccessDevice instances, then following the ProtocolControllerForUnit association linking a MediaAccessDevice with a ProtocolController. Properties within Contoller can then be queried for more information. If the MediaAccessDevice has a fibre channel interface, an FCPort instance is linked to its ProtocolController by a ProtocolControllerForPort association. See B.3.3, “Fibre Channel Connection Subprofile” for more information on fibre channel connectivity.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.6.3.9.8 Find asset information

Information about the entire storage library is modelled in the Chassis instances associated with the StorageLibrary. Chassis properties include Manufacturer, Model, Version and Tag. Tag is an arbitrary identifying string.

To identify asset information for the logical devices, a client should access the corresponding logical device through the StorageLibrary object's SystemDevice association. For each logical device instance the client may then check for asset information from the PhysicalElement associated through a Realizes association. Product information may also be available through the corresponding ProductPhysicalElement/ ProductPhysicalComponent aggregation.

7.3.6.3.9.9 Discovery of Mailslots, Import/Export Elements or LimitedAccessPorts in a Storage Library

Clients may determine the number of LimitedAccessPorts in a library by enumerating the LimitedAccessPorts connected to a StorageLibrary instance via the SystemDevice association.

Note that some smaller libraries do not have the type of import/export element modelled by LimitedAccessPort. As a result, LimitedAccessPort elements are included in an (optional) subprofile. See 7.3.6.3.14, "Limited Access Port Elements Subprofile".

7.3.6.3.10 Instrumentation requirements

7.3.6.3.10.1 Indications

Agents are encouraged to support CIM indications.

7.3.6.3.10.2 Storage inventory

To be useful to client management applications, the agent for a Storage Library resource needs to accurately represent the required elements of a Storage Media Library and their proper state. In order to provide consistent Storage Inventory it is important that PackagedComponent association instances between a Storage Media Library's Chassis and PhysicalMedia instances be maintained.

Entry and exit of PhysicalTape instances from the Storage Media Library shall result in updating this set of associations. Details on this procedure vary from library to library, but, at a minimum, require an update each time a library is powered up. Other considerations involve updates whenever a LimitedAccessPort or InterLibraryPort changes state.

7.3.6.3.10.3 Hosted services

It is not uncommon for libraries to include the following services:

- Web Server – typically supports administration and configuration of the library.
- SNMP Agent – for resource monitoring and management within legacy System Management Frameworks or even to support the SMI-S proxy agent.
- NDMP Services – NDMP may be present within the library to support the NDMP Backup Process.

As an additional out-of-band management service, client management applications would be well served if they can locate these services via the agent's implementation of a corresponding instance of Service.

7.3.6.3.10.4 Media Changer Control software

The ability to determine the type of control software required by a library is an important use case for clients. For this reason, it is imperative that the agent for a library resource instantiate the appropriate subclass of ProtocolController for the ChangerDevice instances. Library vendors may subclass ProtocolController for specifying proprietary library controllers for media changer devices.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.6.3.10.5 Mixed Media Libraries

This profile fully supports mixed media style libraries. A mixed media library is a library that supports PhysicalMedia with varying properties (e.g., DLT or LTO tapes, as well as optical media). The StorageMediaLocation class' MediaTypesSupported property specifies the type of media accepted. Agent developers should implement the Container association from a MediaAccessDevice to a StorageMediaLocation so that there is a mechanism in place for determining media and drive compatibility.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.6.3.11 Required CIM Elements

Table 259 - Required CIM Elements

Profile Classes & Associations	Notes
ChangerDevice (7.3.6.3.12.1)	representing the robotic arm or picker.
Chassis (7.3.6.3.12.2)	representing the physical library frame.
Container (7.3.6.3.12.3)	links Chassis and StorageMediaLocation.
ProtocolControllerForUnit (7.3.6.3.12.4)	links ChangerDevice and ProtocolController.
ProtocolControllerForUnit (7.3.6.3.12.4)	links MediaAccessDevice and ProtocolController.
SCSIProtocolController (7.3.6.3.12.5)	representing a SCSI ProtocolController for MediaAccessDevices and ChangerDevices.
DeviceSoftware (7.3.6.3.12.6)	links ChangerDevice and SoftwareIdentity.
DeviceSoftware (7.3.6.3.12.6)	links MediaAccessDevice and SoftwareIdentity.
LibraryPackage (7.3.6.3.12.7)	links Chassis and StorageLibrary.
MediaAccessDevice (7.3.6.3.12.8)	representing a tape, optical or other drive.
PackagedComponent (7.3.6.3.12.9)	links Chassis and PhysicalMedia (or PhysicalTape).
PhysicalMedia. (7.3.6.3.12.10)	representing a tape cartridge, optical platter or other media.
PhysicalMediaInLocation (7.3.6.3.12.11)	links PhysicalMedia and StorageMediaLocation.
ProductPhysicalComponent (7.3.6.3.12.12)	links Product and Chassis.
Realizes (7.3.6.3.12.13)	links MediaAccessDevice and PhysicalPackage.
Realizes (7.3.6.3.12.13)	links MediaAccessDevice and StorageMediaLocation.
Realizes (7.3.6.3.12.13)	StorageMediaLocation and ChangerDevice.
Realizes (7.3.6.3.12.13)	StorageMediaLocation and LimitedAccessPort.
SoftwareIdentity (7.3.6.3.12.14)	representing the Changer.
SoftwareIdentity (7.3.6.3.12.14)	representing the TapeDrive or MeidaAccessDevice.
StorageLibrary (7.3.6.3.12.15)	representing the logical library itself.
StorageMediaLocation (7.3.6.3.12.16)	representing a physical location that holds media, such as a simple slot or Magazine or a location within a ChangerDevice.
SystemDevice (7.3.6.3.12.17)	links ChangerDevice and StorageLibrary.
SystemDevice (7.3.6.3.12.17)	links LimitedAccessPort and StorageLibrary.
SystemDevice (7.3.6.3.12.17)	links StorageLibrary and ProtocolController.
SystemDevice (7.3.6.3.12.17)	links StorageLibrary and MediaAccessDevice.
Packages	

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table 259 - Required CIM Elements (Continued)

Profile Classes & Associations	Notes
Physical Package Package (7.3.2.2)	For system, changer, tape drive and other media access devices.
Associated Indications	
Creation/Deletion of a StorageLibrary	SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_StorageLibrary SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_StorageLibrary.
Creation/Deletion of a PhysicalMedia	SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA PhysicalMedia SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_PhysicalMedia.
Creation/Deletion of a TapeDrive	SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_TapeDrive SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_TapeDrive.
Creation/Deletion of a ChangerDevice	SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_ChangerDevice CIM_SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_ChangerDevice.
Creation/Deletion of an FCPort	SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_FCPort SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_FCPort.
Change in operational status of a StorageLibrary	SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_StorageLibrary AND PreviousInstance.OperationalStatus <> SourceInstance.OperationalStatus.
Change in operational status of a PhysicalMedia	SELECT * FROM CIM_InstModification WHERE SourceInstance ISA *CIM*_PhysicalMedia AND PreviousInstance.OperationalStatus <> SourceInstance.OperationalStatus.
Change in operational status of a TapeDrive	SELECT * FROM CIM_InstModification WHERE SourceInstance ISA *CIM*_TapeDrive AND PreviousInstance.OperationalStatus <> SourceInstance.OperationalStatus.
Change in operational status of a ChangerDevice	SELECT * FROM CIM_InstModification WHERE SourceInstance ISA *CIM*_ChangerDevice AND PreviousInstance.OperationalStatus <> SourceInstance.OperationalStatus.
Change in operational status of an FCPort	SELECT * FROM CIM_InstModification WHERE SourceInstance ISA *CIM*_FCPort AND PreviousInstance.OperationalStatus <> SourceInstance.OperationalStatus.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.3.6.3.12 Required Properties for CIM Elements

7.3.6.3.12.1 ChangerDevice

Table 260 - Required Properties for ChangerDevice

Property/Method	Type	Qualifier/ Parameter	Description/Notes
SystemCreationClassName	string	key	The scoping System's CreationClassName.
SystemName	string	key	The scoping System's Name.
CreationClassName	string	key	Indicates the name of the class or subclass used in the creation of an instance.
DeviceID	string	key	
MediaFlipSupported	boolean		
ElementName	string		User friendly name
OperationalStatus	uint16[ ]	valuemap	"Unknown", "Other", "OK", "Degraded", "Stressed", "Predictive Failure", "Error", "Non-Recoverable Error", "Starting", "Stopping", "Stopped", "In Service", "No Contact", "Lost Communication", "Aborted", "Dormant", "Supporting Entity in Error", "Completed".
Caption	string		
Description	string		
Availability	uint16	valuemap	Values include: Other, Unknown, Running/Full Power, Warning, In Test, Power Off, and Offline. See CIM schema for values.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.6.3.12.2 Chassis

Table 261 - Required properties for Chassis

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Tag	string	key, maxlen(256)	An arbitrary string that uniquely identifies the PhysicalElement. See PhysicalElement MOF.
CreationClassName	string	key, maxlen(256)	Indicates the name of the class or subclass used in the creation of an instance.
LockPresent	boolean		Boolean indicating whether the Frame is protected with a lock.
SecurityBreach	uint16	valuemap	“Other”, “Unknown”, “No Breach”, “Breach Attempted”, “Breach Successful”.
IsLocked	boolean		Boolean indicating that the Frame is currently locked.
ElementName	string		
Manufacturer	string	maxlen(256)	The name of the organization responsible for producing the PhysicalElement.
Model	string	maxlen(256)	The name by which the PhysicalElement is generally known.
SerialNumber	string		A manufacturer-allocated number used to identify the PhysicalElement.

7.3.6.3.12.3 Container

Table 262 - Required properties for Container

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
GroupComponent	ref	override	PhysicalPackage Reference
PartComponent	ref	override	PhysicalElement Reference

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.3.6.3.12.4 ProtocolControllerForUnit

Table 263 - Required properties for ProtocolControllerForUnit

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Antecedent	ref	key, override	The ProtocolController.
Dependent	ref	key, override	The controlled Device.
DeviceNumber	string		Address of associated Device in context of the antecedent ProtocolController. Formatted as unseparated uppercase hexadecimal digits, with no leading “0x”.

7.3.6.3.12.5 SCSIProtocolController

Table 264 - Required properties for SCSIProtocolController

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
SystemCreationClassName	string	key	
SystemName	string	key	
CreationClass	string	key	
ElementName	**string**		User friendly name/caption for port. This property is OPTIONAL.
OperationalStatus[ ]	uint16		Status of device. This property is OPTIONAL.
DeviceID	string	key	Opaque.
MaxUnitsControlled	uint32		Maximum number of units controlled by this ProtocolController. This property is OPTIONAL.

7.3.6.3.12.6 DeviceSoftware

Table 265 - Required properties for DeviceSoftware

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Antecedent	ref	key, override	The SoftwareIdentity.
Dependent	ref	key, override	The LogicalDevice that requires or uses the software.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.6.3.12.7 LibraryPackage

Table 266 - Required properties for LibraryPackage

Property/ Method	Type	Qualifier or Parameter	Notes
Antecedent	ref	override	PhysicalPackage Reference.
Dependent	ref	override	StorageLibrary Reference.

7.3.6.3.12.8 MediaAccessDevice

Table 267 - Required properties for MediaAccessDevice

Property/ Method	Type	Qualifier or Parameter	Notes
NeedsCleaning	boolean		
MountCount	uint64		
SystemCreationClassName	string	key, maxlen(256)	
SystemName	string	key, maxlen(256)	
CreationClassName	string	key, maxlen(256)	
DeviceID	uint64	key	
Availability	uint16	valuemap	Values include: Other, Unknown, Running/Full Power, Warning, In Test, Power Off and Offline. See CIM schema for values.
PowerOnHours	uint64	counter, units(“hours”)	
TotalPowerOnHours	uint64	counter, units(“hours”)	

7.3.6.3.12.9 PackagedComponent

Table 268 - Required properties for PackagedComponent

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
GroupComponent	ref	override	PhysicalPackage Reference.
PartComponent	ref	override	PhysicalComponent Reference.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.6.3.12.10 PhysicalMedia

Table 269 - Required properties for PhysicalMedia

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Tag	string	maxlen(256), key	An arbitrary string that uniquely identifies the Physical Element.
CreationClassName	string	key	The name of the concrete subclass.
Capacity	uint64	units ("bytes")	
MediaType	uint16	valuemap	See CIM schema for values.
MediaDescription	string		Additional detail related to the MediaType enumeration.
CleanerMedia	boolean		
DualSided	boolean		
PhysicalLabels	string[ ]		One or more strings on 'labels' on the PhysicalMedia.
Removable	boolean		
Replaceable	boolean		
HotSwappable	boolean		

7.3.6.3.12.11 PhysicalMediaInLocation

Table 270 - Required properties for PhysicalMediaInLocation

Property/ Method	Type	Qualifier or Parameter	Notes
Antecedent	ref	override	StorageMediaLocation Reference.
Dependent	ref	override	PhysicalMedia Reference.

7.3.6.3.12.12 ProductPhysicalComponent

Table 271 - Required properties for ProductPhysicalComponent

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Product	ref		The Product.
Component	ref		The PhysicalElement that is part of this product.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.6.3.12.13 Realizes

Table 272 - Required properties for Realizes

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Antecedent	ref	override	PhysicalElement reference.
Dependent	ref	override	LogicalDevice reference.

7.3.6.3.12.14 SoftwareIdentity

The SoftwareIdentity is used to model either software or firmware.

SoftwareIdentity is subclassed from LogicalElement.

Table 273 - Required properties for SoftwareIdentity

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
InstanceID	string	key	The name used to identify this SoftwareIdentity.
VersionString	string		Software Version should be in the form <Major>.<Minor>.<Revision> or <Major>.<Minor><letter><revision>.
Manufacturer	string		Manufacturer of this software.
BuildNumber	uint16		OPTIONAL. The internal identifier for this compilation of software, if available.
RevisionNumber	uint16		OPTIONAL. This is the numeric representation of the revision number in the VersionString.
MajorVersion	uint16		OPTIONAL. This is the numeric representation of the Major number in the VersionString.
MinorVersion	uint16		OPTIONAL. This is the numeric representation of the Minor number in the VersionString.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.6.3.12.15 StorageLibrary

Table 274 - Required Properties for StorageLibrary

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
OperationalStatus	valuemap		
CreationClassName	string	maxlen(256), key	Name of Class
Name	string	maxlen(256), key	
Automated	boolean		
PrimaryOwnerName	string		
PrimaryOwnerContact	string		
Caption	string		
Description	string		
ElementName	string		

7.3.6.3.12.16 StorageMediaLocation

Table 275 - Required Properties for StorageMediaLocation

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Tag	string	key, maxlen(256)	An arbitrary string that uniquely identifies the PhysicalElement. See PhysicalElement MOF.
CreationClassName	string	key, maxlen(256)	Indicated the name of the class or subclass.
LocationType	uint16	valuemap	Values include "Unknown", "Other", "Slot", "Magazine", "MediaAccessDevice", "InterLibrary Port", "Limited Access Port", "Door", "Shelf", "Vault".
LocationCoordinates	string		General location information about the physical location of the StorageMediaLocation.
MediaTypesSupported	uint16[ ]	valuemap	Complete list of accepted media types. See CIM schema for list of values.
MediaCapacity	uint32		The maximum number of PhysicalMedia that this StorageMediaLocation can hold.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.6.3.12.17 SystemDevice

Table 276 - Required properties for SystemDevice

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
GroupComponent	ref	override	System Reference.
PartComponent	ref	override	LogicalDevice Reference.

7.3.6.3.12.18 TapeDrive

This object inherits all of its properties for its superclass MediaAccessDevice (7.3.6.3.12.8).

7.3.6.3.13 Optional Subprofiles

Table 277 - Optional Profiles or Subprofiles

Name	Notes
Physical Package Package (7.3.2.2)	
Access Points Subprofile (7.3.3.2)	
Location Subprofile (7.3.3.6)	
Software Subprofile (7.3.3.7)	
Limited Access Port Elements Subprofile (7.3.6.3.14)	Representing an import/export element or "mail slot".

7.3.6.3.14 Limited Access Port Elements Subprofile

7.3.6.3.14.1 Description

Most libraries contain Limited Access Ports elements (also known as mailslots, cartridge access ports or import/export elements). This subprofile defines the required classes necessary to publish information about these common components.

7.3.6.3.14.2 Standard dependencies

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

7.3.6.3.14.3 Profile dependencies

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

7.3.6.3.14.4 CIM Server requirements

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.6.3.14.5 Instance diagrams

Figure 67 shows the relationship between LimitedAccessPorts and other portions of the Storage Library profile.

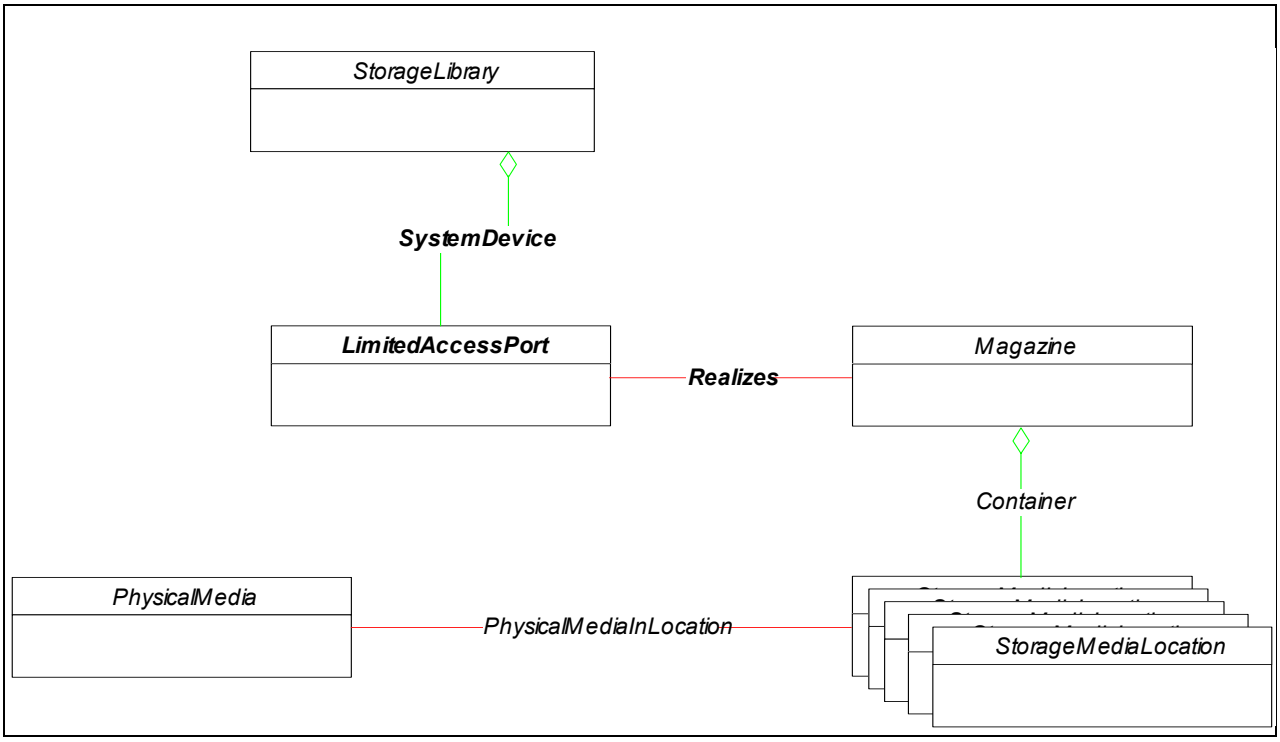


Figure 67 - LimitedAccessPort Linkages

7.3.6.3.14.6 Durable names and correlatable IDs

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

7.3.6.3.14.7 Methods

None.

7.3.6.3.14.8 Client considerations

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

7.3.6.3.14.9 Instrumentation requirements

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.6.3.14.10 Required CIM Elements

Table 278 - Required CIM Elements

Profile Classes & Associations	Notes
LimitedAccessPort (7.3.6.3.14.11.1)	
Realizes (7.3.6.3.12.13)	Connects LimitedAccessPort to StorageMediaLocations (or Magazines).
SystemDevice (7.3.6.3.12.17)	Connects LimitedAccessPort to StorageLibrary.
Packages	
None.	
Associated Indications	
Creation/Deletion of a LimitedAccessPort	SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_LimitedAccessPort SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_LimitedAccessPort.
Change in operational status of a LimitedAccessPort	SELECT * FROM CIM_InstModification WHERE SourceInstance ISA *CIM*_LimitedAccessPort AND PreviousInstance.OperationalStatus <> SourceInstance.OperationalStatus

7.3.6.3.14.11 Required properties for CIM Elements

7.3.6.3.14.11.1 LimitedAccessPort

Table 279 - Required Properties for LimitedAccessPort

Property/Method	Type	Qualifier/ Parameter	Description/Notes
SystemCreationClassName	string	key	
SystemName	string	key	
CreationClassName	string	key	
DeviceID	string	key	
Extended	boolean		When a Port is 'Extended' its StorageMediaLocations are accessible to a human operator.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table 279 - Required Properties for LimitedAccessPort (Continued)

Property/Method	Type	Qualifier/ Parameter	Description/Notes
ElementName	string		User friendly name.
Caption	string		
Description	string		

7.3.6.3.14.12 Optional subprofiles

Table 280 - Optional Profiles or Subprofiles

Name	Notes
None	

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.3.7 Server Profile

7.3.7.1 Description

A CIM Server is anything that supports the CIM-XML protocol and supports the basic read functional profile as defined by the CIM Operations over HTTP specification.

The Server Profile is the profile that all SMI-S Servers shall support for compliance.

The Object Manager part of the model defines the capabilities of a CIM Object Manager based on the communication mechanisms that it supports.

The namespace model of the Server Profile describes the namespaces managed by the Object Manager and the type information contained within the namespace. The main information provided in the namespace part of the model is the namespace itself and its association to the CIM_ObjectManager.

The RegisteredProfile part of the model is used to specify the Profiles supported by the Object Manager. It also includes the specification of subprofiles that are supported by the profile.

In this subclause there are references to the InteropNamespace and the use of the InteropNamespace for finding RegisteredProfiles and other related classes associated with the Server Profile. The InteropNamespace refers to the first namespace found in the InteropSchemaNamespace attribute of the SLP Template provided in 9.10, "'Standard WBEM' Service Type Templates".

7.3.7.2 Standard dependencies

The CIM Server Profile is based on the specifications listed in Table 281:

Table 281 - CIM Server Standard Dependencies

Standard	Version	Organization
CIM Specification	2.2	DMTF
CIM Operations over HTTP	1.1	DMTF
CIM Schema	2.8	DMTF

7.3.7.3 Profile dependencies

The Server Profile does not require any other Profiles.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.7.4 CIM Server requirements

7.3.7.4.1 Functional Profiles

Table 282 - Required Functional Profiles

Profile Required	Functional Group	Dependency
YES	Basic Read	None
NO	Basic Write	Basic Read
NO	Instance Manipulation	Basic Write
NO	Schema Manipulation	Instance Manipulation
YES	Association Traversal	Basic Read
NO	Query Execution	Basic Read
NO	Qualifier Declaration	Schema Manipulation

7.3.7.4.2 Extrinsic methods

The CIM Server shall support extrinsic methods for the Server profile.

7.3.7.4.3 Discovery

The CIM Server shall support SLP discovery as defined in the CIM Operations over HTTP specification.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.7.5 Instance Diagram

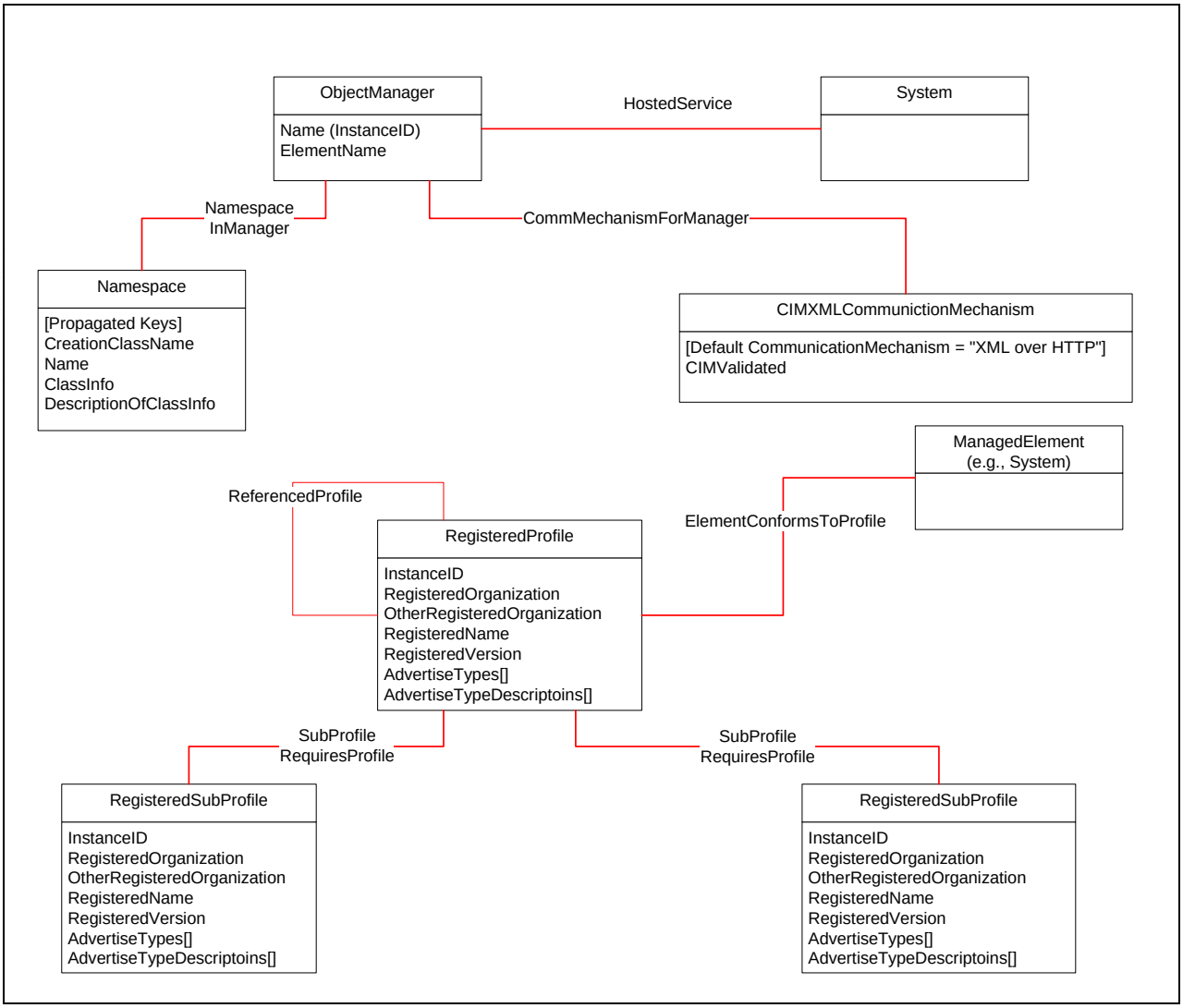


Figure 68 - Server Model

A Server is modelled as a System with a HostedService association to an ObjectManager. The ObjectManager is subclassed from Service.

This profile assumes that all namespaces supported by the Server are identified (the Namespace class) and associated to the ObjectManager via the NamespaceInManager association.

NOTE All classes of the Server Profile (as shown in Figure 68) are in the Interop Namespace, with the exception of the "ManagedElement" that is referenced from the RegisteredProfile. This makes traversing the Server Profile relatively simple. The only time a traversal may require crossing namespaces is when following the "ElementConformsToProfile" association.

The communication protocols supported by the ObjectManager will also be identified. Specifically, the CIMXMLCommunicationMechanism shall be present for standard communication support for clients. This class is associated to the ObjectManager via the CommMechanismForManager association.

The next set of classes and associations deal with Profiles supported by the ObjectManager. A Profile is modelled using the RegisteredProfile class. One instance is created for each ManagedElement that is covered by a profile and is managed by the Server. The RegisteredProfile instances can be found by enumerating RegisteredProfiles within the interop namespace. A client would find all profiles supported by the

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Server by enumerating RegisteredProfiles, enumerating RegisteredSubprofiles and subtracting the second list from the first list. This will yield the list of Profiles supported by the ObjectManager.

For each Profile instance, the subprofiles that have been implemented (for the instance) should be identified via the SubprofileRequiresProfile association. Subprofiles are modelled using the RegisteredSubProfile class. However, the RegisteredVersion property of subprofiles shall be the same as the RegisteredVersion in the parent profile.

In addition, the ElementConformsToProfile association ties the “top-level” Profile (RegisteredProfile) to the scoping ManagedElements. These ManagedElements are typically ComputerSystems or AdminDomains.

A single ManagedElement may have zero or more ElementConformsToProfile associations to RegisteredProfiles. Regardless of the number of associated RegisteredProfiles the ManagedElement represents one set of resources. So, for example, consider a ManagedElement that is a System that supports both the Array and In-Band Virtualization Appliance profiles. If one asks for the total amount of mapped capacity, the answer applies to both Array and Virtualizer and is not additive.

7.3.7.6 Durable names and other correlatable IDs

7.3.7.6.1 Durable names and other correlatable IDs exported

The Server Profile exports the following:

CIM Server ID - ObjectManager.Name - The Name property is used to uniquely identify a CIM Server. The CIM Server shall ensure that this value is globally unique. In order to ensure uniqueness, this value shall be constructed using the following 'preferred' algorithm: <OrgID>:<LocalID> .

Where <OrgID> and <LocalID> are separated by a colon ':', and where <OrgID> shall include a copyrighted, trademarked or otherwise unique name that is owned by the business entity creating/defining the name, or a registered ID that is assigned to the business entity by a recognized global authority. (This is similar to the <Schema Name>_<Class Name> structure of Schema class names.) In addition, to ensure uniqueness, <OrgID> shall not contain a colon (':'). When using this algorithm, the first colon to appear in InstanceID shall appear between <OrgID> and <LocalID>.

<LocalID> is chosen by the organizational entity to uniquely identify different underlying (real-world) elements. If the above 'preferred' algorithm is not used, the defining entity shall assure that the resultant InstanceID is not re-used across any InstanceIDs produced by this or other providers for this instance's NameSpace.

NOTE Name is semantically the same as InstanceID. In the next major version of the CIM Schema, Name is to be renamed to InstanceID and become the only key of this class.

ProfileInstance - RegisteredProfile.InstanceID - The InstanceID property is used to uniquely identify a Profile Instance. The Server shall ensure that this value is globally unique. In order to ensure uniqueness, this value shall be constructed in the following manner: <OrgID>:<LocalID> .

7.3.7.6.2 Durable names and other correlatable IDs used

None.

7.3.7.7 Methods

All Basic Read and Association Traversal methods are supported for the Server profile. However, basic write or instance manipulation methods are optional for the Server profile. The model itself is instantiated by the CIM Server and cannot be directly modified. Indirectly, the ObjectManager (RequestedStatus) can be modified via the StopService extrinsic method.

StopService()

The StopService method may be applied to the ObjectManager. The StopService method places the ObjectManager in the stopped state.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

NOTE The StartService method is not supported for Object Managers.

7.3.7.8 Client considerations

7.3.7.8.1 Using the CIM Server Model to Determine SNIA Profiles Supported

All SNIA Profiles require the implementation of the Server Profile as part of the CIM Server. This allows a client to determine which SNIA Profiles are supported by the a proxy, embedded or general purpose SMI-S Server. SMI-S clients can use SLP to search for services that support SNIA profiles. Indeed, a client may restrict its search to specific types of SNIA profiles. The client would get a response for each CIM Server service that supports a SNIA profile. From the responses, the client should use the “service-ID” to determine the unique CIM Servers it is dealing with.

For each CIM Server, the client can determine the types of entities supported by inspecting the RegisteredProfilesSupported attribute returned for the SLP entries. This identifies the types of entities (e.g., devices) supported by the CIM Server.

The Client may determine more detail on the support for the Profiles by going to the service advertised for the CIM Server and inspecting the RegisteredProfiles maintained in the server profile. This would be done by enumerating RegisteredProfiles and RegisteredSubprofiles within the interop namespace. By inspection of the actual profile instances, the client can determine the SNIA version (RegisteredVersion) of profile, associated namespaces and associated managed elements (e.g., systems).

7.3.7.8.2 Using the CIM Server Model to Determine Optional Features supported

From the RegisteredProfiles within the namespace of the ObjectManager, a client can determine the “optional features” that are supported for the profile by following the SubprofileRequiresProfile association. This returns a set of RegisteredSubProfile instances that represent Subprofiles of the specific Profile instance. The name of the subprofile is scoped by the Profile. See individual Profile descriptions in this standard for the specific list of “optional subprofiles” supported. For a given profile instance there may be zero, one or many subprofiles. The optional subprofiles documented in this specification merely list the subprofiles that may be associated with the profile (via the SubprofileRequiresProfile association).

All Subprofiles that are supported by a Profile shall be directly associated to the Profile via the SubprofileRequiresProfile association. All subprofiles (either direct or indirect via subprofiles) shall be directly attached to the Profile. For example, the Array Profile instance can support two subprofiles: LUN Creation and Job Control. Both of these subprofiles would be directly attached to the Array Profile instance, even though the Job Control subprofile is actually a subprofile of LUN Creation.

NOTE The RegisteredVersion property of subprofiles shall match the RegisteredVersion property of its parent Profile.

7.3.7.9 Recipes

7.3.7.9.1 Assumptions

For discovery recipes, the following are assumed:

a) a top-level object (class instance) exists for each Profile and

b) the client knows what the top level object is.

The top-level object for each of the SMI-S Profiles are:

- ComputerSystem: For JBOD, Array, Virtualizers, Switches, Routers and HBAs. This is the top-level ComputerSystem instance for the Profile (not the component ComputerSystem or the member ComputerSystem);
- AdminDomain: For Fabric and HostDiscoveredResources;
- StorageLibrary: For Storage Libraries;

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

- ObjectManager: For Server.

The top-level object (class instance) is associated to the RegisteredProfile instance for the Profile via the ElementConformsToProfile association.

NOTE Other ManagedElement instances may be associated to the RegisteredProfile, but the meaning and behavior of these associations are not defined by SMI-S and are optional.

7.3.7.9.2 Find Servers supporting a given Profile

```
// DESCRIPTION
// A management application wishes to find all CIM Servers on a
// particular subnet that support one or more SMI-S profiles.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
// 1.Assume CIM Servers have advertised their services (SrvReg)
// 2.Assume there may (or may not) be Directory Agents in the subnet
// 3.Assume no security on SLP discovery
// 4.#DirectoryList[] is an array of directory URLs
// 5.#ServiceList[] is an array of service agent URLs
// 6.#DirectoryEntries [] is an array of directory entry Structures.
//   The structure matches the “wbem” SLP Template (see Clause 5,
//   section 10).

// Step 1:  Set the Previous Responders List to the Null String.
#PRList = ““

// Step 2: Multicast a Service Request for a Directory Server Service.
//   This is to find Directory Agents in the subnet.
//
SrvRqst (
     #PRList,        // The Previous Responders list
     ”service:directory-agent” // Service type
     “DEFAULT”,          // The scope
     NULL,             // The predicate
     NULL)             // SLP SPI (security token)

// Step 3: Listen for Response from Directory Agent(s)
#DirectoryList[] = DAAdvert (
     BootTimestamp, // Time of last reboot of DA
     URL,      // The URL of the DA
     ScopeList,// The scopes supported by the DA
     AttrList,// The DA Attributes
     SLP SPI List,// SLP SPI (SPIs the DA can verify)
     Authentication Block)
// Iterate on Steps 2 & 3, until a response has been received or the
                           client has
// reached a UA configured CONFIG_RETRY_MAX seconds.  If no DA if found,
// proceed to step 4.  If a DA is found, proceed to step 7.

// Step 4: Set the Previous Responders List to the Null String.
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
#SAPRList = ""

// Step 5: Multicast a Service Request for Service Agent Services. This
//   is to find Service Agents in the subnet that are not advertised
//   in a Directory.

SrvRqst (
     #SAPRList,        // The Previous Responders list
     "service:service-agent" // Service type
     "DEFAULT",          // The scope
     "(Service-type=WBEM)",        // The predicate
     NULL)             // SLP SPI (security token)

// Step 6: Listen for Response from Service Agent(s)
#SAList[] = SAAdvert (
     URL,      // The URL of the SA
     ScopeList,// The scopes supported by the SA
     AttrList,// The SA Attributes
     Authentication Block)
// Iterate on Steps 5 & 6, until a response has been received or the
                           client has
// reached a UA configured CONFIG_RETRY_MAX seconds.  If no SA if found,
// Then record an error.   There are NO WBEM SAs.  Otherwise proceed to
// Step 8.

//Step 7: Unicast a Service Request to each of the DAs specifying
//   a query predicate to select CIM Servers that support SNIA profiles
//   and listen for responses.
for #j in #DirectoryList[]
{
     SrvRqst (
        #PRList,        // The Previous Responders list
        "service:wbem",    // Service type
        "DEFAULT",          // The scope
        RegisteredProfilesSupported="SNIA:*", // The predicate
        NULL)             // SLP SPI (security token)

     #ServiceList [#j] = SrvRply  (
        Count,       // count of URLs
        URL for each SA returned)
}
Go to Step 9.

//Step 8: Unicast a Service Request to each of the SAs specifying
//   a query predicate to select CIM Servers that support SNIA profiles
//   and listen for responses.
for #j in #SAList[]
{
```

--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
        SrvRqst (
            #PRList,        // The Previous Responders list
            ”service:wbem”,    // Service type
            “DEFAULT”,          // The scope
            RegisteredProfilesSupported=“SNIA:*”, // The predicate
            NULL)             // SLP SPI (security token)

        #ServiceList [#j] = SrvRply  (
            Count,       // count of URLs
            URL for each SA returned)
    }


    // Step 9: Next retrieve the attributes of each advertisement
    For #i in #ServiceList[]  // for each url in list
    {
        AttrRqst (
            #PRList,       // The Previous Responders list
            #ServiceList[#i],// a url from #ServiceList[]
            “DEFAULT”, // The scope
            NULL,  // Tag list.  NULL means return all attributes
            NULL)  // SLP SPI (security token)
        #DirectoryEntries [#i] = AttrRply (attr-list)
    }

    // Step 10: Correlate responses to the Service Request on unique
    //   “service-id” to determine unique CIM Servers. The client will get
    //   multiple responses (one for each access point) for each CIM
    //   Server. At this point, the client has a list of CIM Servers that
    //   claim to support SNIA profiles.
```

7.3.7.9.3 Enumerate Profiles Supported by a Given CIM Server

```
    // DESCRIPTION
    // A management application wishes to determine the Profiles supported
                                by
    // a particular CIM Server.
    //
    // PRE-EXISTING CONDITIONS AND ASSUMPTION
    // 1.Assume the client only wants to know the “top level” profiles
    //   supported by the CIM Server
    // 2.Assume the client has used SLP to find the CIM Servers and has a
    //   #DirectoryEntries [] structure
    // 3.This recipe describes the operations for one of the entries in
    //   the #DirectoryEntries [] structure.
    // 4.    Assume the index into #DirectoryEntries[] for the CIM Server of
    //        interest is #i.
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
// Step 1: Get the server url for the CIM Server.
#ServerName = #DirectoryEntries[#i].service-id

// Step 2: Get the Interop Namespace for the CIM Server.
#Inamespace = #DirectoryEntries[#i].InteropSchemaNamespace[1]

// Step 3:  Establish a connection to the CIM Server with
// #INameSpace. Note that the WBEM operations throughout the remainder
// of this recipe are performed with this client handle.
<Make client connection to this server using the interop namespace>

// Step 4:  Get the names of all the RegisteredProfiles in the
// Interop Namespace
#ProfileName[] = EnumerateInstances(“CIM_RegisteredProfile”,
TRUE, TRUE, FALSE, FALSE,
[“RegisteredName”])

// Step 5:  Get all the RegisteredSubprofiles in the Interop Namespace
#SubprofileName[] = EnumerateInstances(“CIM_RegisteredSubprofile”,
TRUE, TRUE, FALSE, FALSE,
[“RegisteredName”])


// Step 6:  Subtract the list RegisteredSubprofiles from the list of
// RegisteredProfiles
     #k = 0
     for #i in #ProfileName[i] {
        for #i in #SubprofileName[j] {
           if #ProfileName[#i] != #SubProfileName[#j] {
              #TempArray[#k+1]=#ProfileName[#i]
           }
        }
     }
     #ProfileName[] = #TempArray[]
```

7.3.7.9.4 Identify the ManagedElement defined by a Profile

```
// DESCRIPTION
// A management application wishes to determine the ManagedElement that
// is defined by a particular Profile.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
// 1.Assume the client has located the profile and has its object path
//   ($RegisteredProfile->)

// Step 1: Determine the ManagedElement (System) by traversing the
//    ElementConformsToProfile association from the RegisteredProfile
//    that is the top level Profile that applies to the System
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
$ManagedElement->[] = AssociatorNames (
     $RegisteredProfile->,
     “CIM_ElementConformsToProfile”,
     “CIM_System”,    // Note: substitute “CIM_AdminDomain” for Fabrics
                  // or “CIM_ComputerSystem for Arrays
                  // or “CIM_StorageLibrary for Libraries
                              // or “CIM_ObjectManager for Servers
     NULL,
     NULL)


// Step 2: The object name of more than one System may be contained
// in the array returned. Examine the contents of $ManagedElement[]
// and save the name of the System of interest as $Name.

// NOTE: “Top” level object for each profile will be returned. It MUST
                            have
// an ElementConformsToProfile association.  To accommodate other
// potential ManagedElements, then it will be necessary need to throw
                            out
// the ones that are NOT top level objects.

// NOTE: The object path for the ManagedElement MAY be in a Namespace
//   that is different than the Interop Namespace. As a result, if the
//   client wishes to actually access the ManagedElement, the client
//   may get the namespace for the element by cracking the REF to the
//   element:
#NameSpace=$Name.getNameSpace()
```

7.3.7.9.5 Determine the SNIA Version of a Profile

```
// DESCRIPTION
// A management application wishes to determine the SNIA version
// that a particular Profile supports.

//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
// 1.Assume the client only wants to know version information
//   for a SNIA profile
// 2.Assume the client has already found the profile and has the
//   $RegisteredProfile-> reference


     // Step 1: Get the Instance of the Profile name.
     $Profile = GetInstance($RegisteredProfile->)

     // Step 2: Determine the SNIA Version for the Profile selected.
     #SNIAVersion = $Profile.RegisteredVersion
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.7.9.6 Determine the subprofile capabilities of a Profile

```
// DESCRIPTION
// A management application wishes to determine the optional
                              subprofiles
// supported by a SNIA Profile.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
// 1.Assume the client has already discovered the CIM Server that
//   supports the SNIA profile
// 2.Assume the client already has a $ObjectManager-> reference for
//   the CIMOM on the WBEM Server.
// 3.Assume the client already has a $RegisteredProfile-> reference
//   for the profile in question.

// Step 1: Check the version of the supported profile. Based on the
//   RegisteredVersion property, the client should know what functions
//   are REQUIRED as part of the profile definition.
$Profile = GetInstance($RegisteredProfile->)
#ProfileVersion = $Profile.RegisteredVersion

// Step 2: For each Profile, traverse the SubProfileRequiresProfile
//   association to determine what optional subprofiles are also
//   supported. If the subprofile (e.g., CopyServices subprofile)
//   exists for a profile, this means that the copy services are
//   supported. The Copy Services also has a Version
//   (RegisteredSubProfile.RegisteredVersion). The RegisteredVersion
//   of the subprofile MUST match the RegisteredVersion of the profile.
//   The RegisteredVersion implies a set of functional capabilities
//   that are defined for that version of the subprofile.
$Subprofiles[] = Associators (
     $RegisteredProfile->,
“CIM_SubProfileRequiresProfile”,
     “CIM_RegisteredProfile”,
     NULL, NULL, false, false, NULL)

// Step 3: Verify that each Subprofile has the same version as the
//   Profile
for #i in $Subprofiles[]
{

     #SubprofileVersion = $Subprofile[#i].RegisteredVersion
     if (!compare(#SubprofileVersion, #ProfileVersion))
     {
        Error(“Subprofile version mismatch with Profile version”)
     }
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
}
```

7.3.7.9.7 Find all profiles and subprofiles on a Server

```
// DESCRIPTION
// A management application wishes to list all the SNIA profiles and
// their related subprofiles for a specific CIM Server.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
// 1.Assume the client has already discovered the CIM Servers that
//   support SNIA profiles

// Step 1:  Get the names of all the RegisteredProfiles and their names
// in the Interop Namespace
$ProfileName[] = EnumerateInstances(“CIM_RegisteredProfile”
                           true, true, false, false, {“RegisteredName”})

// Step 2:  Get all the RegisteredSubprofiles in the Interop Namespace
$SubprofileName[] = EnumerateInstances(“CIM_RegisteredSubprofile”,
                           true, true, false, false, {“RegisteredName”})


// Step 3:  Subtract the list RegisteredSubprofiles from the list of
// RegisteredProfiles
#k = 0
for #i in #ProfileName[#i] {
    for #j in $SubprofileName[#j] {
        if ($ProfileName[#i] != $SubProfileName[#j]) {
               #TempArray[#k+1]=#ProfileName[#i]
        }
     }
}
#ProfileName[] = #TempArray[]

// Step 4:  Get the ObjectName for the Profiles
for #i in #ProfileName[] {
$Profile->[#i]=$Name.getObjectPath(#ProfileName[#i])
}

// Step 5: Get the subprofiles associated to the profiles.
for #i in $ProfileName[]
{
    $Subprofile[] = Associators(
                           $ProfileName[#j].getObjectPath(),
                           “CIM_SubprofileRequiresProfile”,
                           “CIM_RegisteredSubprofile”,
                           NULL, NULL, false, false, NULL)
}
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.7.9.8 Segregate a SAN device type

```
// DESCRIPTION
// A management application wishes to manage a particular type of SAN
// device, but not other devices. So the management application needs to
// isolate the particular CIM Servers that support the type of device it
// wants to manage.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
// 1.Assume CIM Servers have advertised their services (SrvReg)
// 2.Assume there are one or more Directory Agents in the subnet
// 3.Assume no security on SLP discovery
// 4.#DirectoryList[] is an array of directory URLs
// 5.#DirectoryEntries [] is an array of directory entry Structures.
//   The structure matches the “wbem” SLP Template (see “Standard
//   WBEM Service Type Templates”).
// 6.Assume that the device is #DesiredProfile and the device is an
//   SMI-S device (a SNIA defined profile)

// Step 1:  Set the Previous Responders List to the Null String.
#PRList = ““

// Step 2: Multicast a Service Request for a Directory Server Service.
//   This is to find Directory Agents in the subnet.
//
SrvRqst (
     #PRList,        // The Previous Responders list
     ”service:directory-agent” // Service type
     “DEFAULT”,          // The scope
     NULL,             // The predicate
     NULL)             // SLP SPI (security token)

// Step 3: Listen for Response from Directory Agent(s)
#DirectoryList[] = DAAdvert (
     BootTimestamp, // Time of last reboot of DA
     URL,      // The URL of the DA
     ScopeList,// The scopes supported by the DA
     AttrList,// The DA Attributes
     SLP SPI List,// SLP SPI (SPIs the DA can verify)
     Authentication Block)
// Iterate on Steps 2 & 3, until a response has been received or the
                           client
// has reached a UA configured CONFIG_RETRY_MAX seconds.

// Step 4: Unicast a Service Request to each of the DAs specifying a
//   query predicate to select CIM Servers that support SNIA
//   #DesiredDevice profiles and listen for responses.
```

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
for #j in #DirectoryList[]
{

     SrvRqst (
        #DAPRList,        // The Previous Responders list
        “service:wbem”,    // Service type
        “DEFAULT”,          // The scope
        “RegisteredProfilesSupported=SNIA:”+#DesiredProfile+”*”,
                                          // The predicate
        NULL)             // SLP SPI (security token)
     #ServiceList [#j] = SrvRply  (
        Count,       // count of URLs
        #SAPRList[])
}

// Step 5: Next retrieve the attributes of each advertisement
For #i in #ServiceList[]  // for each url in list
{
     AttrRqst (
        #SAPRList,       // The Previous Responders list
        #ServiceList[#i ],// a url from #ServiceList[]
        “DEFAULT”, // The scope
        NULL,  // Tag list.  NULL means return all
               // attributes
        NULL)  // SLP SPI (security token)
     #DirectoryEntries [#i] = AttrRply (#attr-list)
}

// Step 7: Correlate the responses to the Service Request on unique
//   “service-id” to determine unique CIM Servers. The client will get
//   multiple responses (one for each access point) for each CIM
//   Server. At this point, the client has a list of CIM Servers that
//   claim to support SNIA #DesiredProfile profiles.
```

7.3.7.10 Instrumentation requirements

7.3.7.10.1 Use of model fields to Populate the SLP template

The data used to populate the SLP template for advertising SMI-S profiles is found in the CIM Server profile. The SLP template fields are populated as follows:

template-url-syntax: The template-url-syntax shall be the WBEM URI Mapping of the location of one service access point offered by the WBEM Server over TCP transport. This attribute shall provide sufficient addressing information so that the WBEM Server can be addressed directly using the URL. The WBEM URI Mapping is defined in the WBEM URI Mapping Specification 1.0.0 (DSP0207). Example: (template-url-syntax=https://localhost:5989)

service-hi-name: ObjectManager.ElementName

service-hi-description: ObjectManager.Description

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

service-id: ObjectManager.Name

CommunicationMechanism: ObjectManagerCommunicationMechanism.CommunicationMechanism

OtherCommunicationMechanism:
ObjectManagerCommunicationMechanism.OtherCommunicationMechanism

InteropSchemaNamespace: Namespace.Name for the InteropNamespace

ProtocolVersion: ObjectManagerCommunicationMechanism.Version

FunctionalProfilesSupported: ObjectManagerCommunicationMechanism.FunctionalProfilesSupported

FunctionalProfileDescriptions: ObjectManagerCommunicationMechanism.FunctionalProfileDescriptions

MultipleOperationsSupported: ObjectManagerCommunicationMechanism.MultipleOperationsSupported

AuthenticationMechanismsSupported:
ObjectManagerCommunicationMechanism.AuthenticationMechanismsSupported

OtherAuthenticationDescription:
ObjectManagerCommunicationMechanism.AuthenticationMechanismDescriptions

Namespace: Namespace.Name for each Namespace instance supported

Classinfo: Namespace.Classinfo for each Namespace instance

RegisteredProfilesSupported: The concatenation of:

- RegisteredProfile.RegisteredOrganization;
- RegisteredProfile.RegisteredName;
- RegisteredProfile.RegisteredName (where the second RegisteredName is the name of a subprofile that is identified for SLP advertisement).

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.7.11 Required CIM Elements

Table 283 - Profile Required Classes, Associations, Methods and Indications

Profile Class & Associations	Notes
ObjectManager	This is the Object Manager service of the CIM Server.
System	The System that is hosting the Object Manager (CIM Server).
HostedService	Connects the ObjectManager to the System that is hosting the ObjectManager.
CIMXMLCommunicationMechanism	For SMI-S, this shall be supported.
HostedAccessPoint	This associates the communication mechanisms with the hosting System.
CommMechanismForManager	This associates the ObjectManager and the communication classes it supports.
Namespace	There would be one for every namespace supported.
NamespaceInManager	This osculates the namespace to the ObjectManager.
RegisteredProfile (for Profiles)	A registered profile that is supported by the CIM Server.
RegisteredSubProfile(for Subprofiles)	For each subprofile of a profile that is supported.
SubprofileRequiresProfile	Ties profiles to their subprofiles.
ElementConformsToProfile	Ties managed elements (e.g., Device system) to the registered profile that applies.
Profile methods	**Notes**
StopService()	This method is RECOMENDED.
Profile indications	**Notes**
This would be used to indicate that the object manager has been stopped This indication is recommended.	SELECT * FROM CIM_InstModification WHERE SourceInstance ISA ObjectManager AND SourceInstance.Started <> PreviousInstance.Started.
This would tell the client when a new profile has been installed on the CIM Server. This indication is recommended.	SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA RegisteredProfile.
This would tell the client when a profile has been dropped from a CIM Server. This indication is recommended.	SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA RegisteredProfile.

7.3.7.12 Required properties for CIM Elements

7.3.7.12.1 ObjectManager Class

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

An Object Manager is a type of WBEMService that defines the capabilities of the CIM Server in which this ObjectManager class resides. Details related to communicating with the ObjectManager and the Manager's basic capabilities, are stored in instances of the associated CommunicationMechanism class available through the CommMechanismForManager association. It is assumed that Basic Read operations are supported by all ObjectManager's in order to retrieve any additional detail.

The ObjectManager class subclasses from WEBMService.

Table 284 - Required Properties for ObjectManager

ObjectManager Properties	Type	Qualifier/ Parameter	Notes
SystemCreationClassName	string	key, maxlen(256)	
SystemName	string	key, maxlen(256)	
CreationClassName	string	key	
Name	string	key, maxlen(256)	
ElementName	string		
Description	string		
OperationalStatus[ ]	uint16		
StatusDescriptions[ ]	string		This shall not be NULL if “Other” is identified in OperationalStatus.
Started	boolean		

7.3.7.12.2 System Class

System represents an entity made up of component parts (defined by the SystemComponent relationship), that operates as a 'functional whole'. Systems are top-level objects in the CIM hierarchy, requiring no scoping or weak relationships in order to exist and have context. It should be reasonable to uniquely name and manage a System at an enterprise level. For example, a ComputerSystem is a kind of System that can be uniquely named and independently managed in an enterprise. However, this is not true for the power supply (or the power supply sub-'system') within the computer.

Note that System is a subclass of EnabledLogicalElement that allows the entire abstraction to be functionally enabled/disabled - at a higher level than enabling/disabling its component parts.

Table 285 - Required Properties for System

Class Properties	Type	Qualifier/ Parameter	Notes
Description	string		
ElementName	string		
OperationalStatus[ ]	uint16		
StatusDescriptions[ ]	string		This shall not be NULL if “Other” is identified in OperationalStatus.
CreationClassName	string	key, maxlen (256)	
Name	string	key, maxlen (256)	

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table 285 - Required Properties for System (Continued)

Class Properties	Type	Qualifier/ Parameter	Notes
NameFormat	string	maxlen (64)	

7.3.7.12.3 HostedService Association

HostedService is an association between a Service and the System on which the functionality resides. The cardinality of this association is one-to-many. A System may host many Services. Services are weak with respect to their hosting System. Heuristic: A Service is hosted on the System where the LogicalDevices or SoftwareFeatures that implement the Service are located. The model does not represent Services hosted across multiple systems. This is modelled as an ApplicationSystem that acts as an aggregation point for Services that are each located on a single host.

It is subclassed from Dependency.

Table 286 - Required Properties for HostedService

Class Properties	Type	Qualifier/ Parameter	Notes
System	ref		The hosting System.
Service	ref		The Service hosted on the System.

7.3.7.12.4 CIMXMLCommunicationMechanism Class

The CIMXMLCommunicationMechanism class specializes ObjectManagerCommunicationMechanism, adding properties specific to the CIM-XML protocol (XML encoding and CIM Operations).

This class is subclassed from ObjectManagerCommunicationMechanism.

Table 287 - Required Properties for CIMXMLCommunicationMechanism

Class Properties	Type	Qualifier/ Parameter	Notes
ElementName	string		
SystemCreationClassName	string	key, maxlen(256)	
SystemName	string	key	
CreationClassName	string	key	
Name	string	key, maxlen(256)	
OperationalStatus[ ]	uint16		
StatusDescriptions[ ]	string		This shall not be NULL if “Other” is identified in OperationalStatus.
CommunicationMechanism	uint16	req	This shall be 2.
OtherCommunicationMechanismDescription	string		This shall not be NULL if “Other” is identified in CommunicationMechanism.
FunctionalProfilesSupported[ ]	uint16	req	

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table 287 - Required Properties for CIMXMLCommunicationMechanism

Class Properties	Type	Qualifier/ Parameter	Notes
FunctionalProfileDescriptions[ ]	string		This shall not be NULL if “Other” is identified in ProfilesSupported.
MultipleOperationsSupported	boolean	req	
AuthenticationMechanismsSupported[ ]	uint16	req	
AuthenticationMechanismDescriptions[ ]	string		This shall not be NULL if “Other” is identified in AuthenticationMechanismsSupported.
Version	string	req	It shall be 1.1 or greater.
CIMValidated	boolean	req	

7.3.7.12.5 HostedAccessPoint Association

HostedAccessPoint is an association between a ServiceAccessPoint and the System on which it is provided. The cardinality of this association is one-to-many and is weak with respect to the System. Each System may host many ServiceAccessPoints. Heuristic: If the implementation of the ServiceAccessPoint is modelled, it shall be implemented by a Device or SoftwareFeature that is part of the System hosting the ServiceAccessPoint.

Table 288 - Required Properties for HostedAccessPoint

Class Properties	Type	Qualifier/Parameter	Notes
System	ref		The hosting System.
ServiceAccessPoint	ref		The ServiceAccessPoints hosted by this system.

7.3.7.12.6 CommMechanismForManager association

The CommMechanismForManager is an association between an ObjectManager and an ObjectManagerCommunicationMechanism class. The latter describes a possible encoding/protocol/set of operations for accessing the referenced ObjectManager.

It is subclassed from ServiceAccessBySAP.

Table 289 - Required Properties for CommMechanismForManager

Class Properties	Type	Qualifier/ Parameter	Notes
ObjectManager	ref		
ObjectManagerCommunicationMechanism	ref		

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.7.12.7 Namespace Class

Namespace provides a domain (in other words, a container), in which the instances [of a class] are guaranteed to be unique per the KEY qualifier definitions. It is named relative to the CIM_ObjectManager implementation that provides such a domain.

Namespace is subclassed from ManagedElement.

Table 290 - Required Properties for Namespace

Class Properties	Type	Qualifier/ Parameter	Notes
SystemCreationClassName	string	key, maxlen(256)	
SystemName	string	key, maxlen(256)	
ObjectManagerCreationClassName	string	key, maxlen(256)	
ObjectManagerName	string	key, maxlen(256)	
CreationClassName	string	key, maxlen(256)	
Name	string	key, maxlen(256)	
ClassInfo	string	req	

7.3.7.12.8 NamespaceInManager

The NamespaceInManager is an association describing the Namespaces hosted by a CIM ObjectManager.

It is subclassed from Dependency.

Table 291 - Required Properties for NamespaceInManager

Class Properties	Type	Qualifier/ Parameter	Notes
Antecedent	ref		The ObjectManager containing a Namespace.
Dependent	ref		The Namespace in an ObjectManager.

7.3.7.12.9 RegisteredProfile

A RegisteredProfile describes a set of CIM Schema classes with required properties and/or methods, necessary to manage a real-world entity or to support a usage scenario, in an interoperable fashion. RegisteredProfiles can be defined by the DMTF or other standards' organizations. In the case of SMI-S, the SMI-S Profiles are defined by SNIA. Note that this class should not be confused with CIM_Profile, which collects SettingData instances, to be applied as a 'configuration profile' for an element.

A RegisteredProfile is a named 'standard' for CIM-based management of a particular System, subsystem, Service or other entity, for a specified set of uses. It is a complete, standalone definition, as opposed to the subclass RegisteredSubProfile, which requires a scoping profile for context.

For SNIA profiles, the uses for a RegisteredProfile or SubProfile shall be specified in a version of SMI-S. The name of the profile is defined and scoped by its authoring organization (e.g., SNIA).

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

RegisteredProfile is subclassed from ManagedElement.

Table 292 - Required Properties for RegisteredProfile

Class Properties	Type	Qualifier/ Parameter	Notes
InstanceID	string	key	This is a unique value for the profile instance.
RegisteredOrganization	uint16	req, maxlen(256)	This is an index into the value map of official name of the organization that created the Profile. For SMI-S profiles, this would be 11 (“SNIA”).
OtherRegisteredOrganization	string	maxlen(256)	This shall not be NULL if “Other” is identified in RegisteredOrganization.
RegisteredName	string	req, maxlen(256)	This is the name assigned by the organization that created the profile.
RegisteredVersion	string	req	This is the version number of the organization that defined the Profile.
AdvertiseTypes[ ]	uint16	req	Defines the advertisement of this profile. If the property is null then no advertisement is defined. A value of 1 is used to indicate “other” and a 3 is used to indicate “SLP”.
AdvertiseTypeDescriptions[ ]	string		This shall not be NULL if “Other” is identified in AdvertiseType.

7.3.7.12.10 RegisteredSubProfile

The RegisteredSubProfile class defines a SubProfile.

A subprofile is a named subset of a profile. The name of the subprofile is scoped by its parent profile.

RegisteredSubProfile is subclassed from RegisteredProfile.

Table 293 - Required Properties for RegisteredSubProfile

Class Properties	Type	Qualifier/ Parameter	Notes
InstanceID	string	key	This is a unique value for the subprofile instance.
RegisteredOrganization	uint16	req, maxlen(256)	This is the index into the value map of official names of the organization that created the subprofile. For SMI-S profiles, this would be 11 (”SNIA“).

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table 293 - Required Properties for RegisteredSubProfile (Continued)

Class Properties	Type	Qualifier/ Parameter	Notes
OtherRegisteredOrganization	string	maxlen(256)	This shall not be NULL if “Other” is identified in RegisteredOrganization.
RegisteredName	string	req, maxlen(256)	This is the name assigned by the organization that created the profile (or subprofile).
RegisteredVersion	string	req	This is the version number of the organization that defined the subprofile. It shall be the same as its parent profile.
AdvertiseTypes[ ]	uint16	req	Should be “Not Advertised” for subprofiles.
AdvertiseTypeDescriptions[ ]	string		This field should be null.

7.3.7.12.11 ReferencedProfile

This association is used to define a profile that is referenced by another RegisteredProfile.

The association is subclassed from Dependency.

Table 294 - Required Properties for ReferencedProfile

Class Properties	Type	Qualifier/ Parameter	Notes
Antecedent	ref		The RegisteredProfile that is referenced by the Dependent Profile.
Dependent	ref		A RegisteredProfile that references other profiles.

7.3.7.12.12 SubProfileRequiresProfile

This association is used to define the Subprofiles that are part of the defined RegisteredProfile. A subprofile requires another RegisteredProfile for context. This association mandates the scoping relationship between a subprofile and its scoping profile.

The association is subclassed from ReferencedProfile.

Table 295 - Required Properties for SubProfileRequiresProfile

Class Properties	Type	Qualifier/ Parameter	Notes
Antecedent	ref		The RegisteredProfile that is referenced/required by the subprofile.
Dependent	ref		A RegisteredSubProfile that requires a scoping profile, for context.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

NOTE This association is always between a top-level Profile and its subprofiles. This association cannot be between subprofiles.

7.3.7.12.13 ElementConformsToProfile

The ElementConformsToProfile association defines the standards (RegisteredProfile) to which a ManagedElement conforms. In the context of the Server Profile, it is used to identify the broadest scope to which the standard applies. It can be used to cover all of the namespace or to apply to a particular system.

The association is not subclassed from anything.

Table 296 - Required Properties for ElementConformsToProfile

Class Properties	Type	Qualifier/ Parameter	Notes
ConformantStandard	ref		The RegisteredProfile to which the ManagedElement conforms.
ManagedElement	ref		The ManagedElement that conforms to the RegisteredProfile.

The ManagedElements would typically be the System of the device that is represented by the profile.

7.3.7.13 Optional Subprofiles and Profiles.

Table 297 - CIM Server Profile Optional Subprofiles and Profiles

Optional Subprofiles & Profiles	Notes
ObjectManagerAdapter	

7.3.7.14 ObjectManagerAdapter Subprofile

7.3.7.14.1 Description

The ObjectManagerAdapter model defines the protocol adapters that are supported for a CIM Server. This model is optional for the CIM Server Profile. If implemented, the ObjectManagerAdapterModel shall adhere to the “required elements” table.

7.3.7.14.1.1 Standard dependencies

The ObjectManagerAdapter subprofile is defined using the CIM Schema 2.8. The subprofile will operate correctly with CIM Specification 2.2 (or later) and CIM Operations over HTTP 1.1 (or later).

7.3.7.14.1.2 Subprofile dependencies

The ObjectManagerAdapter subprofile introduces no Profile dependencies.

7.3.7.14.1.3 CIM Server Requirements

For the SMI-S uses of the ObjectManagerAdapter subprofile, support for Basic Read and Association Traversal functional profiles shall be supported (by the base Profile CIM server).

TheObjectManagerAdapter subprofile depends upon support for extrinsic methods.

The ObjectManagerAdapter subprofile is not advertised.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.7.14.1.4 Instance diagrams

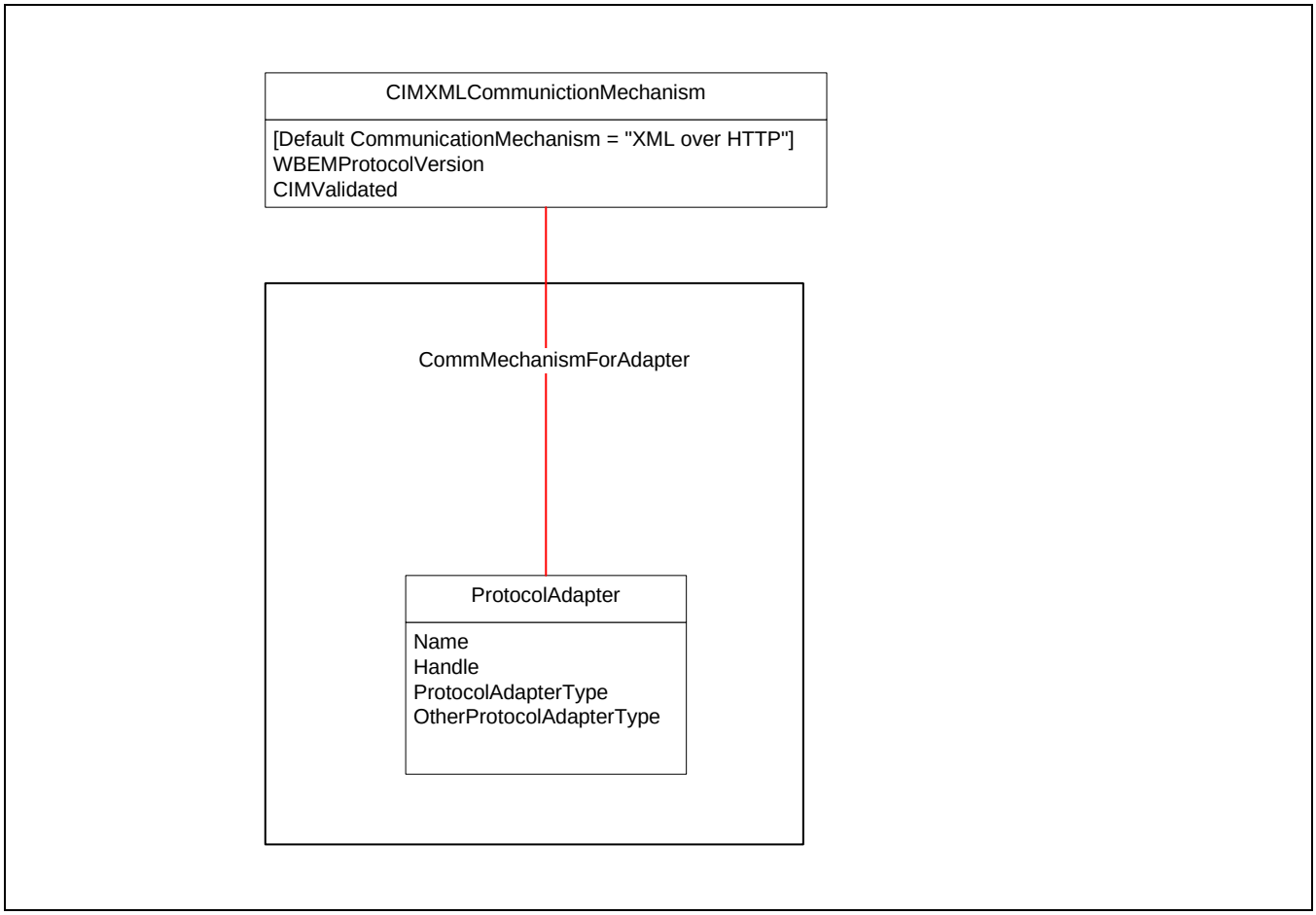


Figure 69 - ObjectManagerAdapter Subprofile Model

7.3.7.14.1.5 Durable names and correlatable IDs

The ObjectManagerAdapter subprofile does not introduce any objects that have durable names or correlatable IDs. And it does not use any durable names or correlatable IDs.

7.3.7.14.1.6 Methods

StartService()

StopService()

7.3.7.14.1.7 Client considerations

None.

7.3.7.14.1.8 Recipes

None.

7.3.7.14.1.9 Instrumentation requirements

None.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.7.14.1.10 Required CIM Elements

Table 298 - Subprofile Required Classes, Associations, Methods and Indications

Subprofile Class & Associations	Notes
ObjectManagerAdapter	
CommMechanisForAdapter	
Profile methods	**Notes**
StartService()	
StopService()	
Profile indications	**Notes**

7.3.7.14.1.11 Required Properties for CIM Elements

7.3.7.14.1.11.1 ObjectManagerAdapter

A ObjectManagerAdapter is a Service of the CIM Object Manager. It is responsible for accepting incoming requests on a particular protocol, and translating and forwarding the request to the CIM Object Manager. It is also responsible for translating and sending the response from the CIM Object Manager. This class is subclassed from WBEMService.

Table 299 - Required Properties for ObjectManagerAdapter

Class Properties	Type	Qualifier/ Parameter	Notes
OperationalStatus[ ]	uint16		
StatusDescriptions[ ]	string		This shall not be NULL if “Other” is identified in OperationalStatus.
SystemCreationClassName	string	key, maxlen(256)	
SystemName	string	key,mmaxlen(256)	
CreationClassName	string	key, maxlen(256)	
Started	boolean		
StartService()			
StopService()			
Name	string	override	
Handle	string	req	
AdapterType	uint16	req	
OtherAdapterType	string		

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.3.7.14.1.11.2 CommMechanismForObjectManagerAdapter

CommMechanismForObjectManager is an association between an ObjectManager's communication mechanism and a ObjectManagerAdapter that supports that mechanism to translate requests and responses for the Object Manager.

CommMechanismForObjectManager is subclassed from Dependency.

Table 300 - Required Properties for CommMechanismForObjectManagerAdapter

Class Properties	Type	Qualifier/ Parameter	Notes
Antecedent	ref	override	The specific ObjectManagerAdapter whose communication mechanism with the CIM Object Manager is described.
Dependent	ref	override, min(1)	The encoding/protocol/set of operations that may be used to communicate between the Object Manager and the referenced Object-ManagerAdapter.

7.3.7.14.1.12 Optional Subprofiles and Profiles

There are no optional subprofiles or profiles for this subprofile.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.4 Cross Client considerations

7.4.1 Overview

Many client applications are required to access data from multiple profiles to perform operations. This subclause describes algorithms that can be used to associate objects from different profiles to understand connections between the profiles. The algorithms use Durable Names to match objects from different profiles. Below are simplified instance diagrams that are used to illustrate the algorithms.

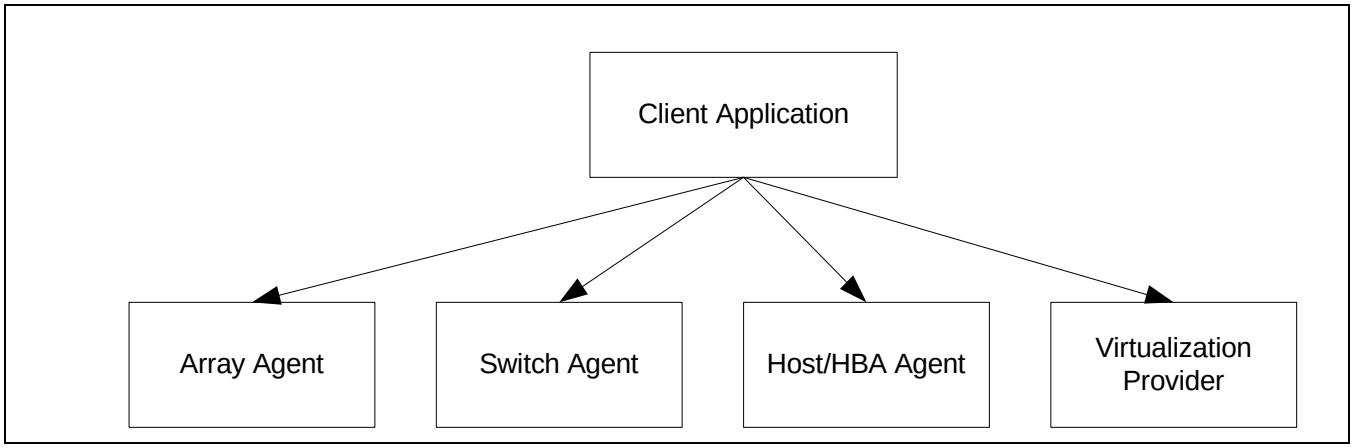


Figure 70 - System Diagram

7.4.1.1 HBA model

This model represents a simple “Host Bus Adapter”. The model includes objects that represent a single port Fibre channel HBA. The model also includes a storage volume being accessed through the HBA.

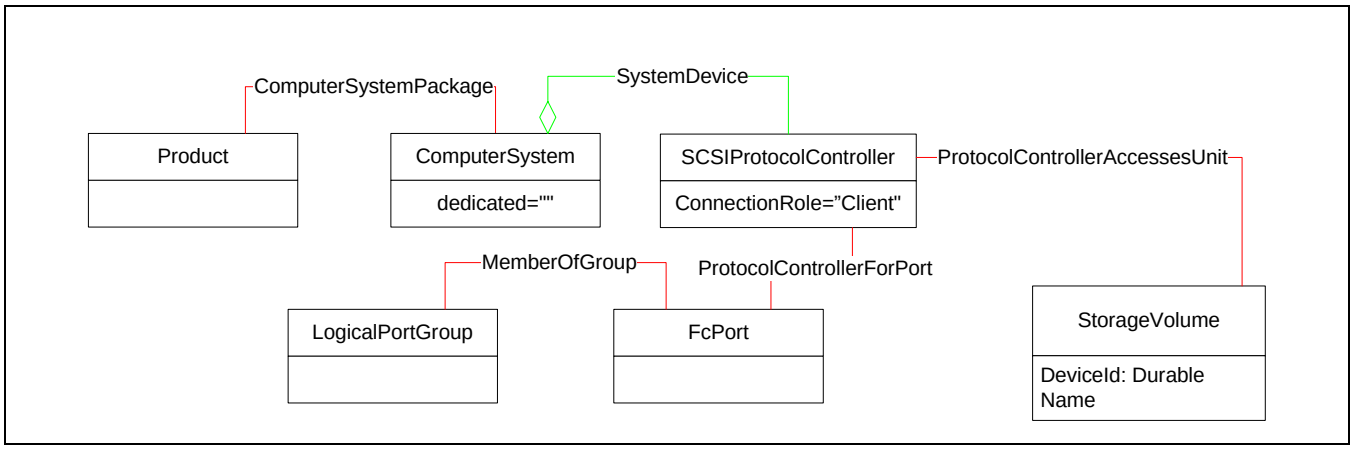


Figure 71 - Host Bus Adapter Model

7.4.1.1.1 Recipes

7.4.1.1.1.1 Determine storage device health

```
// DESCRIPTION
// Determine the health of the storage device given the health of the
// components. (Operational status on managed elements)
//
// PRE-EXISTING CONDITIONS AND ASSUMPTIONS
// None
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
// Check status on FCPorts and SCSIProtocolControllers
if (&systemDeviceOK($Device->))
{
     <health of device is OK>
}
```

7.4.1.2 Switch model

This model represents a two-port Fibre channel switch. The model also includes objects representing links to remote ports the switch agent knows about and ComputerSystems.

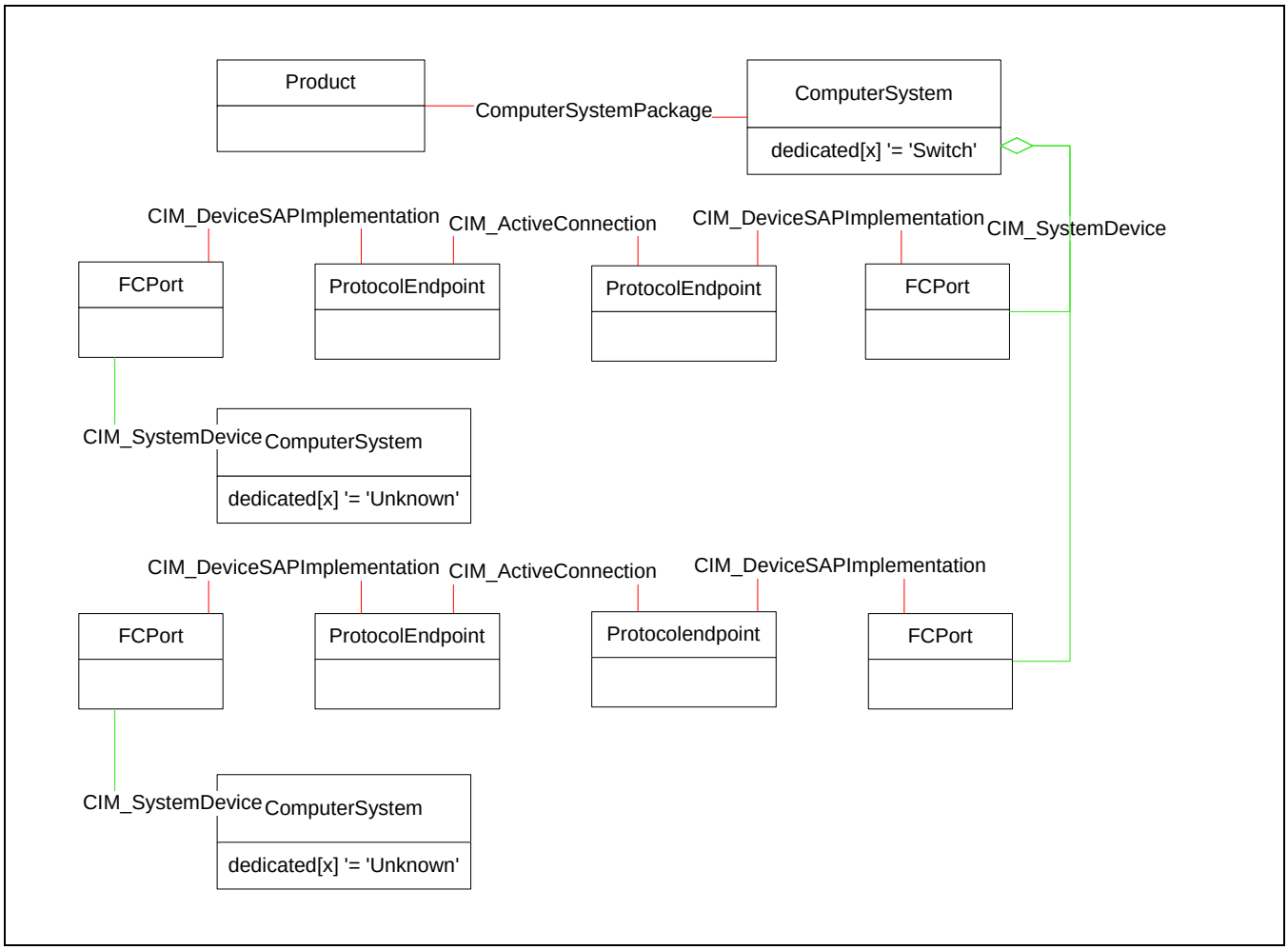


Figure 72 - Switch Model

7.4.1.3 Array model

This is a simple model of a disk array. The array has a single controller with a single Fibre channel port on the front end and a single parallel SCSI port for the disks. The model shows two disks that are members of a

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

single redundancy group. Part of the redundancy group is made available over the Fibre channel as a single volume.

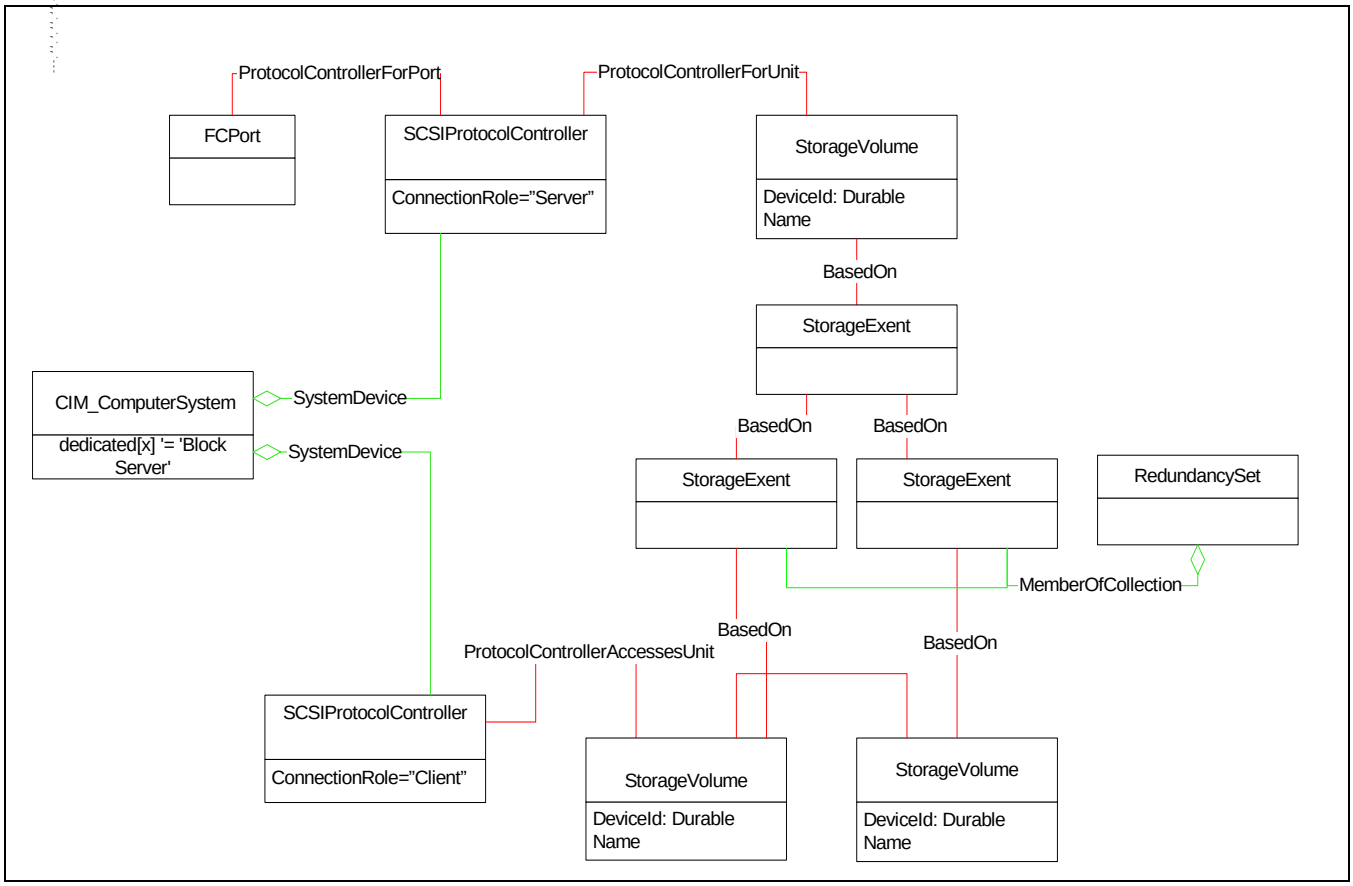


Figure 73 - Array Instance

7.4.1.3.1 Recipes

7.4.1.3.1.1 Determine the health of the Array

```
// DESCRIPTION
// Determine the health of the array given the health of the
// components. (Operational status on managed elements)
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
// None

$Device = GetInstance($Device->, false, false, false, null)
if (contains(2, $Device.OperationalStatus) && &systemDeviceOK($Device-
                                >))
{
     <health of device is OK>
}
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.4.1.4 Out of band virtualization model

This is a simple model of an out of band virtualization system. The system has a metadata controller and a single translation engine. The model also shows a single volume being used and a single volume being served to a host.

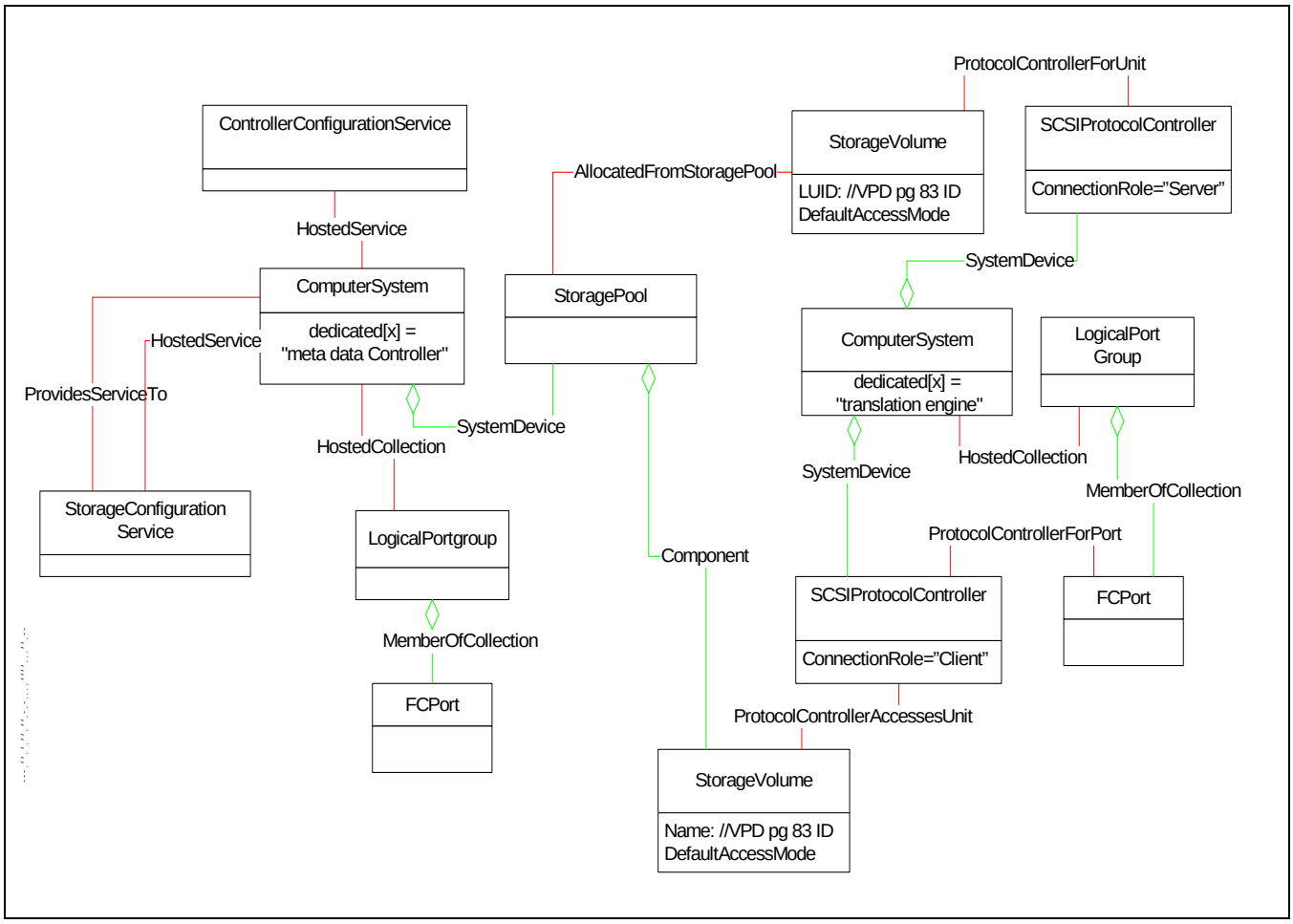


Figure 74 - Virtualization Instance

7.4.1.5 Durable names

Mapping objects across profiles and namespaces depends on “durable names”. Table 301 lists the names used in the subsequent examples

Table 301 - Cross-Profile Durable Names

Class	Reference	Example
FCPort	7.3.4.1.12.9, “FCPort”	WWN
StorageVolume	7.3.6.1.12.13, “StorageVolume”	VPD page 83 LU id

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.4.1.6 Fabric Topology (HBA, Switch, Array)

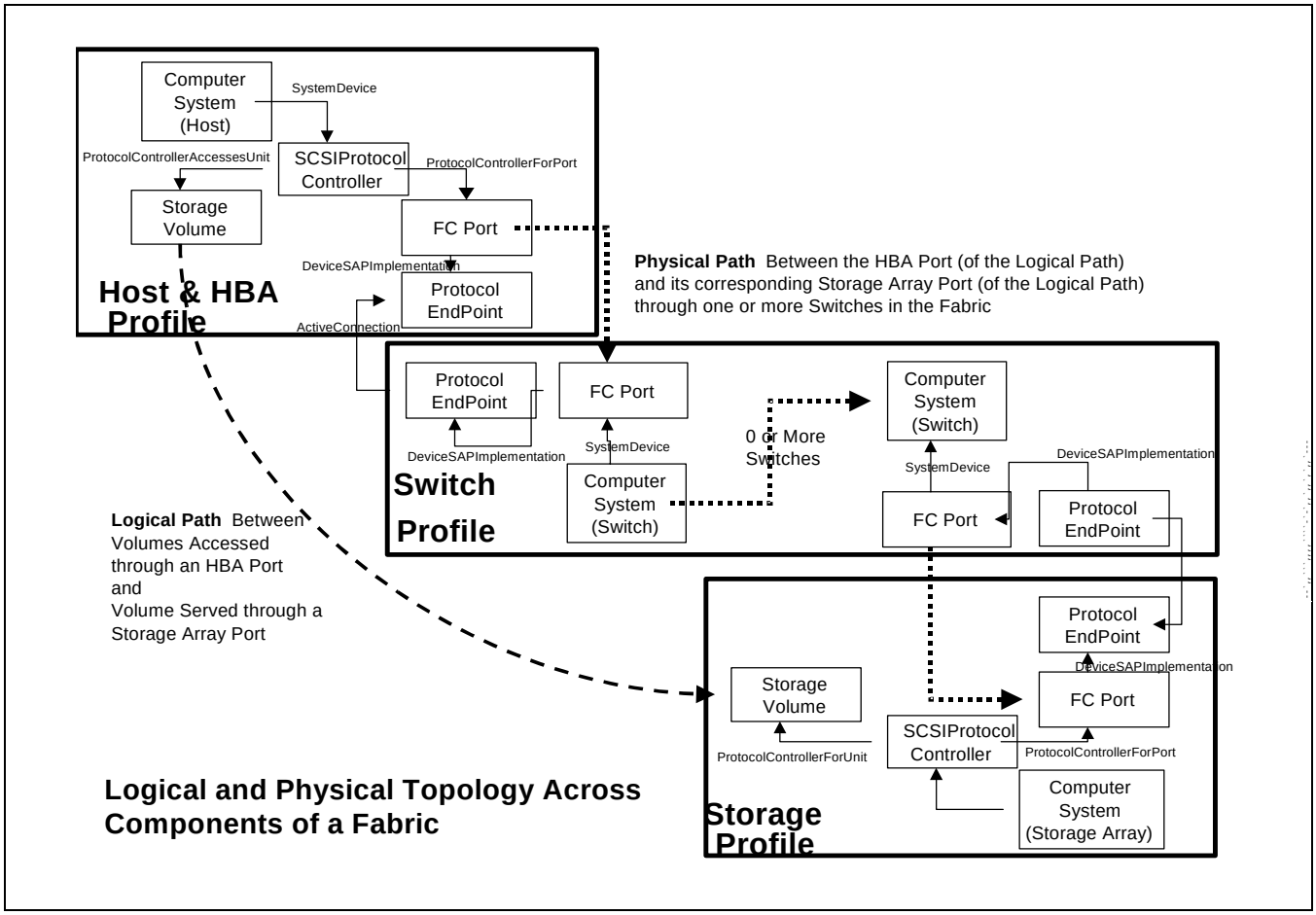


Figure 75 - Fabric Topology

A map of a SAN that shows all the elements and the connections between them is very useful. To create the map all the elements in the SAN with their Fibre channel ports are first located. Next the ports are linked together.

To locate all the elements in a SAN, you start by locating the agents. SMI-S agents are located using SLP. Once the agents are located, intrinsic methods are used to enumerate ComputerSystem objects. Each ComputerSystem object represents an element in the SAN. The ComputerSystem object's "Dedicated" attribute can be used to identify the type of the element.

After the elements are located, Fibre channel ports for each element are discovered. For each ComputerSystem object follow SystemDeviceFCPort objects and ProtocolController objects. For each ProtocolController object follow the ProtocolControllerForPort associations to FCPort objects. Use the information in the FCPort objects found to determine the Durable Name for the FCPort object. The Durable Name is used to match the ports to objects in other profiles.

To link the elements' ports together find the Switch elements. Switches know about ports on elements logged into their ports. To find this information start by locating the ComputerSystem objects that represents switches. Switches can be identified by the "Dedicated" attribute of the ComputerSystem object being set to "Switch". For each switch follow the SystemDeviceFCPort objects that represent the ports of the switch. Next look for ActiveConnection ActiveConnectionFCPort objects. These FCPort objects represent the ports on the other side of a link. Use attributes from the FCPort object to determine the Durable Name. These identifiers are then matched to identifiers found in other profiles to complete the connections.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.4.1.6.1 Recipes

7.4.1.6.1.1 General

Find all the switches in a SAN. Query fabric-profile agents to find the WWNs of all the device-ports connected into the switches. Query all the device-profile agents to find the WWNs of ports on devices. Correlate the data from the fabric-profile agents and the device-profile agents by matching WWNs (durable names).

Preexisting conditions and assumptions:

- All agents/namespaces supporting Fabric Profile (previously identified using SLP)
- All agents/namespaces supporting Array Profile
- All agents/namespaces supporting FC HBA Profile

7.4.1.6.1.2 Map FCPorts

```
// DESCRIPTION
// Create a map of how elements in a SAN are connected together via
                            Fibre-Channel ports
//
// The map is built in array $attachedFcPorts->[], where the index is a
// WWN of any device port on the SAN, and the value at that index is
// the object path of the connected switch port.
//
// First find all the switches in a SAN.  Get all the FCPorts for each
// switch and get the Attached FCPorts for each Switch FCPort.  Save
// these device ports in the map described above.

// PREEXISTING CONDITIONS AND ASSUMPTIONS
// 1.  All agents/namespaces supporting Fabric Profile previously
                            identified using SLP

// Do this for each CIMOM supporting Fabric Profile

switches[] = enumerateInstances("CIM_ComputerSystem", true, false,
                            true, true, null)

for #i in $switches[]
{
    if (!contains(5, $switches[#i].Dedicated))
        continue // only process switches, not other computer systems

    $fcPorts->[] = AssociatorNames(
        $switches[#i].getObjectPath(),
        "CIM_SystemDevice",
        "CIM_FCPort",
        "GroupComponent",
        "PartComponent")

    for #j in $fcPorts->[]
    {
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
            $protocolEndpoints->[] = AssociatorNames(
                 fcPorts->[#j],
                 “CIM_DeviceSAPImplementation”,
                 “CIM_ProtocolEndpoint”,
                 “Antecedent”,
                 “Dependent”);

            // NOTE - It is possible for this collection to be empty (ports
                              that are not
            // connected).  It is NOT possible for this collection to have
                              more than one
            // element
            if ($protocolEndpoints->[].length == 0)
                continue

            $attachedProtocolEndpoints->[] = AssociatorNames(
                $protocolEndpoints->[0],
             “CIM_ActiveConnection”,
                “CIM_ProtocolEndpoint”,
                null, null) // NOTE: role & resultRole are null as the
                            // direction of the association is not
                            // dictated by the specification

            for #k in $attachedProtocolEndpoints->[] {
                // $attachedFcPort is either a device port or an ISLÂ’d
             // switch port from another switch. We store this result
             // (i.e. which device FCPort is connected to which switch
             // FCPort) in a suitable data structure for subsequent
             // correlation to ports discovered on devices.
                $attachedFcPorts->[] = Associators(
            $attachedProtocolEndpoints->[#k],
            “CIM_DeviceSAPImplementation”,
            “CIM_FCPort”,
                    “Dependent”,
                    “Antecedent”,
                    false,
                    false,
                    [“PermanentAddress”])

               $attachedFcPort = $attachedFcPorts[0] // Exactly one member
                              guaranteed by model
                #wwn = $attachedFcPort.PermanentAddress
             $attachedFcPorts->[#wwn] = $fcPorts->[#j]
            }
        }
    }
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7.4.1.6.1.3 HBA to switch paths

```
// DESCRIPTION
// Determine physical path from HBA to switch.
//
// For each HBA port on every host, determine the connected switch
// port.  NOTE: Not every HBA port will be connected to a switch port,
// and not every switch port will be connected to a device port.  Only
// the connections between HBA ports and switch ports are discovered
// by this recipe
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
// 1.  All agents/namespaces supporting HBA Profile previously
                              identified using SLP
// 2.  Array $attachedFcPorts->[] is a map of how elements in a SAN are
// connected together via Fibre-Channel ports. Each index is a WWN of
// any device port on the SAN, and the value at that index is the
// connected switch port.

// Do this for each CIMOM supporting HBA Profile

$hosts[] = enumerateInstances(“CIM_ComputerSystem”)

for #i in $hosts->[]
{
    if (!contains(0, $hosts[#i].Dedicated))
        continue // only process systems that are “not dedicated”

    $fcPorts[] = Associators(
        $hosts[#i].getObjectPath(),
        “CIM_SystemDevice”,
        “CIM_FCPort”,
        “GroupComponent”,
        “PartComponent”,
        false,
        false,
        [“PermanentAddress”])

    for #j in $fcPorts[]
    {
        // Get the FCPort WWN
        #wwn = $fcPorts[#j].PermanentAddress

        // Match this device port WWN to one (or less) switch
        // ports, by using the mapping table
        $attachedSwitchPort-> = $attachedFcPorts->[#wwn]

       // Note - if there is no entry in the mapping array, this
       // port is not connected to any switch
```

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
    }
}
```

7.4.1.6.1.4 Determine physical path from Switch to Storage Arrays

```
// DESCRIPTION
// Determine physical path from Storage Arrays to Switches
//
// For each fibre-channel port on every array, determine the connected
// switch port.  NOTE: This identifies the FrontEnd I/O Controllers
// (and Storage Arrays) whose ports are physically connected to
// some of the ports of some of the Switches. This recipe does not
// distinguish and does not filter the front-end FC Port from the
// back-end FC Ports.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
// 1.  All agents/namespaces conforming to the Array profile previously
                            identified (see e.g. 7.3.7.9.2)
// 2.  Array $attachedFcPorts[] is a map of how elements in a SAN are
// connected together via Fibre-Channel ports. Each index is a WWN of
// any device port on the SAN, and the value at that index is the
// connected switch port.

// Do this for each CIMOM supporting the Array Profile:

// First identify upper-level computer systems for storage arrays -
// see 7.3.7.9.4 for how to use the Server profile to do this,
// or (as here) enumerate all systems within a conforming namespace
$computerSystems[] = enumerateInstances(“CIM_ComputerSystem”);

#n = 0
for #i in $computerSystems[]
{
    if (!contains(3, $computerSystems[#i].Dedicated))
        continue // only process systems that are dedicated “storage”

    if (!contains(15, $computerSystems[#i].Dedicated))
        continue // only process systems that are dedicated “block
                            server”

    $storageSystems[#n++] = $computerSystems[#i]
}

// Now accumulate all subsidiary computerSystems (cluster members or
// storage controllers) - treat $storageSystems[] as a queue and stuff
// newly discovered subsidiaries onto the end, so that ComponentCS
// associations are followed to arbitrary depth
```

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
#i = 0
while (#i < #n)
{
    $subsidiaries[] = Associators(
        $storageSystems[#i].getObjectPath(),
        “CIM_ComponentCS”,
        “CIM_ComputerSystem”,
        “GroupComponent”,
        “PartComponent”,
        false,
        false,
        null)

    for #j in $subsidiaries[]
    {
        $storageSystems[#n++] = $subsidiaries[#j]
    }

    #i++;
}

// Now get scoped FC ports for all the systems that have been
                           accumulated

// NOTE: Some of the ports contained will be back-end ports, but they
                           will
//      have no connectivity to switches, so we won’t distinguish them
//      from unconnected front-end ports

for #i in $storageSystems[]
{
    $fcPorts[] = Associators(
        $storageSystems[#i].getObjectPath(),
     “CIM_SystemDevice”,
     “CIM_FCPort”,
        “GroupComponent”,
        “PartComponent”,
        false,
        false,
        [“PermanentAddress”])

    for #j in $fcPorts[]
    {
     // Get the FCPort WWN
     #wwn = $fcPorts[#j].PermanentAddress

     // Match this device port WWN to one (or less) switch
     // ports, by using the mapping table
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
            $attachedSwitchPort-> = $attachedFcPorts->[#wwn]

            // Note - if there is no entry in the mapping array, this
            // port is not connected to any switch
        }
    }
```

7.4.1.7 Storage Connections (FC HBA, Array)

The Array profile includes objects and associations that represent the serving of SCSI volumes to the SAN. The HBA model represents the access of these volumes. To link them together locate the StorageVolume objects and use Durable Names to link them together.

To locate the volumes being accessed through the FC HBA use SLP to find agents that support the FC HBA profile. Use intrinsic methods to enumerate ComputerSystem objects on these agents. Use the “Dedicated” attribute to identify “host” systems. Use other attributes to identify the correct host. Next follow SystemDeviceProtocolController objects. The ProtocolController objects represent the HBAs on the host. From the ProtocolController objects follow ProtocolControllerAccessesUnit associations to StorageVolume association and its ProtocolControllerForUnit associations to determine the SCSI address and server of the volume.

To find the array that is serving the volume use SLP to locate agents that support the array profile. Use intrinsic methods to enumerate ComputerSystem objects from the agents. Then use the “Dedicated” attribute to identify “Block Server” systems. Use SystemDeviceProtocolController objects. Then follow ProtocolControllerForPort associations to find FCPort objects. Match attributes in the FCPort object with information from the ProtocolControllerAccessesUnit attributes associated with the HBA. When a match is found use ProtocolControllerForUnit and UnitAccessProtocolController object to locate StorageVolume objects. The StorageVolume objects can be matched to the StorageVolume objects from the Host/FC HBA profile using durable identifiers.

7.4.1.8 Zoning

Physical disks that are part of the storage pool are exported to consumers of virtualized storage via the Virtualization Appliance. An administrator may wish to ensure that all physical disks in the storage pool are not allowed direct access to the consumers of virtualized storage. At the same time, the administrator may want to allow disks that are not part of the storage virtualization pool access from hosts. In addition, consumers of virtualized storage may need to be selectively allowed access to a set of virtual disks that are exported by Translation Engines and Virtualization agents may selectively be allowed access to certain disks from the pool. The controlled access can be provided by port or LUN level zoning within the fabric.

Zone members (ports, nodes, LUNs, etc.) are the elements that can be zoned together. While either port (or node) level zoning may be used for virtualization, LUN zoning gives a higher level of granularity. The Virtualization Appliance would have to create and activate zone sets that have zones whose members are either ports or LUNs. The zone set may be configured such that:

physical disks that are part of the StoragePool can be zoned out from consumers of virtualized storage to prevent these consumers from directly accessing the disks;

all physical disks that are not part of the virtualized pool can be zoned with hosts in such a way that these hosts can have direct access;

virtual disks that are exported by Translation Engines can selectively be zoned with consumers of virtualized storage. Translation Engines would have to present a target port address for zoning;

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

physical disks that should be accessible to Translation Engines can be zoned together.

The Virtualization Appliance should be allowed access to all physical storage in the storage pool.

7.4.1.9 Fabric Route Discovery

In order to load balance such that the virtual disks that are exported to consumers of virtualized storage based on the available bandwidth or redundancy in the fabric, from the Translation Engines to the physical storage, the virtualization appliance would have to know the routes available in the fabric. Access to the physical disks from Translation Engines may then take place via multiple paths based on the available routes. The virtualization appliance may wish to access data from multiple providers in order to perform the fabric topology discovery operations to be able to determine which HBAs, switches and storage devices exist in the fabric and how they are connected and then determine the available routes in the fabric. Similarly, the management application client may first want to determine the fabric topology and routes and then know the zoning configuration to determine whether the HBA is permitted to connect to the device port or to access its LUNs.

A topological map of a SAN that shows all the elements and the connections between them is very useful. To create the map, all the elements in the SAN with their Fibre channel ports are first located. Next the ports are linked together. Next the LUNs that can be accessed via the ports are discovered. Having built the physical topology map, routes can be detected using the ActiveConnection association between ports.

7.4.1.10 Durable names

In order to find out the zoning constraints, the Zones of the active ZoneSet are discovered and the ZoneMemberSettingData is discovered. The ZoneMemberID is either the port WWN or a LU ID and hence the association is made to the zone member object.

Mapping objects across profiles and namespaces depends on “durable names”. Table 302 provides the durable names used in the examples below.

Table 302 - Cross Profile Durable Names

Class	Reference	Example
FCPort	See 7.3.3.13.12.1, “FCPort”	WWN
StorageVolume	See 7.3.6.1.12.13, “StorageVolume”	VPD page 83 LU ID

7.4.2 General recipes

7.4.2.1 Indications’ Status

```
// DESCRIPTION
// Determine if the indication subscription requested already exists.
                              If
// not, then attempt to create the indication subscription passed in. If
// the CIM Server does not support the addition of indication, then the
// CIM Client will need to poll for these instance changes. This recipe
// does not handle the issue of providing the target URL for
                              indications.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
// 1.The namespace of interest has previously been identified and
//   defined in the #SomeNameSpace variable
// 2.The list of filters of interest has been previously built in the
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
//   #filters[] array. Each element in this array is the WQL filter
                            itself

// FUNCTION: createIndication
sub createIndication ($Filter)
{
     try {
        <create indications as per SMIS specification>
     } catch(CIM_ERROR_NOT_SUPPORTED) {
        <setup this class of instances to be polled for>
     }
}

// MAIN
$ExistingInstances[] = EnumerateInstances(#SomeNameSpace,
                            “CIM_IndicationFilter”)
for #i in $ExistingInstances
{
     for #j in #filters
     {
        if(!compare($ExistingInstances[#j].Query, #filters[#j])
        {
           &createIndiciation(#filters[#j])
        }
     }
}
```

7.4.2.2 Listenable Instance Notification

```
// DESCRIPTION
// Create an indication subscription for every indication that is
// required by the profile.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTIONS
// 1.The namespace of interest has previously been identified and
//   defined in the #SomeNameSpace variable

#filters[] = <array of SMIS filters for the target profile>
@{Determine if Indications already exist or have to be created} #filters
```

7.4.2.3 Life Cycle Event Subscription description

```
// DESCRIPTION
// Create an indication subscription for the operational status for a
// computer systems defined within a given CIM agent and namespace. This
// subscription   will only be made in those CIM agents that have SAN
// devices or applications of   interest defined in them. The client
                            will
// have to determine once having received the indication, whether the
// computer system related to this indication (AlertingManagedElement
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
// attribute) is of interest. This recipe does not handle the target URL
// for the indication.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTIONS
// None

#filter[0] = “SELECT * FROM CIM_InstModification
    WHERE SourceInstance ISA CIM_ComputerSystem
      AND SourceInstance.OperationalStatus[0] <>
          PreviousInstance.OperationalStatus[0]”
@{Determine if Indications already exist or have to be created} #filter
```

7.4.2.4 Subscription for alert indications

```
// DESCRIPTION
// Create an indication subscription for the alert indications defined
// within a given CIM agent and namespace. This subscription will only
                              be
// made in those CIM agents that have SAN devices or applications of
// interest defined in them. The client will have to determine once
                              having
// received the indication, whether the computer system related to this
// indication (AlertingManagedElement attribute) is of interest.   Each
// specific alert indication will have also specific handling required
// for it by the CIM Client.
// NOTE: This recipe does not handle the target URL for the indication.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTIONS
// None

#filter[0] = “SELECT * FROM CIM_AlertIndication”
@{Determine if Indications already exist or have to be created} #filter
```

7.4.2.5 Listenable Interface Modification Notification

```
// DESCRIPTION
// Create an indication subscription for every indication
// that is required by the profile
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
// 1.The namespace of interest has previously been identified and
//   defined in the #SomeNameSpace variable

#filters[] = <array of SMIS filters for the target profile>
@{Determine if Indications already exist or have to be created} #filters
```

7.4.2.6 Subscribe for Lifecycle Events where OperationalStatus Changes

```
// DESCRIPTION
// Create an indication subscription for the operational
// status for computer systems defined within a given CIM agent and
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
// namspace.  This subscription will only be made in those CIM agents
// that have SAN devices or applications of interest defined in them.
                              The
// client will have to determine once having received the indication,
// whether the computer system related to this indication
// (AlertingManagedElement attribute) is of interest.  This recipe does
// not handle the target URL for the indication.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
// None

#filter[0] = “SELECT * FROM InstModification
  WHERE SourceInstance ISA CIM_ComputerSystem
    AND SourceInstance.OperationalStatus[0] <>
        PreviousInstance.OperationalStatus[0]”
@{Determine if Indications already exist or have to be created} #filter
```

7.4.2.7 Subscription for alert indications

```
// DESCRIPTION
// Create an indication subscription for the alert
// indications defined within a given CIM agent and namspace.  This
// subscription will only be made in those CIM agents that have SAN
// devices or applications of interest defined in them. The client will
// have to determine once having received the indication, whether the
// computer system related to this indication (AlertingManagedElement
// attribute) is of interest. Each specific alert indication will
// have also specific handling required for it by the CIM Client.  This
// recipe does not handle the target URL for the indication.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
// None

#filter[0] = “SELECT * FROM AlertIndication”
@{Determine if Indications already exist or have to be created} #filter
```

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

8 Security

8.1 Introduction

Security requirements can be divided into four major categories: authentication, authorization, confidentiality and integrity (including non-repudiation), brief definitions follow. Authentication is verifying the identity of an entity (client or server). Authorization is deciding if an entity is allowed to perform a given operation. Confidentiality is restricting information to only those intended recipients. Integrity is guaranteeing that information, passed between entities, has not been modified.

This standard is primarily concerned with authentication and confidentiality. Authorization is not covered and is implementation dependent. Valid implementations may assume an entity to be authorized if the entity has been authenticated, or they may perform an authorization check. Additionally, storage of and access to authorization rules, necessary for an authorization check, is implementation dependent. Specification of an interoperable authorization method is left for future work.

Other issues not covered include threat models, protection against specific attack vectors (such as denial of service, replay, buffer overflow, man in the middle, etc.), topics related to key management and data integrity. Development of threat models and specific attack countermeasures required for robust security elements, such as integrity has been left for future work.

Security concerns occur in three areas of a SMI-S implementation. First, a device, such as a switch, may require a login before discovery or operations such as zoning can be performed. The information needed to perform this login is generically referred to as “device credentials”. A SMI-S server or provider needs to obtain these credentials in order to talk to the device, and they should be provided confidentially.

Second, a SMI-S Client may need to authenticate itself to a SMI-S Server. Not all Clients may be allowed to query the object model, and not all Clients may be allowed to perform operations on objects in the model. Authenticating the client is the first step in determining what that Client is allowed to do.

Thirdly, within the SMI-S implementations themselves, should developers be unaware of secure development practices, attackers maybe able to exploit insecurely developed implementations. (Note, potential attacks might include, but not be limited to buffer overflows, obtaining secure information handled by the SMI-S implementation, like passwords, etc.) In an effort to increase the general knowledge of SMI-S developers, for secure development practices, one source is recommended (Building Secure Software by Gary McGraw and John Viega (ISBN: 020172152X).

8.2 Background

Section 4.4 of “Specification for CIM Operations over HTTP, Version 1.1” from DMTF describes the requirements for CIM clients and servers. The authentication methods referred to in the above specification are described in the IETF RFCs 1945 and 2068, “Hypertext Transfer Protocol - HTTP/1.0(1.1)” and IETF RFC 2069 “An Extension to HTTP: Digest Access Authentication””. The Transport Layer Security Protocol Version 1.0 (TLS) is defined in IETF RFC2246. The Secure Sockets Layer 3.0 (SSL 3.0) protocol specification can be downloaded from HTTP://wp.netscape.com/end/ssl3/.

Section 4.4 of “Specification for CIM Operations over HTTP, Version 1.1" defines additional requirements for HTTP authentication, above those found in HTTP 1.1 [RFC 2068], or the HTTP authentication documents [RFC 2069, RFC 2617]. HTTP authentication generally starts with an HTTP client request, such as “GET Request-URI” (where Request-URI is the source requested). If the client request does not include an “Authorization” header line and authentication is required, the server responds with a “401 unauthorized” status code, and a “WWW-Authenticate” header line. The HTTP client shall then respond with the appropriate “Authorization” header line in a subsequent request. The format of the “WWW-Authenticate” and “Authorization” header lines varies depending on the type of authentication required: basic authentication or digest authentication. If the authentication is successful, the HTTP server will respond with a status code of “200 OK”.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Basic authentication involves sending the user name and password in the clear and should only be used on a secure network, or in conjunction with a mechanism that ensures confidentiality, such as TLS. Digest authentication sends a secure digest of the user name and password (and other information including a nonce value), so that the password is not revealed. "401Unauthorized" responses should not include a choice of authentication.

SSL 3.0 and TLS provide for confidentiality and integrity in communication. An initial handshake defines a session key, which is used to encrypt the data with a symmetric algorithm, such as RC4. A keyed secure hash, such as SHA-1 is used to check message integrity. For interoperability, the initial handshake defines the algorithms to be used for message encryption and hashing.

8.3 Requirements

8.3.1 General

SMI-S Servers and Clients shall conform to section 4.4 of “Specification for CIM Operations over HTTP, Version 1.1”. HTTP Basic Authentication shall be implemented. HTTP Digest Authentication implementation is desirable. Using HTTP Basic Authentication only in conjunction with SSL 3.0 or TLS will minimize the compromising of user IDs and passwords. In the case of clear text CIM-XML over HTTP, HTTP Digest Authentication usage is encouraged. This standard determines the protocol for authentication between a Client and the SMI-S Server, but not the mechanism of authentication used by the SMI-S Server.

SSL 3.0 shall be supported. In the 1.1 version of the SMI-S specification, this requirement is expected to change to state that TLS shall be supported. Appendix E of RFC 2246 describes backwards compatibility between TLS and SSL 3.0.

Clients that fail to contact an SMI-S server via HTTP on TCP port 5988 should retry with HTTP over SSL 3.0 or TLS on TCP port 5989. Clients whose security policy requires use of a secure channel should not attempt initial contact via HTTP on TCP port 5988. Servers can accelerate discovery that a secure channel is needed by responding to HTTP contacts on TCP port 5988 with an HTTP REDIRECT to the appropriate https: URL (HTTP over SSL or TLS on TCP port 5989) to avoid the need for clients to timeout the HTTP contact attempt. Clients are encouraged to honor such redirects in this situation.

Client authentication to the SMI-S Server is based on an authentication provider. A provider plug-in allows for differing authentication schemes. Possible mechanisms include host-based authentication, Kerberos, PKI or other.

A SMI-S Server can be configured with the device credentials necessary to talk to the device. If a SMI-S Server supports SSL 3.0 or TLS, the Client shall use 3.0 or TLS to pass device credentials to the SMI-S Server. When new device credentials are passed to a SMI-S Server, the device credential information in the device shall be updated immediately.

Only the SMI-S Server responsible for communicating with the device has access to the properties of the SharedSecret object. No other SMI-S Client may read the Secret property of this object as it shall be implemented Write-Only.

8.3.2 Certificate usage with SSL 3.0 and TLS

8.3.2.1 Functional goals

SSL 3.0 and TLS are based on the use of certificates for authentication. The requirements in this subclause apply to any SMI-S implementation that supports SSL 3.0 or TLS. These requirements are intended to accomplish the following functional goals:

a) require support for existing common practice for certificate usage;

b) allow customers to enforce their own certificate usage and acceptance policies;

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

c) default the facilitating interoperability where not specifically disallowed by security policy;

d) require support for certificate acquisition from and revocation by common Public Key Infrastructure (PKI) and Certification Authority (CA) software;

e) allow security policy control to be restricted to security administrators.

These goals are realized by the requirements stated below, organized by functional goal.

8.3.2.2 Requirements

a) Require support for existing common practice for certificate usage.

Server certificates shall be supported, client certificates are optionally supported. A server certificate is presented by the server to authenticate the server to the client. Similarly, a client certificate is presented by the client to authenticate the client to the server. For public web sites offering secure communications via SSL or TLS, server certificate usage is common, but client certificates are rarely used.

b) Allow customers to enforce their own certificate usage and acceptance policies.

All certificates identifying SMI-S management entities and their associated private keys shall be replaceable by certificates from any Certification Authority (CA). All certificate acceptance policies in clients and servers shall be configurable. Any client or server that accepts and checks certificates from a counterpart shall support multiple CA root certificates and have an interface that allows CA root certificates to be added and removed. CA root certificates are used to verify that a certificate has been signed by a key from an acceptable certificate authority.

c) Default to facilitating interoperability where not specifically disallowed by security policy.

Interactive clients are encouraged to provide a means to query the user about acceptance of a certificate from an unrecognized certificate authority (no corresponding CA root certificate installed in client) and accept responses allowing use of the certificate presented or all certificates from the issuing CA. Servers are encouraged to not accept unrecognized certificates. It is expected that a limited number of CAs will be acceptable for client certificates in any site that uses them.

Preconfiguring root certificates from widely used CAs is optional, but simplifies initial configuration of certificate-based security, as certificates from those CAs will be accepted. These CA root certificates can be exported from widely available web browsers.

d) Require support for certificate acquisition from and revocation by common PKI/CA software.

All interfaces for certificate configuration in (b) and (c) above shall support the following certificate formats:

- DER encoded X.509
  International Telecommunications Union Telecommunication Standardization Sector (ITU-T), Recommendation X.509: Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks, May 2000.
  Specification and technical corrigenda can be obtained from:
  http://www.itu.int/ITU-T/publications/recs.html;
- Base64 encoded X.509 (often called PEM)
  N. Freed and N. Borenstein, Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies, IETF RFC 2045, November 1996, Section 6.8.
  Available at: http://www.ietf.org/rfc/rfc2045.txt;
- PKCS#12
  RSA Laboratories, PKCS #12: Personal Information Exchange Syntax, Version 1.0, June 1999. Specification and Technical Corrigendum. Available at:
  http://www.rsasecurity.com/rsalabs/pkcs/pkcs-12/index.html.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

All certificate validation software shall support local certificate revocation lists and at least one list per CA root certificate supported. Support is required for both DER encoded X.509 and Base64 encoded X.509 formats, but this support may be provided by using one format in the software and providing a tool to convert lists from the other format. OCSP and other means of immediate online verification of certificate validity are OPTIONAL, as connectivity to the issuing Certificate Authority cannot be assured.

e) Allow security policy control to be restricted to security administrators.

All certificate interfaces required above shall support access restrictions that permit access only by suitably privileged administrators. A suitably privileged security administrator shall be able to disable functionality for acceptance of unrecognized certificates described in (c) above.

The above requirements can be satisfied via appropriate use of the readily-available OpenSSL toolkit software (www.openssl.org). Support for PKCS#7 certificate format was deliberately omitted from the requirements. This format is primarily used for online interaction with certificate authorities. This kind of functionality is not required from all SMI-S storage management software, and tools are readily available to convert PKCS#7 certificates to or from other certificate formats.

8.4 Modelling device credentials

8.4.1 General requirements

For a complete discussion of the SMI-S requirements for modelling device credentials, see 7.3.3.12, “Device Credentials subprofile”.

8.4.2 Instrumentation requirements

The SMI-S Server will securely store the device credentials local to the SMI-S Server. A proxy SMI-S Server may need to store the credentials on disk so that they are available upon reboot. In this case, the credentials will need to be encrypted for confidentiality.

To preserve this confidentiality, the device credentials will need to be transmitted securely from the SMI-S Server to the device. The mechanism of communicating the credentials to the device is outside the scope of this standard, but use of a secure channel is recommended where possible.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

9 Service Discovery

9.1 Objectives

Service discovery in the context of SMI-S refers to the discovery of dedicated SMI-S servers, general purpose SMI-S servers and directory servers, and the functions they offer in an SMI-S managed environment. The specific objectives to be addressed by the discovery architecture are the following.

a) Provide a mechanism that allows SMI-S clients to discover the SMI-S constituents in a storage network environment so that they may communicate with these constituents using CIM Operations over HTTP protocol. This includes:

   1) finding the address for the SMI-S constituent;

   2) finding the capabilities of the server, including communications capabilities, security capabilities, CIM operational capabilities and the functional capabilities (CQL, Batch operations support, etc.).

b) Provide a mechanism that is efficient with respect to the amount of information exchanged with minimal exchanges to acquire the information.

c) Provide a mechanism that accurately defines the services in the network, independent of whether or not those services are currently available.

d) Provide a mechanism that provides information on namespaces provided and the CIM Schema supported.

e) Provide a mechanism that allows SMI-S clients the profile(s) supported by agents and object managers.

f) Provide a mechanism that scales to enterprise environments.

g) Utilize existing standard mechanisms to effect the SMI-S service discovery to enable rapid deployment.

h) Provide a mechanism that allows SMI-S clients to determine the level of (SMI-S) support provided by the constituents (e.g., R1, R2, etc.).

9.2 Overview

SMI-S Release 1 uses the Service Location Protocol Version 2 (SLPv2), as defined by IETF RFC2608, for its *basic* discovery mechanism. SLPv2 is used to locate constituents (agents, object managers, etc.), but complete discovery of all the services offered involves traversing the interoperability model for the SMI-S profile supported. This clause deals primarily with the information discovered using SLPv2. There are references to information discovered by traversing the interoperability model, but details are given in 10.3, “Dedicated SMI-S Server”.

SLPv1 is not supported in SMI-S as discovery mechanism. SMI-S requires capabilities that were introduced in SLPv2 in order to support the discovery of SMI-S agents and object managers. SLPv2 defines discovery protocols among three constituents.

User Agent (UA): A process that attempts to establish contact with one or more services. A User Agent retrieves service information from Service Agents or Directory Agents. In SMI-S, a “user agent” would be part of an SMI-S Client.

Service Agent (SA): A process working on behalf of one or more services to advertise the services. In SMI-S, a “service agent” would be supported by SMI-S dedicated or general purpose servers.

Directory Agent (DA): A process that caches SLP service advertisements registered by Service Agents and forwards the service advertisements to User Agents on demand. In SMI-S, the SLP “Directory agent” is defined as the main function of the “directory server” role in the SMI-S Reference Model.

SLPv2 provides a framework for client applications, represented by User Agents, to find and utilize services, represented by Service Agents. The Directory Agent represents an optional part that enhances the

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

performance and scalability of the protocol by acting as a cache for all services that have been advertised. Directory Agents also reduce the load on Service Agents, making simpler implementations of Service Agents possible. User Agents can then query the Directory Agent for services. Service Agents register with Directory Agents and are required to re-register as the registrations expire. If no Directory Agent is present, User Agents are allowed to request service information directly from the Service Agents.

Using SLPv2, a client can discover SMI-S servers and SLPv2 Directory Agents in the storage network. In the case of SMI-S servers, the basic information given are the profiles supported and the URL of the service. Details on the specific services provided with the profile are found by traversing the service structure modelled for the profile.

Using SLPv2, a “service agent” advertises its services. These advertisements have an expiration time period. To avoid getting an advertisement deleted, a service agent shall reregister before the time period expires. SMI-S servers can deregister as part of a graceful shutdown.

NOTE While SLP security topics are important they will be specified in future revisions of the SMI-S standard.

A service advertisement consists of file following components.

- Service type name – describes the general type of service being advertised (for example, Printing, faxing, etc.). The working assumption is that DMTF wants “WBEM Servers” advertised with the service type WBEM. This is used by SMI-S servers (both dedicated and general purpose servers).
- Attributes – The collection of attributes describes the particular instance of the service in more detail. For SMI-S, these would be the attributes defined by the service type template for WBEM. The attributes are defined in 9.5.3, “Service Attributes”.
- Service Access point – the service access point defines the point of connection that the software client of the UA uses to connect to the service over the network.
- Scopes – These are administrative groupings of services. The default value (“default”) is commonly used for SMI-S servers. Other scopes may be defined by the customer, but care shall be taken when this is done. The administrator shall do this correctly or SMI-S servers will not be visible. All the SMI-S recipes assume that DEFAULT is set for scopes.
- Language – Services advertisements contain human readable strings. These are provided in English, but may also be in other languages.

9.3 SLP messages

SLP v2 divides the base set of SLP messages into required and optional subsets.

NOTE SLPv2 also includes a new feature, an extension format. Extension messages are attached to base messages. SMI-S does not use extensions. The discussion of messages introduces the following terms that define the SLP services:

Attribute Reply (AttrRply): A reply to an Attribute Request. (optional)

Attribute Request (AttrRqst): A request for attributes of a given type of service or attributes of a given service. (optional)

DA Advertisements (DAAdvert): A solicited (unicast) or unsolicited (multicast) advertisement of Directory Agent availability.

SA Advertisement (SAAdvert): Information describing a service that consists of the Service Type, Service Access Point, lifetime and Attributes.

Service Acknowledgement (SrvAck): A reply to a SrvReg request.

Service Deregister (SrvDereg): A request to deregister a service or some attributes of a service. (optional)

Service Register (SrvReg): A request to register a service or some attributes of a service.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Service Reply (SrvRply): A reply to a Service Request.

Service Request (SrvRqst): A request for a service on the network.

Service Type Reply (SrvTypeRply): A reply to a Service Type Request. (optional)

Service Type Request (SrvTypeRqst): A request for all types of service on the network. (optional)

Service Agents (SAs) and User Agents (UAs) shall support Service Request, Service Reply and DAAdvertisement message types. Service Agents shall additionally support Service Registration, SA Advertisement and Service Acknowledgement message types. The remaining message types may be supported by Service Agents and User Agents. Directory Agents (DAs) shall support all message types with the exception of SA Advertisement. Table 303 lists each base message type, its abbreviation, function code and required/optional status.

Table 303 - Message Types

Message Type	Abbreviation	Function Code	Required (R)/ Optional (O)		
			DAs	SAs	UAs
Service Request	SrvRqst	1	R	R	R
Service Reply	SrvRply	2	R	R	R
Service Registration	SrvReg	3	R	R	O
Service Deregistration	SrvDereg	4	R	O	O
Service Acknowledgement	SrvAck	5	R	R	O
Attribute Request	AttrRqst	6	R	R	R
Attribute Reply	AttrRply	7	R	R	R
DA Advertisement	DAAdvert	8	R	R	R
Service Type Request	SrvTypeRqst	9	R	O	O
Service Type Reply	SrvTypeRply	10	R	O	O
SA Advertisement	SAAdvert	11		R	O

NOTE The requirements in Table 303 extend the requirements defined for SLP, version 2. This standard adds additional requirements for AttrRqst and AttrRply beyond those defined by the RFC.

9.4 Scopes

SLPv2 defines a scope as follows:

Scope: A set of services, typically making up a logical administrative group.

Scopes are sets of service instances. The primary use of scopes is to provide the ability to create administrative groupings of services. A set of services may be assigned a scope by network administrators. A User Agent (UA) seeking services is configured to use one or more scopes. The UA only discovers those

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

services that have been configured for its use. By configuring UAs and Service Agents with scopes, administrators may make services available. Scopes strings are case insensitive. The default SCOPE string is “DEFAULT”.

SMI-S does not dictate how Scopes are set. That is, scopes can be set by customers to match their needs. However, SMI-S requires that SMI-S servers use the “default” scope as a means of making SMI-S advertisements visible to SMI-S clients.

To be compliant with SMI-S, User Agents (SMI-S clients) and Service Agents (SMI-S servers) shall not require scope settings that get in the way of administrator use of scopes. Specifically, this means:

- SMI-S clients and servers shall allow an administrator to set scopes to define what is to be searched, and,
- SMI-S clients and servers shall allow an administrator to configure scopes, including turning off the “default” scope.

9.5 Services definition

9.5.1 General

Services definition uses the following terms defined in SLPv2.

Service Type Template: A formalized, computer-readable description of a Service Type. The template defines the format of the service URL and attributes supported by the service type.

Service URL: A Uniform Resource Locator for a service containing the service type name, network family, Service Access Point and any other information needed to contact the service.

Services are defined by two components: the Service URL and the Service Type Template. The Service URL defines an access point for the service and identifies a unique resource in the network. Service URLs may be either existing generic URLs or URLs from the service: URL scheme.

The second component in a Service definition is a Service Type Template. Service Type Templates define the attributes associated with a service. These attributes, through inclusion in registrations and queries, allow clients to differentiate between similar services.

SMI-S servers use a Service Type Template defined in DMTF for advertising “WBEM Servers” (e.g., CIMOMs). The template name for WBEM Servers is “WBEM”.

9.5.2 Service Type

Service Type: The class of a network service represented by a unique string (for example a namespace assigned by IANA).

The service type describes a class of services that share the same attributes (e.g., the service printer or the service “WBEM”). DMTF is considering an SLP-based discovery mechanism that locates “WBEM” (e.g., CIMOMs). The SMI-S design builds on the DMTF proposal.

The basic function of SLP discovery is the identification of the service offered by a constituent. In the case of SMI-S, the service type advertised by all constituents is “WBEM.” This follows a DMTF proposal for advertising WBEM Servers. The only exception to this is the Directory Server, which advertises itself as a “directory-agent.” That is, SMI-S uses a standard SLP directory service.
SMI-S does not require a unique SMI-S directory server.

For other roles (SMI-S servers) the role advertises its services as a WBEM services (e.g., “WBEM”).

9.5.3 Service Attributes

Attributes: A collection of tags and values describing the characteristics of a service.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

SMI-S servers shall advertise a standard set of attributes. These attributes are the following.

- Service-hi-name – This is the name of the service for use in human interfaces.
- Service-hi-description – This is a description of the CIM service that is suitable for use in human interfaces.
- Service-id – A unique ID for the CIM Server that is providing the service.
- Service-location-tcp – This is a list of TCP addresses that can be used to reach the service.

NOTE This need only be one (for CIM-XML). But it could hold others (for other communications protocols).

- CommunicationMechanism – “cim-xml” (at least). The SMI-S server could support others, but shall support “cim-xml”.
- OtherCommunicationMechanismDescription – used only if “other” is also specified for CommunicationMechanism.
- InteropSchemaNamespace – The Namespace within the SMI-S server where the CIM Interop Schema can be accessed. Each namespace provided shall contain the complete information and if multiple namespaces are provided they shall contain the same information. Even though multiple InteropSchemaNamespaces may be provided, an SMI-S client may rely on the first namespace as the definitive namespace for accessing the Interop Schema (including the class instances of the Server Profile).
- ProtocolVersion – The Version of the cim-xml protocol if this is the defined one. This shall be implemented for SMI-S server.
- FunctionalProfilesSupported: Permissible values are “Unknown”, “Other”, “Basic Read”, “Basic Write”, “Schema Manipulation”, “Instance Manipulation”, “Association Traversal”, “Query Execution”, “Qualifier Declaration”, “Indications”. This defines the CIM Operation Profiles supported by the SMI-S server. It can return multiple values.
- FunctionalProfileDescriptions – If the “other” value is used in the FunctionalProfilesSupported attribute, this shall be populated. If provided, it shall be derived from the CIM_CommunicationMechanism.FunctionalProfileDescriptions property. Use of this attribute is not specified by SMI-S.
- MultipleOperationsSupported – A boolean that defines whether the SMI-S server supports batch operations.
- AuthenticationMechanismsSupported – Permissible values are “Unknown”, “None”, “Other”, “Basic”, “Digest”. Defines the authentication mechanism supported by the SMI-S server. It can return multiple values.
- AuthenticationMechanismDescriptions – Defines other Authentication mechanism supported by the SMI-S server. The value shall be supplied if the “Other” value is set in the AuthenticationMechanismsSupported attribute. This attribute is optional. It is to be provided only when the AuthenticationMechanismsSupported attribute is “other”.
- Namespace – Namespace(s) supported on the SMI-S server. This attribute may have multiple values (one for each namespace defined in the SMI-S server) and is literal (L) because the namespace names may not be translated into other languages.
- Classinfo – The values are taken from the interop schema Namespace.classinfo property. The values represent the classinfo (CIM Schema version, etc.) for the namespaces defined in the corresponding namespace listed in the namespace attribute. Each entry in this attribute shall correspond to the namespace defined in the same position of the namespace attribute. There shall be one entry in this attribute for each entry in the namespace attribute.
- RegisteredProfilesSupported – The SMI-S profile(s) supported by the server prefixed by “SNIA” (at least). An SMI-S server may also support other RegisteredProfiles, but it shall support at least one “SNIA”. In addition, this attribute can also be used to advertise subprofiles, when subprofiles are to be advertised.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

The RegisteredProfilesSupported is an array. Each entry includes a RegisteredOrganization (i.e., SNIA), a Profile name and possibly a subprofile name. Each name is separated by a colon.

Note that a single SMI-S server can support multiple profiles. As a result, the profile attribute is an array of values. Each entry shall be separated by a comma. Each subprofile shall be scoped to its parent profile. For example, if a vendor implements the Software subprofile as part of the Array profile, the entry would be “SNIA:Array:Software”. Note that entries are not case sensitive.

Additional attributes, such as specific profile services supported, model subprofiles supported and the SMI-S release level are not discovered via SLP. They would be found by traversing the model presented by the SMI-S server.

9.6 User Agents (UA)

A User Agent is a Client process working on the user’s behalf to establish contact with some service. A User Agent retrieves service information from Service Agents (see 9.7, “Service Agents (SAs)”) or Directory Agents (see 9.8, “Directory Agents (DAs)”). Further description of a Client and its role may be found in 10.2, “SMI-S Client”.

The only required feature of a User Agent is that it can issue SrvRqsts and interpret DAAdverts, SAAdverts and SrvRply messages. If Directory Agents exist, User Agents shall issue requests as Directory Agents are discovered.

An SMI-S Client commonly acts as an SLP user agent (UA) using the query functions of SLPv2 to determine location and other attributes of the “WBEM” SLP Service Type Template defined in 9.5.2.

The basic search methodology for SMI-S clients is to search for directory agents and service agents within their scope. If all SMI-S servers are supported by a directory agent, then the search yields nothing but directory agents. The client can then obtain a list of services (and their URLs) for management of the SMI-S servers.

If any Service agents are not covered by a directory agent (i.e., are not within its scope), then the client obtains service replies from those service agents.

A client would typically search for all service types available in their scope(s). This returns a list of service types available in the network. However, an SMI-S client can be assumed to be searching for “WBEM” service types. If a client only manages selected devices (e.g., switches or arrays), the SMI-S client can issue a request for the specific services by using predicates on the “RegisteredProfilesSupported” attribute.

9.7 Service Agents (SAs)

A Service Agent supports an SMI-S server process working on behalf of one or more services to advertise the services. See clause 10, “SMI-S Roles” for further description of SMI-S servers.

Service Agents shall accept multicast service requests and unicast service requests. SAs may accept other requests (Attribute and Service Type Requests). An SA shall reply to appropriate SrvRqsts with SrvRply or SAAdvert messages. The SA shall also register with all DAs as they are discovered.

To provide for SMI-S Client discovery of SMI-S servers, a CIM Server shall act as a Service agent (SA) for the IETF Service Level Protocol (SLP) v2 as defined in IETF RFC 2608. The service shall correspond to v2 of SLP (IETF RFC 2608 and 2609) and shall use the Service Templates defined in 9.10, “‘Standard WBEM’ Service Type Templates” for advertisements. An SMI-S server acting as an SA shall provide a separate SLP advertisement for each address/port that the CIM Server advertises.

9.8 Directory Agents (DAs)

SMI-S supports existing SLPv2 Directory Agents (without modification). That is, SMI-S makes no assumptions on Directory Agents that are not made by SLPv2. Note that this cannot quite be said for User Agents, which are looking for SMI-S specific services, or Service Agents, which are advertising SMI-S specific services.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

9.9 Service Agent Server (SA Server)

9.9.1 General Information

The reserved listening port for SLP is 427, the destination port for all SLP messages. Service Agents (SAs) are required to listen for both unicast and multicast requests. A Directory Agent (DA) shall listen for unicast request and specific multicast DA discovery service requests. SAs and User Agents (UAs) that perform passive DA discovery shall listen for multicast DA Advertisements (DAAdverts).

TCP/IP requires that a single server process per network interface control all incoming messages to a port. That requirement necessitates a mechanism to share the SLP port (427).

Sharing the SLP port (427) is accomplished with a Service Agent Server (SA Server) process that 'owns' the port on behalf of all SAs, UAs and optional DA that are listening for SLP messages. The SA Server listens for incoming messages that request advertisement information and either answer each request or forward it to the appropriate SA. The SA Server also performs passive DA discovery and distribute the DA addresses and scopes to the SAs and UAs that it serves.

A SA Server may also function as a DA if the SA Server is implemented so that it answers requests for advertisement information rather than forwarding each request to the appropriate SA. The combined DA/SA Server is acting as an intermediary between a SA that registered an advertisement and a UA requesting information about the advertisement.

9.9.2 SA Server (SAS) Implementation

The RFC 2 614 document describes APIs for both the C and Java languages. Both APIs are designed for standardized access to the Service Location Protocol (SLP).

The goals of the C API are:

- directly reflect the structure of SLP messages in API calls and return types as character buffers and other simple data structures;
- simplify memory management to reduce API client requirements;
- provide API coverage of just the SLP protocol operations to reduce complexity;
- allow incremental and asynchronous access to return values, so small memory implementations are possible;
- support multithreaded library calls on platforms where thread packages are available.

The Java API goals are:

- provide complete coverage of all protocol features, including service type templates, through a programmatic interface;
- encourage modularity so that implementations can omit parts of the protocol that are not needed;
- in conformance with Java's object-oriented nature, reflect the important SLP entities as objects and make the API itself object-oriented;
- use flexible collection data types consistently in the API to simplify construction of parameters and analysis of results;
- designed for small code size to help reduce download time in networked computers.

9.9.3 SA Server (SAS) Clients

9.9.3.1 Description

An SAS Client is a Service Agent (SA), User Agent (UA) or Directory Agent (DA) that is associated with a SA Server. The SA Server listens on the SLP port (427) and appropriately handles all incoming messages for each SAS Client. A DA acting as a SAS Client is separately configured on the same host as the SA Server.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

9.9.3.2 SAS Client Requests – SA Server Responses

A SA Server responds when appropriate, to incoming unicast and multicast messages from SAS Clients. The SA Server may answer with the appropriate advertisement, if available, or forward the request on to the appropriate SAS Client. If the SA Server is also functioning as a DA, it discards a multicast SrvRqst of “service:directory-agent” that has either a missing scope list or the scope list does not contain a scope the Service Agent Server/DA is configured with.

9.9.4 SA Server configuration

9.9.4.1 Overview

SA Servers may be configured via an individual SLP configuration file, programmatically or a combination of the two. DHCP may also be used to obtain the scope list for a SA Server. Figure 76 illustrates the various ways of configuring a SA Server.

9.9.4.2 SLP Configuration File

9.9.4.2.1 If a SA Server is also functioning as a DA, the following DA configuration properties shall be set.

Table 304 - Required Configuration Properties for SA as DA

Keyword	Data Type	Value
net.slp.isDA	boolean	true
net.slp.DAAttributes	string	(SA-Server=true)

The DA attribute/value pair of “SA-Server=true” allows a query to be used when a SA Server/DA needs to be identified. In addition, when the SA Server/DA responds to a SrvRqst message with a DAAdvert message, the DA attribute/value pair is included.

9.9.4.2.2 The remaining DA configuration property, net.slp.DAHeartBeat, with a default of 10,800 s, can be set as appropriate. If a SA Server is not functioning as a DA, the following SA configuration property shall be set.

Table 305 - Required Configuration Properties for SA

Keyword	Data Type	Value
net.slp.SAAttributes	string	(SA-Server=true)

9.9.4.3 Programmatic configuration

Both the C and Java language APIs provide access to SLP properties contained in the SLP configuration file. The actual SLP configuration file is not accessed or modified via the interfaces. Once the file is loaded into memory at the start of execution, the configuration property accessors work on the in-memory representation.

The C language API provides the SLPGetProperty() and SLPSetProperty() functions. The SLPGetProperty() function allows read access to the SLP configuration properties while the SLPSetProperty() function allows modification of the configuration properties.

The SLPSetProperty() function has the following prototype:

```
void SLPSetProperty(const char *pcName, const char *pcValue).
```

The SLPSetProperty() function takes two string parameters: pcName and pcValue. The pcName parameter contains the property name and pcValue contains the property value. The following example uses the SLPSetProperty() function to configure a SA Server that is not functioning as a DA:

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
void setSAAttributes() {
char value[80]; /* A buffer for storing the attribute string. */
value = “SA Server=true”;
SLPSetProperty(“net.slp.SAAttributes”, value);
}
```

9.9.4.4 DHCP configuration

If the Service Agent Server is also functioning as a DA, its scope list may be obtained via DHCP. Scopes discovered via DHCP take precedence over the net.slp.useScopes property in the SLP configuration file.

9.9.4.5 Scope

A Service Agent Server is configured with a minimum scope of DEFAULT. If a Service Agent Server is not functioning as a DA, DEFAULT is the only scope configured. If a Service Agent Server is functioning as a DA, it may have additional scopes configured. Use of the DEFAULT scope enables the associated SAS Clients (UAs, SAs and DA) to actively discover the Service Agent Server using a well-known value for scope.

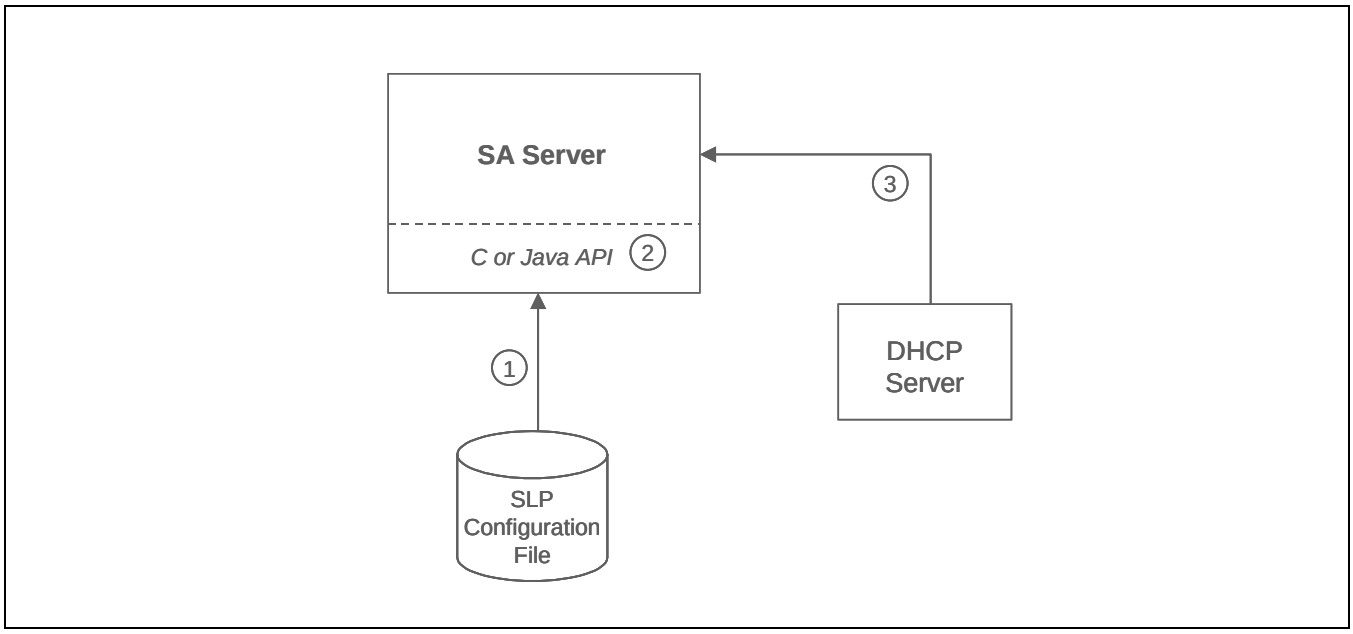


Figure 76 - SA Server Configuration

a) The SA Server may obtain specific configuration values via an individual SLP Configuration file.

b) The C or Java API provides programmatic access to the configuration file properties.

c) The SA Server may obtain its scope values from a DHCP Server.

9.9.5 SA Server discovery

“Discovery” of a SA Server by its SAS Clients is accomplished by successfully establishing the required communication link between the two entities. There is no need for active or passive discovery as described by SLP since both the SA Server and SAS Clients reside on the same host system.

9.9.6 SAS Client registration

Service Agents (SAs) that are SAS Clients register and deregister with the local SA Server using the SrvReg/ SrvDereg messages. The SA Server responds with a Service Acknowledgement (SrvAck) message. The SA Server store a service advertisement until either its lifetime expires or a SrvDereg message is received.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

If the SA Server is also functioning as a DA, the DA registration requirement is also met. The SA server also forwards any SA registration to other DAs that have the same scope as the SA.

9.10 'Standard WBEM' Service Type Templates

NOTE For each description in the template that states the value shall be the CIM_ClassName.PropertyName value, the format/rules for these values are defined in the Interop Model of the CIM Schema and in 7.3.7. This SLP Template requires a minimum Schema version of 2.7 to support the required values. Some of the optional values require CIM Schema version 2.8.

```
Name of submitter: "DMTF" <technical@dmtf.org>

Language of service template: en

Security Considerations:

Information about the specific CIM Server implementation or the
Operating System platform may be deemed a security risk in certain
environments. Therefore these attributes are optional but
recommended.

Template Text:

-------------------------template begins here-----------------------
template-type=wbem

template-version=1.0

template-description=
    This template describes the attributes used for advertising
    WBEM Servers.

template-url-syntax=string
# The template-url-syntax MUST be the CIM_ObjectManager.Name property
                           value.

service-hi-name=string O
# This string is used as a name of the CIM service for human
# interfaces. This attribute MUST be the
# CIM_ObjectManager.ElementName property value.

service-hi-description=string O
# This string is used as a description of the CIM service for
# human interfaces.This attribute MUST be the
# CIM_ObjectManager.Description property value.

service-id=string L
# This attribute is set to the same value as the serviceid
# used in the service URL registered with SLP. A User Agent can use
# this attribute to discover a WBEM Server which the User Agent
# discovered previously.
```

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
service-location-tcp=string L
# The location of one service access point offered by the WBEM Server
# over TCP transport. This attribute must provide sufficient addressing
# information so that the WBEM Server can be addressed directly using
# this attribute.
# Example: (service-location-tcp=http://localhost:5988)

CommunicationMechanism=string L
# The communication mechanism (protocol) used by the CIM Object Manager
                          for
# this service-location-tcp defined in this advertisement. This
                          information
# MUST be the
                          CIM_ObjectManagerCommunicationMechanism.Comm
                          unicationMechanism
# property value.
# CIM-XML is defined in the CIM Operations over HTTP specification which
                          can
# be found at http://dmtf.org/
“Unknown”, “Other”, “cim-xml”


OtherCommunicationMechanismDescription = String L O
# The other communication mechanism defined for the CIM Server in the
                          case
# the “Other” value is set in the CommunicationMechanism string.
# This attribute MUST be the
                          CIM_ObjectManagerCommunicationMechanism.Othe
                          rCommunicationMechanism
# property value. This attribute is optional because it is only required
                          if the
# “other” value is set in CommunicationMechanism. The value returned is
# a free-form string.

InteropSchemaNamespace=string L M
# Namespace within the target WBEM Server where the CIM Interop Schema
                          can be
# accessed. Multiple namespaces may be provided. Each namespace
                          provided
# MUST contain the same information.

ProtocolVersion=String O L
# The version of the protocol. It MUST be the
# CIM_ObjectManagerCommunicationMechanism.Version property value.

FunctionalProfilesSupported=string L M
# ProfilesSupported defines the CIM Operation profiles supported by the
# CIM Object Manager. This attribute MUST be the
# CIM_ObjectManagerCommunicationMechanism.FunctionalProfilesSupported
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
# property value.
“Unknown”, “Other”, “Basic Read”, “Basic Write”,
“Schema Manipulation”, “Instance Manipulation”,
“Association Traversal”, “Query Execution”,
“Qualifier Declaration”, “Indications”

FunctionalProfileDescriptions=string L O M
# Other profile description if the “other” value is set in the
                          ProfilesSupported
# attribute.  This attribute is optional because it is returned only if
                          the “other”
# value is set in the ProfilesSupported attribute. If provided it MUST
# be equal to the
                          CIM_ObjectManagerCommunicationMechanism.Func
                          tionalProfileDescriptions
# property value.

MultipleOperationsSupported=Boolean
# Defines whether the CIM Object Manager supports batch operations.
# This attribute MUST be the
# CIM_ObjectManagerCommunicationMechanism.MultipleOperationsSupported
# property value.

AuthenticationMechanismsSupported=String L M
# Defines the authentication mechanism supported by the CIM Object
                          Manager.
# This attributed MUST be the
#
                          CIM_ObjectManagerCommunicationMechanism.Auth
                          enticationMechanismsSupported property
                          value.
“Unknown”, “None”, “Other”, “Basic”, “Digest”

AuthenticationMechanismDescriptions=String L O M
# Defines other Authentication mechanisms supported by the CIM Object
                          Manager
# in the case where the “Other” value is set in any of the
# AuthenticationMechanismsSupported attribute values. If provided, this
                          attribute MUST be the
#
                          CIM_ObjectManagerCommunicationMechanism.Auth
                          enticationMechanismDescriptions
# property value.

Namespace=string L M O
# Namespace(s) supported on the CIM Object Manager.
# This attribute MUST be the
# CIM_Namespace.name property value for each instance of CIM_Namespace
# that exists. This attribute is optional.
# NOTE: This value is literal (L) because
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
# the namespace names MUST not be translated into other languages.

Classinfo=string M O
# This attributes is optional but if used, the values MUST be the
# CIM_Namespace.classinfo property value.
# The values represent the classinfo (CIM Schema version, etc.) for
# the namespaces defined in the corresponding namespace listed in the
# Namespace attribute. Each entry in this attribute MUST correspond
# to the namespace defined in the same position of the namespace
# attribute. There must be one entry in this attribute for each
# entry in the namespace attribute.


RegisteredProfilesSupported=string L M
# RegisteredProfilesSupported defines the Profiles that
# this WBEM Server has support for. Each entry in this
# attribute MUST be in the form of
# Organization:Profile Name{:Subprofile Name}
#
# examples:
#     DMTF:CIM Server
#     DMTF:CIM Server:Protocol Adapter
#     DMTF:CIM Server:Provider Registration
# The Organization MUST be the
# CIM_RegisteredProfile.RegisteredOrganization property value.
# The Profile Name MUST be the
# CIM_RegisteredProfile.RegisteredName property value.
# The subprofile Name MUST be the
# CIM_RegisteredProfile.RegisteredName property value when it is
# used as a Dependent in the CIM_SubProfileRequiresProfile
# association for the specified Profile Name (used as the antecedent).


--------------------------template ends here-----------------------
```

--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

10 SMI-S Roles

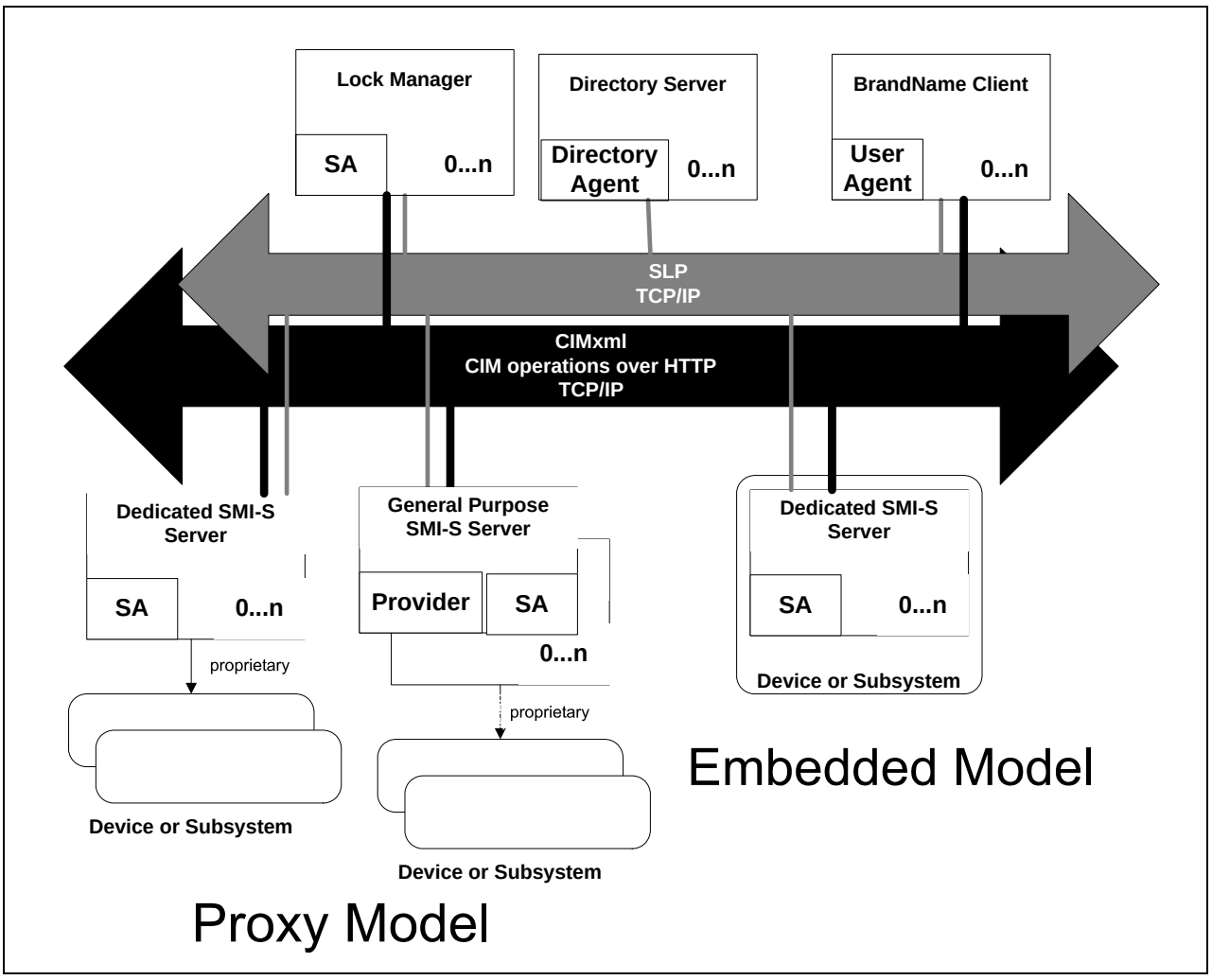


Figure 77 - Complete Reference Model

10.1 Introduction

As shown in Figure 77 above, the complete reference model shows the roles for the various entities of the management system. Any given host, network device or storage device may implement one or more of these roles.

Here we present a concise definition of each of these roles and the requirements on implementations of these roles in a management system. For each of these roles, specific functions are required to be implemented in one or more functional areas.

a) SLP Discovery Functions – the required discovery capabilities that the role performs in the overall management system.

b) Basic CIM-XML Operations – the management model operations that the role performs.

c) Security – the security requirements that the role is expected to satisfy.

d) Lock Management Operations – the locking operations that the role is expected to perform.

The detail of these responsibilities for each of the roles is described in the following subclauses.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

10.2 SMI-S Client

10.2.1 Overview

The SMI-S Client role in the overall management system is performed by software that is capable of performing management operations on the resources under management. This includes monitoring, configuration and control of the operations on the resources. Typical clients include user interface consoles, complete management frameworks and higher-level management applications and services such as policy based management systems.

There can be zero or more SMI-S clients in the overall management system. These clients can all coexist simultaneously and can perform independent or overlapping operations in the management system. It is outside the scope of this standard to specify client cooperation with other clients in any way. The semantics of the described management system is that the last successful client operation is valid and persists in the absence of any other client operations (last write wins).

It is expected that development kits for the management system will provide code for the required functions implemented in clients. Consoles, frameworks and management applications can then use this common code in order to comply with this standard. The specification of an API for this client code and specific language bindings for applications is outside the scope of this standard, but is a candidate for follow-on work.

10.2.2 SLP functions

The SMI-S Client role is required to implement SLP User Agent (UA) functionality as specified in 9.6, “User Agents (UA)”. The Client discovers all SMI-S servers within its configured scope that are required for its operations by querying for service specific attributes that match the criteria for those operations.

10.2.3 CIM-XML protocol functions

The SMI-S Client role shall implement CIM-Client functionality as specified by CIM-XML standard and is encouraged to implement CIM-Listener functionality as specified by CIM-XML standard.

10.2.4 Security considerations

The SMI-S Client role shall implement security as specified in clause 8, “Security”.

10.2.5 Lock management functions

There are no requirements for locking in this standard.

10.3 Dedicated SMI-S Server

10.3.1 Overview

The intention of the SMI-S server role in a management system is to provide device management support in the absence of any other role. A simple management system could consist of just a SMI-S Client and a SMI-S Server and all management functions can be performed on the underlying resource. This means that a vendor can offer complete management for the resource by shipping a standalone client for the resource and not depend on any other management infrastructure. Although, at the same time, the SMI-S Server can participate in a more complex management environment through the use of the standard mechanisms described here.

- Embedded SMI-S Server – the SMI-S Server functions are incorporated into the resource directly and do not involve separate installation steps to become operational.
- Proxy SMI-S Server – the SMI-S Server is hosted on a system separate from the resource and communicates with the resource via either a standard or proprietary remote protocol. This typically involves an installation operation for the SMI-S Server and configuration for, or independent discovery of, the desired resource.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

In order to minimize the footprint on the resource or proxy hosts, the required functions of the SMI-S Server role have purposely been scaled back from those of a typical general purpose CIM Server running on host with more significant resources. These required functions are described in the subclauses below.

10.3.2 SLP functions

The SMI-S Server role shall implement SLP Service Agent (SA) functionality, as specified in 9.7, “Service Agents (SAs)”. Optionally, it is encouraged to implement Service Agent Server functionality or use an existing SA Server if one exists. The SMI-S Server shall advertise service specific attributes that allow the Client to locate it based on its profile, as defined in 9.10, “‘Standard WBEM’ Service Type Templates”.

10.3.3 CIM-XML Protocol Functions

10.3.3.1 General

The SMI-S Server role shall implement CIM-Server functionality as specified by the CIM-XML standard.

10.3.3.2 Required intrinsic methods

An SMI-S Server is required to implement a set of intrinsic methods as defined for each profile. The intrinsic methods are grouped by “functional profile” as specified in the CIM-XML standard:

Table 306 - Functional Profiles

Functional Group	Dependency	Methods
Basic Read	None	GetClass EnumerateClasses EnumerateClassNames GetInstance EnumerateInstances EnumerateInstanceNames GetProperty
Basic Write	Basic Read	SetProperty
Instance Manipulation	Basic Write	CreateInstance ModifyInstance DeleteInstance
Schema Manipulation	Instance Manipulation	CreateClass ModifyClass DeleteClass
Association Traversal	Basic Read	Associators AssociatorNames References ReferenceNames
Query Execution	Basic Read	ExecQuery
Qualifier Declaration	Schema Manipulation	GetQualifier SetQualifier DeleteQualifier EnumerateQualifiers
Indication	None	

SMI-S Servers shall implement intrinsic methods as specified in 7.3.7.4, “CIM Server requirements”.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

10.3.3.3 Required model support

The SMI-S Server shall implement the Server Profile as detailed in 7.3.7, “Server Profile”

10.3.4 Security considerations

The SMI-S Server role shall implement security as specified in clause 8, “Security”.

10.3.5 Lock management functions

There are no requirements for locking in this standard.

10.4 General Purpose SMI-S Server

10.4.1 Overview

The General Purpose SMI-S Server role in an overall management system is intended to reduce the number of network connections needed by a Client to manage large numbers of resources. It is also envisioned as a convenient place to perform operations across multiple resources, further off-loading these from the Client as well.

In addition, the General Purpose SMI-S Server role can provide a hosting environment for the plug-in instrumentation of host-based resources and management proxies for resources with remote management protocols. These plug-ins are called providers and considered sub-roles of the General Purpose SMI-S Server.

A General Purpose SMI-S Server is not required in a management system, but is expected to be deployed at least as a common infrastructure for host-based resources. In any large storage network, there may be several General Purpose SMI-S Servers (as many as one per host). Communication between General Purpose SMI-S Servers may be standardized in the future, but this capability is outside the scope of this standard. General Purpose SMI-S Servers may act as a point of aggregation for multiple SMI-S Profiles as described in 10.6.2, “General Purpose SMI-S Server as a Profile Aggregator” using existing standard mechanisms as specified here.

As General Purpose SMI-S Servers are expected to be deployed on hosts with more resources and less footprint concerns than other managed resources, the required functions, specified below, are more extensive than that of an Dedicated SMI-S Server.

10.4.2 SLP functions

The General Purpose SMI-S Server role is required to implement SLP Service Agent (SA) functionality as specified in 9.7, “Service Agents (SAs)”. The General Purpose SMI-S Server shall advertise service specific attributes that allow the Client to locate it based on the profiles it supports, as defined in 9.9, “Service Agent Server (SA Server)”.

10.4.3 CIM-XML Protocol functions

10.4.3.1 General

The General Purpose SMI-S Server role shall implement CIM-Server functionality as specified by the CIM-XML standard.

10.4.3.2 Required intrinsic methods

The General Purpose SMI-S Server is required to implement the minimum profile as specified in CIM-XML standard. In addition, it shall implement the intrinsic methods needed to support the Profiles that it supports.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

10.4.3.3 Required model support

The General Purpose SMI-S Server shall implement the Server Profile as detailed in the Server Profile (see 7.3.7, “Server Profile”).

10.4.3.4 Security considerations

The General Purpose SMI-S Server role shall implement security as specified in clause 8, “Security”.

10.4.4 Lock management functions

There are no requirements for locking in this standard.

10.4.5 Provider Subrole

10.4.5.1 Overview

A sub-role within a General Purpose SMI-S Server that can be used to provide management support for the resource, especially useful when the resource is host-based (i.e. HBA or Host Software) and the platform provides an CIM Server as part of its operating system.

10.4.5.2 Required model support

The Provider shall implement the Provider Subprofile as detailed in the object model shown in the Server Profile (see 7.3.7, “Server Profile”).

10.5 Directory Server

10.5.1 General

The Directory Server role is used to facilitate Discovery of instances of the various roles in a management system, but may also be used by management systems to store common configurations, user credentials and management policies. Functions outside of Discovery are outside the scope of this standard. The Directory Server role is optional for a compliant management system.

10.5.2 SLP functions

The Directory Server role is required to implement SLP Directory Agent (DA) functionality as specified in 9.8, “Directory Agents (DAs)”. The Directory registers all Agents and Object Managers within its configured scope and allows queries for their respective service specific attributes.

10.5.3 CIM-XML Protocol functions

There are no additional CIM-XML requirements for this role.

10.5.4 Security considerations

There are no additional security requirements for this role.

10.5.5 Lock management functions

There are no requirements for locking in this standard.

10.6 Combined Roles on a Single System

10.6.1 Overview

As mentioned previously, the various roles of the management system can be deployed in different combinations to different systems throughout the managed environment. In general, there are no restrictions on what roles can be deployed on any given system, but some examples are given below to illustrate typical situations.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

10.6.2 General Purpose SMI-S Server as a Profile Aggregator

10.6.2.1 SLP functions

The General Purpose SMI-S Server role may implement SLP User Agent (UA) functionality as specified in 9.6, “User Agents (UA)”. The General Purpose SMI-S Server discovers all Profiles within its configured scope that are aggregated by querying for service specific attributes that match the criteria for those aggregations.

10.6.2.2 CIM-XML Protocol functions

The General Purpose SMI-S Server role may implement CIM-Client functionality as specified in CIM-XML standard and may implement CIM-Listener functionality as specified in CIM-XML standard. A General Purpose SMI-S Server may reflect instances and classes from the aggregated Profiles (perhaps by delegating operations to the Dedicated SMI-S Servers), but is not required to do so. The Profile’s Model instances will be reflected in the advertised default namespace of the General Purpose SMI-S Server. The hierarchy of General Purpose SMI-S Servers and Dedicated SMI-S Servers in a multi-level system needs to be reflected in the model in such a way that it can be administrated.

10.6.2.3 Security considerations

There are no requirements for security for this role.

10.6.2.4 Lock management functions

There are no requirements for locking in this standard.

--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

11 Installation and upgrade

11.1 Introduction

The interoperability of the management communications in a storage network gives customers a choice of vendors of management solutions, but it can also introduce ease-of-use problems when these different vendors deploy Clients, Agents and Object Managers. In order to supply a complete management solution, many management vendors provide not only management client and object managers, but also other pieces of the management infrastructure (e.g., Lock Managers, Directory Servers, Object Managers, Databases, Messaging Servers, Application Servers and even Providers and Agents). Problems are possible when multiple vendors install/remove these infrastructure components in the same environment and conflicts arise. One of the goals of creating management interoperability is to reduce the time and expense end-users need for the management of their SANs. Thus, the management of constituents in a SMI-S environment should be easy to install, easy to upgrade and easy to reconfigure. Mature products using SMI-S technology should experience seamless and nearly management free installation, upgrade and reconfiguration.

This clause deals with the issues of SMI-S configuration management and recommends some steps that vendors should take to minimize the problem, leading to better customer satisfaction with the overall management solution.

11.2 Role of the Administrator

Ultimately, a vendor's installation software cannot make perfect decisions when conflicts arise, and there may be valid reasons why a customer has deployed software of similar function from multiple vendors. In the situation where two software components are both installed that perform the same shared function, and only one can reasonably operate without conflicts, it is up to the administrator to solve these conflicts and remove or disable the redundant infrastructure component(s).

Installation software can, however, make a best effort to detect any conflicts and notify the administrator of possible conflicts during its installation and initialization. A vendor's installation software commonly allows the administrator to install and uninstall the various infrastructure components on an individual basis should a conflict arise. The implications of this are that vendors are motivated to support interoperation with other vendor's components. The advantage to the vendor is that a customer is more likely to install a component that demonstrates the best interoperability with other components.

11.3 Goals

11.3.1 Non-Disruptive Installation and De-installation

Clients, Agents, Proxy Agents, Lock Managers and Directory Servers shall be capable of being installed and de-installed without disrupting the operation of other constituents in a SMI-S management environment. An Object Manager independent of its providers shall be capable of being installed or de-installed from a SMI-S management environment without disrupting operations. As SANs are often deployed in mission critical environments the up-time of the solution is critical and thus, the uptime of the management backbone as a key component of the solution is equally critical. Additionally, the installation and de-installation of SMI-S interface constituents will not compromise the availability of mission critical applications, if at all possible.

11.3.2 Plug-and-Play

The ultimate goal of management interoperability is zero administration of the management system itself. A customer should be able to install new storage hardware and software and have the new component become part of the management system automatically. The use of discovery and default configuration parameters throughout this standard is intended to assist in achieving this goal.

During the reconfiguration of the management system, the schema that Clients see should remain consistent (Schema forward compatibility is ensured via CIM standard).

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

11.4 Installing device support

11.4.1 General

Manufacturers of storage hardware and software typically install their product and the accompanying management support as an system. SMI-S software falls into one of the following categories.

- Embedded Agent – the hardware device has an embedded SMI-S agent as an integrated component. No other installation of software is needed to enable management of the device.
- Proxy Agent – the hardware or software comes with an Agent that is installed on a host. The Proxy Agent needs to connect to the device and obtain unique identifying information.
- Provider - the hardware or software comes with a Provider that is installed into an Object Manager. The provider provides the management for one or more product instances and needs to either discover those instances or be explicitly configured to communicate with the device.

Conflicts are possible for Proxy Agents and Providers if multiple vendors attempt to install support for the same device. Also, when a device vendor needs to upgrade the Provider or Proxy Agent for the device, the installation software needs to determine all of the locations of the previous installations to insure there are not duplicate management paths to the device and thus, insure reliable on-going operation of the device.

11.4.2 Installation

Installation software for devices needs to be able to find existing object managers that may control the device in order to offer an administrator a choice in management constituents for the device. In addition, the installation software may desire to find existing agents/providers that provide device support in order to reliably upgrade that support. For these reasons, an installation software program may want to act as a SMI-S Client during installation. This allows it to make the automated decisions that eliminate the need for an administrator to manually configure or adjust certain aspects of the management system.

The provider registration schema shows what device support is already installed and installation software will need to consult this schema before installing new software. If the installation software is upgrading device support from one schema to another (for example from a proxy agent to a provider, or a provider to an embedded agent) the installation software needs to uninstall or disable the previous software support elements.

During installation, the installation software, acting as a Lock Aware Client may detect that some agents are Lock Unaware and needs to deal with (warn administrator) that both Lock Aware and Lock Unaware Agents/ Object Managers could be the cause of inconsistent state in their network.

11.4.3 Discovery and initialization of Device support

According to the SMI-S Reference Model, vendors of Host Software, Devices and Subsystems that are managed via a Proxy Agent or are managed through an Object Manager (with providers) are expected to provide a means for establishing a reliable connection between the Host Software, Device or Subsystem and the ProxyAgent or Object Manager. As such, a special Client with administration/installation capability (as supplied by the vendor) is required to supply the IP address of a device/subsystem with related authentication credentials to a Proxy Agent or Object Manager designated to manage the device. This administrative Client may obtain the IP address of the device/subsystem via automated means (for example by probing through an in-band HBA or looking at the object model that the HBA agent already provides), or via manual means (for

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

example by requiring a system manager to manually input the IP address of the device/subsystem from documentation supplied by the vendor). Figure 78, illustrates this requirement.

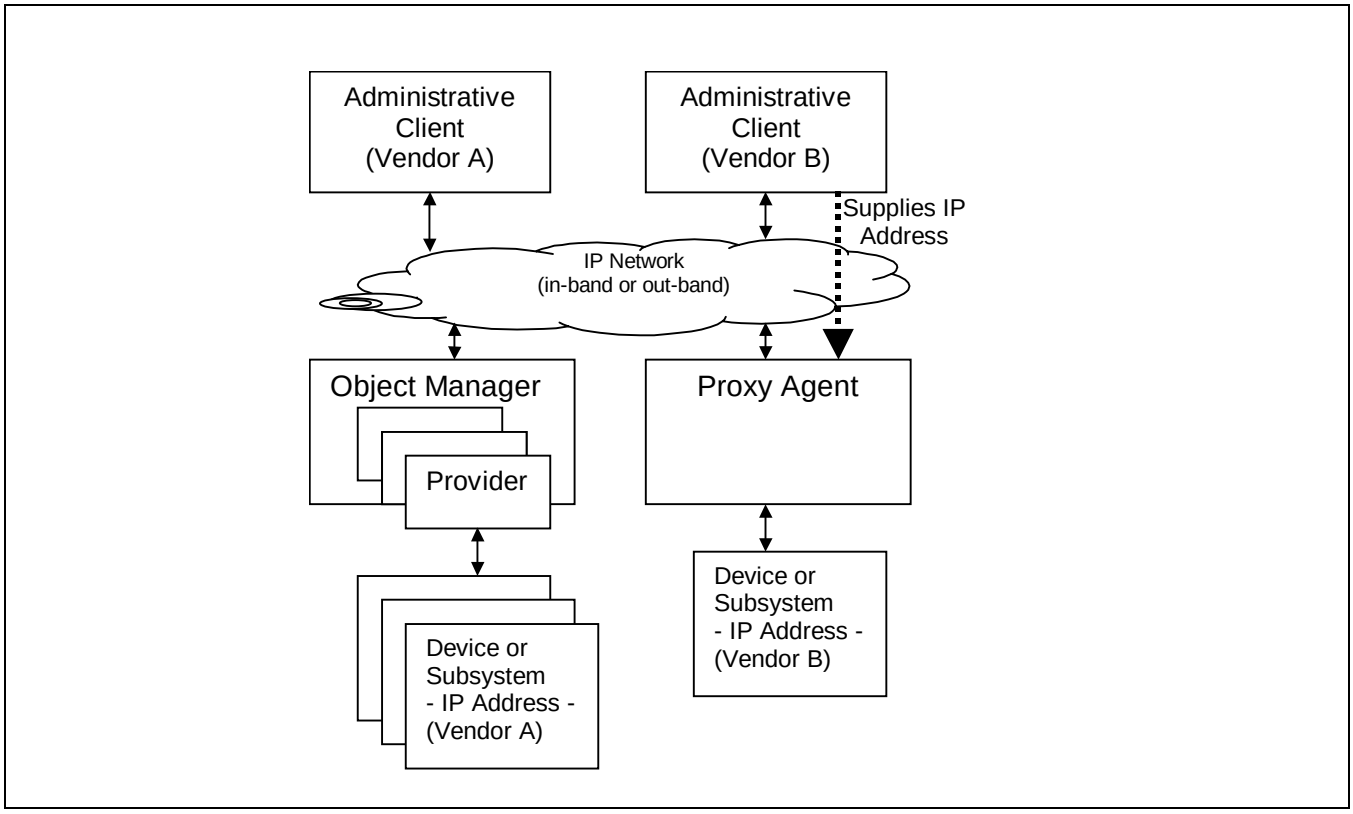


Figure 78 - Configuration Administration

11.4.4 Removal/Update

During the removal of a device support software (agent, provider, object manager), the installation/removal software (if available) should automatically detect existing device support software in order to shutdown and remove these in a consistent manner. This detection process needs to be aware that Clients may be actively using this support device and that thus, the device may need to be disabled for new management operations and administrated through an orderly shut-down procedure prior to de-installation. The implementation of shutdown procedures for components and any shutdown order dependency is outside the scope of this standard, but may need to be considered by implementers.

During the update of device support software, installation software should automatically detect any existing device support software in order to successfully complete the upgrade. This device support may exist on multiple hosts. If the update includes installing a new provider, the installation software needs to use the provider installation/upgrade method that is supported by the existing Object Manager and needs to be coded accordingly (see 11.4.5, “Reconfiguration”).

When a software update involves a major schema version upgrade (e.g., 2.x to 3.x), the installation software needs to be aware of the effect of the schema upgrade on existing clients. For example, it may choose to simultaneously support both versions for some period of time.

11.4.5 Reconfiguration

When device support update requires an update of a provider, the device support installation software should configure the new provider with the same subscriptions that exist in the old provider before removing the old

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

provider. This can be done via the instances of the subscriptions in the agent or object manager that currently exist.

11.4.6 Failure

Agents can become unavailable for several reasons. This includes powering off the device and transient network failures. If a device's model becomes unavailable, it is recommended that Clients do not immediately remove that device from its visualization. If the device model shows up somewhere else, the old visualization should be updated to remove the previous occurrence. Also, the client can keep track of how long the device was down for purposes of availability management, etc. Clients may have to restore indication subscriptions when the device or its proxy becomes available again. In the case of a Proxy or Embedded agent, the agent (or its host) may go down, or the network to it could fail, but the device may still be available and that needs to be factored in to any availability management. In the case of a provider, the provider to device communication channel may also fail, but the device may still be available for access.

11.5 Object manager

11.5.1 Installation

Customers are increasingly sensitive to the size of the memory footprint for management software. The goal is to minimize the impact on hosts that are not dedicated to running management software by making appropriate choices during installation and giving the administrator control over these issues. It is recommended that vendors take advantage of existing object managers if one exists, by installing a provider for device support. If an object manager does not exist, or the device support does not work with the existing object manager (due to interface requirements, for example), it is recommended that the vendor supply a Proxy Agent that is lightweight for device support. Another option is to offer to install an object manager that the vendor has provider support for, allowing other vendors to further leverage that installation.

In band providers have a connection issue where zoning may alter the management path to the device from a provider or proxy agent. In this case, the device support may need to be installed on multiple hosts in the network and the vendor needs to provide some way to coordinate which provider or proxy agent is responsible for a particular device.

Vendors are encouraged to install their providers in a unique namespace for isolation and qualification reasons. The installer then discovers (possibly via an SLP UA) the existing namespaces and insure that the one created for the new device is truly unique.

Installation of a management appliance still needs to be able to turn off built-in providers.

Lock-Aware Clients need to deal with agents and Object Mangers that may or may not themselves be Lock Aware. A failure to recognize this condition and provide appropriate administrative notification could be the cause of an inconsistent state of the network.

11.5.2 Multiple CIMOMs on a Single Server

At installation and setup, a newly-installed CIMOM searches for an open TCP port based on a defined list of well-known TCP port numbers. Also, a user interface is provided by the CIMOM installation utility that allows an administrative user to manually set the TCP port number in a persistent fashion. Both mechanisms shall co-exist to facilitate automated installs as well as manually configured installs.

To support discovery, the SLP Service Agent (SA) associated with a CIMOM that has just been installed and started up registers its TCP port number along with all the other necessary discovery information about the CIMOM. This applies to both automated port selection as well as manually configured installs. Clients, working through their SLP User Agent (UA), then use this information to establish contact with the CIMOM.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

11.5.3 Removal, reconfiguration and upgrade

An Object Manager may be upgraded while keeping the same Providers as before. Depending on the Object Manager, the Providers may have to be reinstalled and reconfigured following such an upgrade. In this case, an administrator may need to re-run the device support installation software and it should be able to restore the previous configuration if possible.

11.5.4 Failure

Temporary failure of an object manager (for example, host powered off) can result in bad installation decisions for installation software. In this case, it is advisable that the installation software provide for manual input of additional components of the management system that the installation software needs to be aware of.

11.6 Client

11.6.1 Removal

When Client software is removed, the removal software should go in and remove any subscriptions for that client that exist in any agent or object manager. In addition, it should release any locks that are held in order to clean up the lock state as well.

11.6.2 Reconfiguration

Client software can include a Listener that is configured to listen on a specific port. When this port is reconfigured, the client should redirect any Indication Handlers in existing agent and object managers as a result.

11.6.3 Failure

If possible, Clients should release locks before shutting down or upon unexpected failure.

11.7 Directory Server

11.7.1 Installation

The installation of more than one directory server in a management system does not impose a significant burden for management clients and adds to the overall availability. Vendors should recommend to administrators of their products that one or more directory servers should be deployed in the management system. Customers may have already done this for network or system management reasons.

11.7.2 Removal/Failure

SLP Clients already handle failure and removal of DAs as according to this standard (see clause 9, “Service Discovery”).

11.8 Management domains

11.8.1 General

The set of agents, object managers and a lock manager that are configured with the same LMGroup value may be considered a management domain for the purposes of administration. The use of SLP scope is independent of the use of LMGroups for these purposes. Clients should not depend on any relationship between LMGroups and SLP Scopes.

11.8.2 Initial configuration

Vendors should recommend that administrators of their products use the same LMGroup value for all agents and object managers in the same storage network (this might include multiple fabrics). Vendors should also recommend to administrators of their products that all agents and object managers be on the same IP subnet

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

or on connected subnets where the intervening router is configured to allow multicast packets between the subnets (this is to allow SLP discovery messages to flow to the entire management system).

11.8.3 Reconfiguration

Vendors of lock managers are encouraged to consider producing software that can easily reconfigure (merge or split) a lock management domain to ease the burden of this task. Splitting or merging an LMGroup should involve bringing the old LockManager(s) down, reconfiguring the agents and object managers with their new LMGroup value.

11.9 Lock manager

This standard does not include support for a lock manager.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Annex A
(informative)

Futures

A.1 Overview

The following clauses outline some of the possible expansions to the SMI standard that are being considered by the development team. They are provided here for informational purposes only. There is no assurance that any of these items will ever be refined into a future standard.

A.2 HBA LUN masking and persistent binding

This section should be refined and expanded to assure that the standard is properly integrated with other standards for LUN masking and persistent binding (i.e., HBA API).

A.3 Managed Hub Section

The current SMI-S standard doesn't address managed hubs as a possible SAN component. If they continue to be of interest, the standard will need to be expanded to address any concerns particular to managed hubs.

A.4 IP Storage

The current SMI-S standard does not address IP storage. As part of the standard's refinement and completion, it will need to be expanded to appropriately address IP security and authentication.

A.5 Multi-Path Modelling

The current standard does not include support for multipathing within its array profile. The profile and its models should be expanded to provide appropriate support and infrastructure.

A.6 Provider Modelling

Provider Modelling is emerging DMTF work that needs to be monitored.

A.7 Non-Fibre Fabrics

In future versions of this standard, it is intended that the fabric model and durable names for ports will be extended to cover other types of connectivity, such as InfiniBand, IP networks, etc.

Although the current fabric model is specific to Fibre Channel, a best-efforts approach was taken in defining it to allow for future extensions to include additional types of connectivity.

When the fabric model is extended for a new type of connectivity, it will be necessary to define durable names for the ports on the new type of connectivity. For instance, the durable names for ports on an IP network might be MAC addresses. It is expected that durable names for ports will be connectivity specific, and they may be different for each different type of connectivity.

A.8 Compliance notification

A method is needed to allow a provider to inform clients that it complies to SMI-S's indications' profiles. Emerging work from the DMTF Interop Workgroup relates to this problem. Rather than offering a competing approach, SMI-S will re-evaluate this work in the near future. The Interop work is described in 7.2.2.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

A.9 Cascaded Agents

Support could be provided for a multi-level agent hierarchy, in which a given agent could provide a management interface to higher-level devices while simultaneously relying on the management interface provided by agents “below” it.

A.10 Network Storage

SMI-S should be extended to encompass the more general storage networks and incorporate a broad range of network storage technologies (e.g., NAS, iSCSI), rather than focusing exclusively on SANs.

A.11 Synchronization of file system elements through copy services

The copy services are designed to support synchronization of file system elements. The specifics of this support will be addressed in a future release of the standard.

A.12 Model size distinctions in Disk Drive

The standard should be expanded to include both the large (or detailed) model that is included currently, and the small model that is allowed by CIM.

A.13 Expanded extent mapping

A future revision of this standard should support recursive extent mapping as an additional profile.

A.14 Locking

Appropriate coordination between increasingly complex SAN fabrics including, for example, cascaded agents, will require the development of a locking model to assure data integrity.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

A.15 Policy management

A.15.1 Policy enhancements to functionality ladder

The introduction of Policy specifications to the SMI-S is intended to enable standardized policy based storage management in an SMI-S managed environment. To this end, the Policy additions to SMI-S are intended to enable the following capability in SMI-S.

a) Provide a mechanism for a persistent repository of policies expressed in a standard interchange format that can be read and acted on by the appropriate policy mechanisms in the environment.

b) Allow SMI-S Policy clients to enter and edit storage management policy information in the persistent repository;

c) Allow a policy manager to get notification when policies that it processes are added or changed.

d) Enable SMI-S agents by defining events that trigger policy actions.

A.15.2 Policy objectives

The specific objectives of the policy specifications for SMI-S are:

- define a mechanism that allows separation of Policy specification and Policy execution;
- define a mechanism that supports distributed policy management:
- define a mechanism that allows common interchange of Policy specification information;
- define a mechanism that standardizes “triggers” for policy based management and augments SMI-S agents as needed to surface triggering events;
- identify methods that SMI-S Agents need in order to support a standard set of Policy Actions;
- provide a mechanism that scales to enterprise environments;
- enable one or more Policy Managers to analyze, invoke and maintain Policies in the Policy repository (server).

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

A.15.3 Policy enhancements to reference model

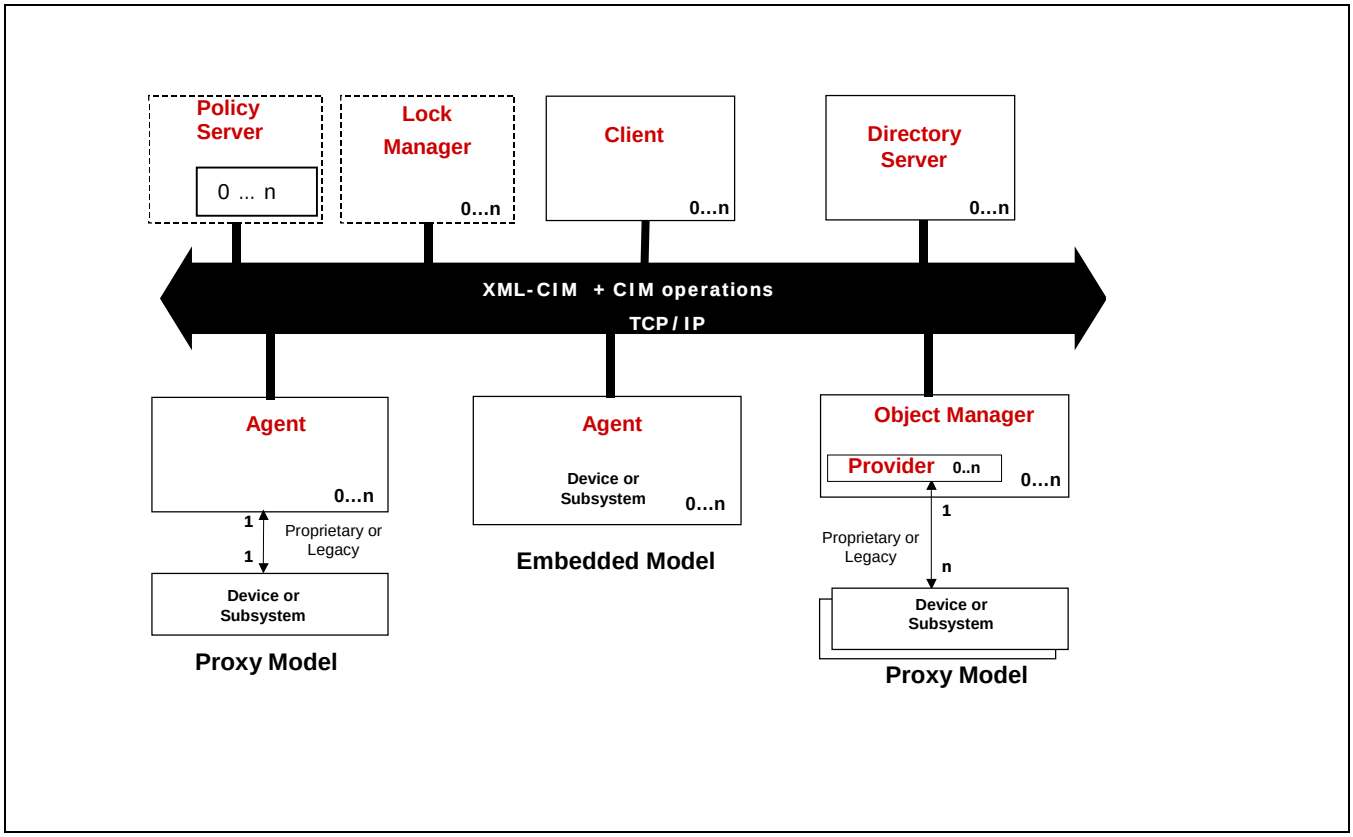


Figure A.1 - Reference Model with Policy Server

Enabling SMI-S for policy based management requires the addition of a “Policy Server” to the reference model for SMI-S. Like other servers in the reference model, there can be many policy servers (or none). The role of the Policy Server is to hold policy information.

A.15.4 Policy components

The design for policy management includes the following components:

Policy Client: A Policy Client is a CIM-XML application that enters and edits policy information in a Policy Server.

NOTE A Policy client may or may not have a policy manager function. That is, it is not required that the Policy client and the Policy Manager be the same application.

Policy Server: A Policy Server is a repository of Policy information. It does not take any action on any information that is entered into the database of policy information. However, it does surface indications to policy managers that are listening to policies that are assigned to them.

Policy Manager: A Policy Manager is a component that acts on a particular set of actions in policies that are defined in the Policy Server. Policy management may be distributed.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

These components are illustrated in Figure A.2.

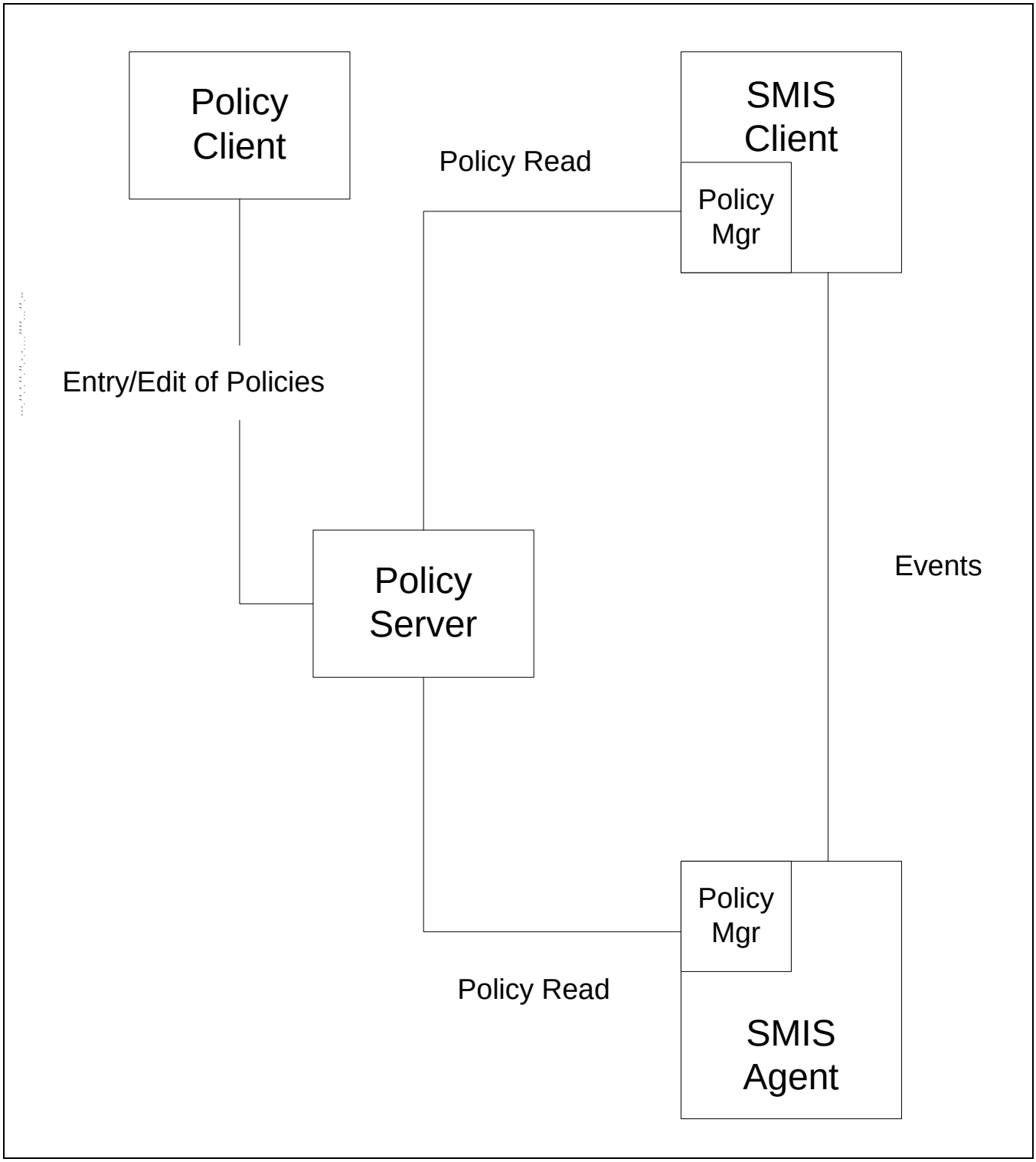


Figure A.2 - Policy Components

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Annex B
(informative)

Experimental Profiles

B.1 Overview

Some of the profiles that were considered for inclusion in SMI-S version 1.0.0 failed to receive sufficient testing to satisfy the publication requirements set forth by the SMI committee within the SNIA. They are presented here as an aid to implementers who are interested in likely future developments within the SMI standard. There is a high likelihood that they will be included in an revision of the standard. The contents of an experimental subprofile is likely to change as implementation experience is gained.

B.2 Common Profiles and Subprofiles

B.2.1 Sparing Subprofile

B.2.1.1 Description

NOTE This subprofile is experimental and is provided for information only. The contents of an experimental subprofile is likely to change as implementation experience is gained. Please provide any implementation feedback to Storage Management Initiative Technical Steering Group (td@snia.org).

7.3.3.5 focuses on the mapping of storage to Volumes. The subprofile specified in this clause enhances that picture by modelling spares.

A spare disk is modelled with the ActsAsSpare association to a SparedSet – which has aggregation associations to other disks.

This SparedSet object associates one or more spares with a group of active disks. The members of this group are implementation and configuration dependent. In some cases, this group might be the disks in a enclosure, disks in a RAID group, or disks of a particular make and model. The cardinality on IsSpare allows multiple spares per group and also allows a spare to participate in multiple groups.

Figure B.1 shows an instance diagram for three disks (StorageExtent only shown) with two disks participating in the SparedSet and one disk acting as the spare. The full model for the disk is shown in 7.3.3.4, “Disk Drive subprofile”.

B.2.1.2 Standard dependencies

This profile uses classes from the 2.8 Preliminary CIM schema.

B.2.1.3 Profile dependencies

The sparing subprofile introduces no Profile dependencies.

B.2.1.4 CIM Server requirements

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

B.2.1.5 Instance diagrams

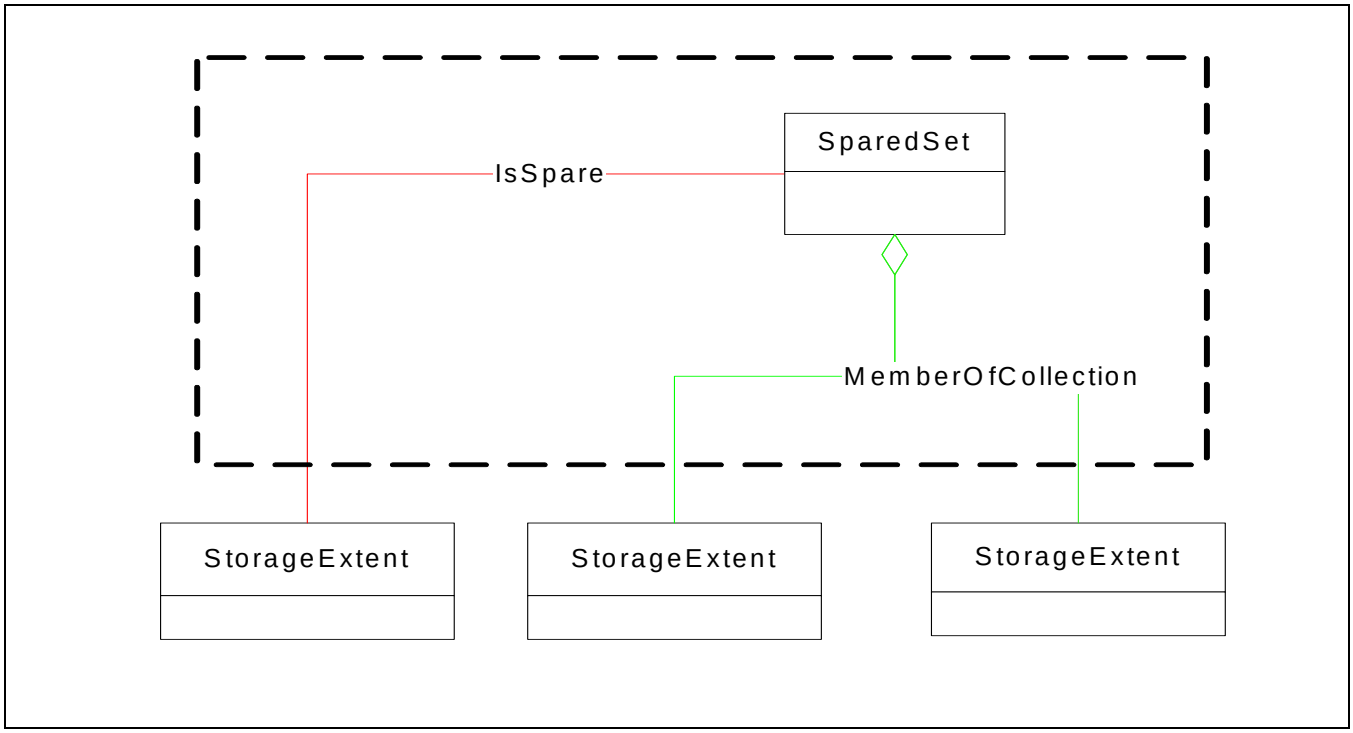


Figure B.1 - Sparing Instance

B.2.1.6 Durable names and correlatable IDs

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

B.2.1.7 Methods

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

B.2.1.8 Client considerations

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

B.2.1.9 Recipes

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

B.2.1.10 Instrumentation requirements

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

B.2.1.11 Required CIM Elements

Table B.1 - Required CIM Elements

Profile Classes & Associations	Notes
IsSpare (B.2.1.12.1)	
MemberOfCollection (B.2.1.12.2)	
SparedSet (B.2.1.12.3)	
Packages	
None	
Associated indications	
None	

B.2.1.12 Required properties for CIM Elements

B.2.1.12.1 IsSpare

Table B.2 - Required Properties for IsSpare

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Antecedent	ref	override	A ManagedElement or Collection of elements acting as a spare.
Dependent	ref	override	The set of elements that ARE spared.
HotStandby	boolean		HotStandby is a boolean indicating that the 'spare' is operating as a hot standby.

B.2.1.12.2 MemberOfCollection

Table B.3 - Required Properties of MemberOfCollection

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Collection	ref	key	The Collection that aggregates members.
Member	ref	key	The aggregated member of the Collection.

B.2.1.12.3 SparedSet

Table B.4 - Required Properties for SparedSet

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
RedundancyStatus	uint16		.

--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table B.4 - Required Properties for SparedSet (Continued)

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
FailoverSupported	uint16		The type of failover algorithms that are supported.

B.2.1.13 Optional Subprofiles

Table B.5 - Optional Profiles or Subprofiles

Name	Notes
None	

This subprofile should be used with either the disk drive or extent mapping subprofiles.

B.3 SML Subprofiles

B.3.1 InterLibraryPort Connection Subprofile

B.3.1.1 Description

NOTE This subprofile is experimental and is provided for information only. The contents of an experimental subprofile is likely to change as implementation experience is gained. Please provide any implementation feedback to Storage Management Initiative Technical Steering Group (td@snia.org).

Support of InterLibraryPort devices, also known as pass-through ports or cartridge exchange mechanisms, is designated as optional in this profile. However, when such a device exists the agent representing the library should instantiate this class for each port. When one or more libraries are connected via an Inter-Library Port and the corresponding agents are working with separate name spaces a mechanism is required for correlating the LibraryExchange association that represents the port connection.

B.3.1.2 Standard dependencies

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

B.3.1.3 Profile dependencies

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

B.3.1.4 CIM Server requirements

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

B.3.1.5 Instance diagrams

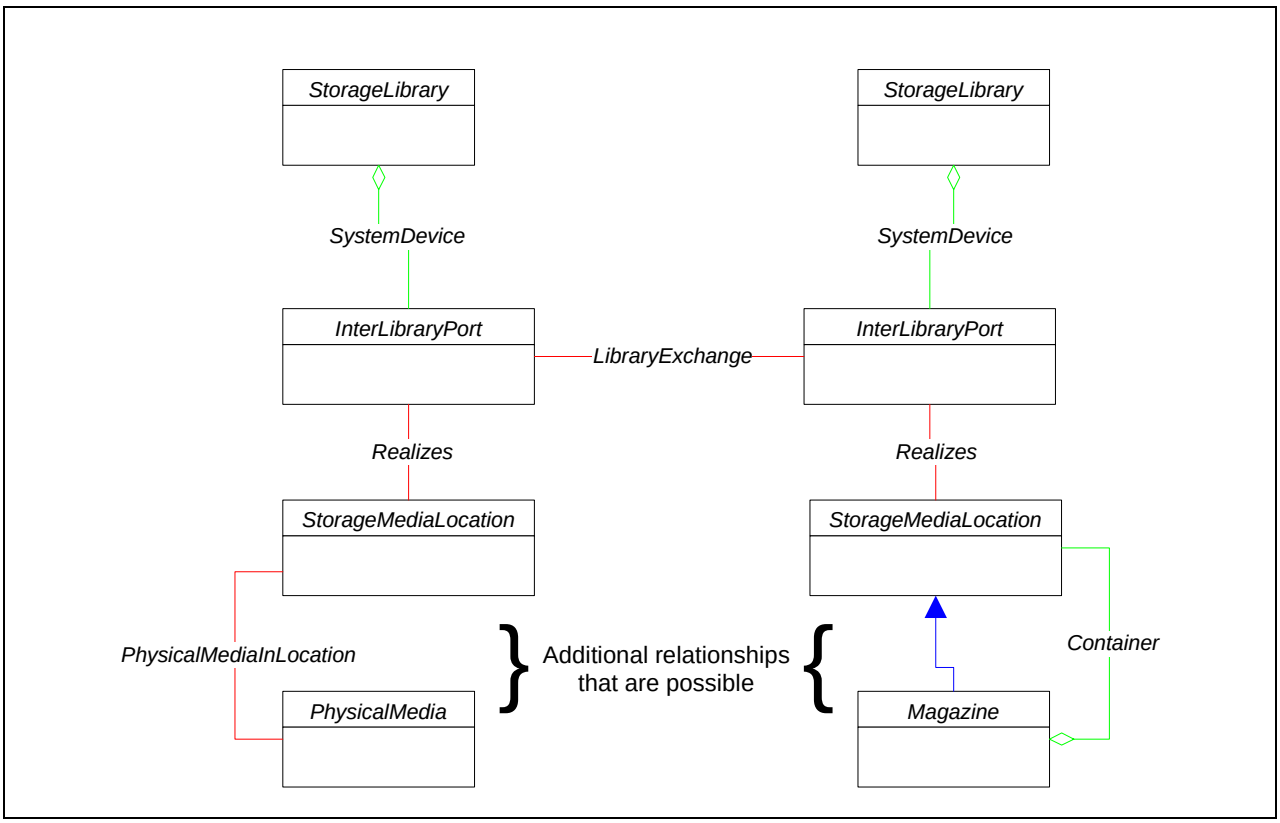


Figure B.2 - InterLibraryPort Connection Instance Diagram

B.3.1.6 Durable names and correlatable IDs

A Durable Name is not defined by this profile for InterLibraryPort instances and remains unspecified. This is not an issue when associated InterLibraryPort instances are within the same name space.

B.3.1.7 Methods

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

B.3.1.8 Client considerations

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

B.3.1.9 Recipes

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

B.3.1.10 Instrumentation requirements

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

B.3.1.11 Required CIM Elements

Table B.6 - Required CIM Elements

Profile Classes & Associations	Notes
InterLibraryPort (B.3.1.12.1)	Representing a connecting port between two libraries.
LibraryExchange (B.3.1.12.2)	
Packages	
None	
Associated Indications	
Creation/Deletion of InterLibraryPorts	SELECT * FROM CIM_InstCreate WHERE SourceInstance ISA CIM_InterLibraryPort SELECT * FROM CIM_InstDelete WHERE SourceInstance ISA CIM_InterLibraryPort.
Changes in OperationalStatus of InterLibraryPorts	SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_InterLibraryPort AND PreviousInstance.OperationalStatus <> SourceInstance.OperationalStatus.

B.3.1.12 Required properties for CIM Elements

B.3.1.12.1 InterLibraryPort

Table B.7 - Required Properties for InterLibraryPort

Property/ Method	Type	Qualifier/ Parameter	Notes
LastAccessed	datetime		
ImportCount	uint64	counter	
ExportCount	uint64	counter	
Direction	uint16	valuemap	"Unknown", "Import", "Export", "Both Import and Export".

B.3.1.12.2 LibraryExchange

Table B.8 - Required Properties for LibraryExchange

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Antecedent	ref	key	InterLibraryPort Reference.
Dependent	ref	key	InterLibraryPort Reference.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

B.3.1.12.3 Optional Subprofiles

Table B.9 - Optional Profiles or Subprofiles

Name	Notes
None	

B.3.2 Partitioned/Virtual Library Subprofile

B.3.2.1 Description

NOTE This subprofile is experimental and is provided for information only. The contents of an experimental subprofile is likely to change as implementation experience is gained. Please provide any implementation feedback to Storage Management Initiative Technical Steering Group (td@snia.org).

Many libraries allow “partitioning”: the splitting up of library resources into pools used by different clients or hosts. Partitioning may also involve “virtualization”, used here to mean the representation of a single physical ChangerDevice as multiple logical ChangerDevices that can each be accessed or controlled independently. Each "virtual" ChangerDevice accesses its own group of StorageMediaLocations. No methods for *configuration* of partitioning, virtualization or access control are provided in this profile. Instead, a simple model is given to allow multiple (virtual) ChangerDevices to exist within a single StorageLibrary, where each ChangerDevice can access a specific subset of pre-existing StorageMediaLocations within that StorageLibrary.

B.3.2.2 Standard dependencies

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

B.3.2.3 Profile dependencies

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

B.3.2.4 CIM Server requirements

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

B.3.2.5 Instance diagrams

In this example, three “virtual” ChangerDevices within a single StorageLibrary have orthogonal access to three sets of Magazines or StorageMediaLocations, all contained within the Chassis.

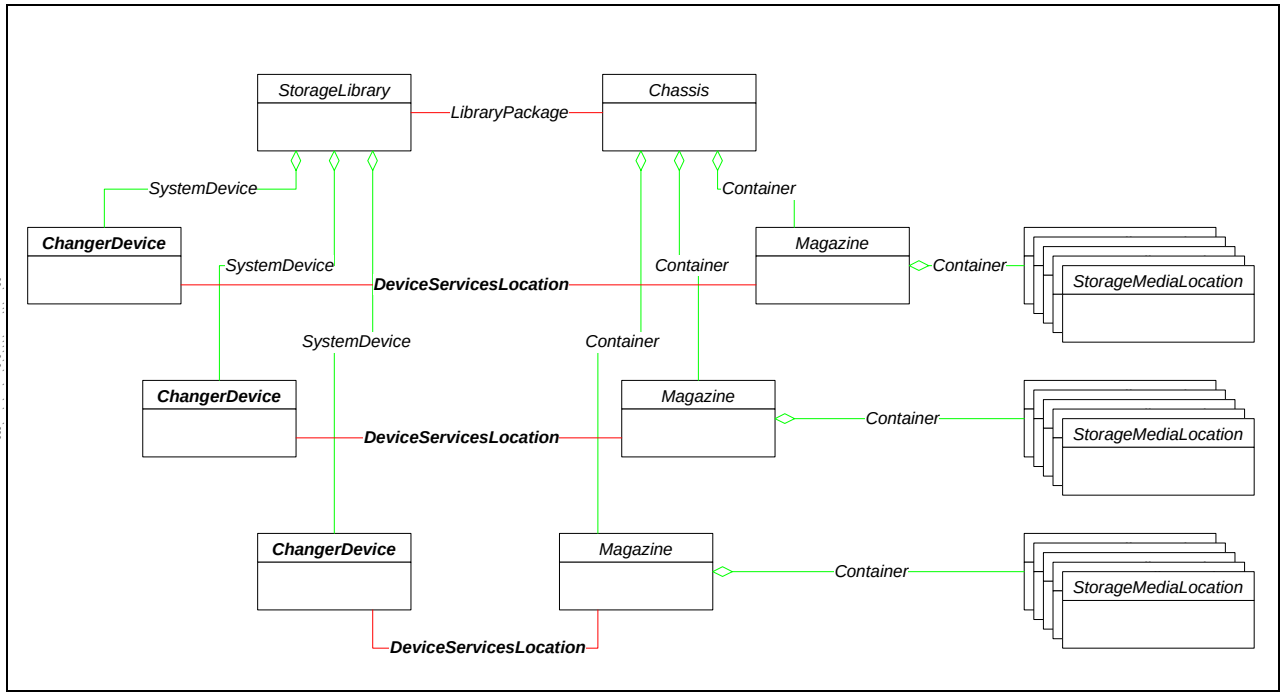


Figure B.3 - Virtual ChangerDevices

B.3.2.6 Durable names and correlatable IDs

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

B.3.2.7 Methods

None.

B.3.2.8 Client considerations

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

B.3.2.9 Recipes

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

B.3.2.10 Instrumentation requirements

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

B.3.2.11 Required CIM Elements

Table B.10 - Required CIM Elements

Profile Classes & Associations	Notes
DeviceServicesLocation (B.3.4.12.1)	
Packages	
None	
Associated Indications	
None	

B.3.2.12 Required properties for CIM Elements

B.3.2.12.1 DeviceServicesLocation

Table B.11 - Required Properties for DeviceServicesLocation

Property/Method	Type	Qualifier/ Parameter	Description/Notes
Antecedent	ref	key	The ChangerDevice.
Dependent	ref	key	The StorageMediaLocation or Magazine.

B.3.2.13 Optional Subprofiles

Table B.12 - Optional Profiles or Subprofiles

Name	Notes
None	

B.3.3 Fibre Channel connection Subprofile

B.3.3.1 Description

NOTE This subprofile is experimental and is provided for information only. The contents of an experimental subprofile is likely to change as implementation experience is gained. Please provide any implementation feedback to Storage Management Initiative Technical Steering Group (td@snia.org).

Many libraries are fibre channel connected. This means that their ChangerDevices and MediaAccessDevices either have “natice” fibre channel interfaces, or are connected to a fibre channel SAN through an interface controller (also known as a bridge or router). To represetnt this in the simplest possible manner, a single class/association pair make up this subprofile. Referring to the instance diagram below, an FCPort is added for every Controller (or SCSIProtocolController) related to a ChangerDevice or MediaAccessDevice. This FCPort is connected to the Controller through ProtocolControllerForPort. Both SCSI target parameters and fibre channel address parameters are captured by these classes.

B.3.3.2 Standard dependencies

See the parent profile.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

B.3.3.3 Profile dependencies

See the parent profile.

B.3.3.4 CIM Server requirements

B.3.3.4.1 Functional Profiles

See the parent profile.

B.3.3.4.2 Extrinsic methods

None.

B.3.3.4.3 Discovery

See the parent profile.

B.3.3.5 Instance diagrams

Note that the FCPort shall not be associated with StorageLibrary directly. This allows the future possibility of integration with the Router profile, focused on modelling internal or external router systems, in which FCPorts whould be associated with a seperate ComputerSystem instance.

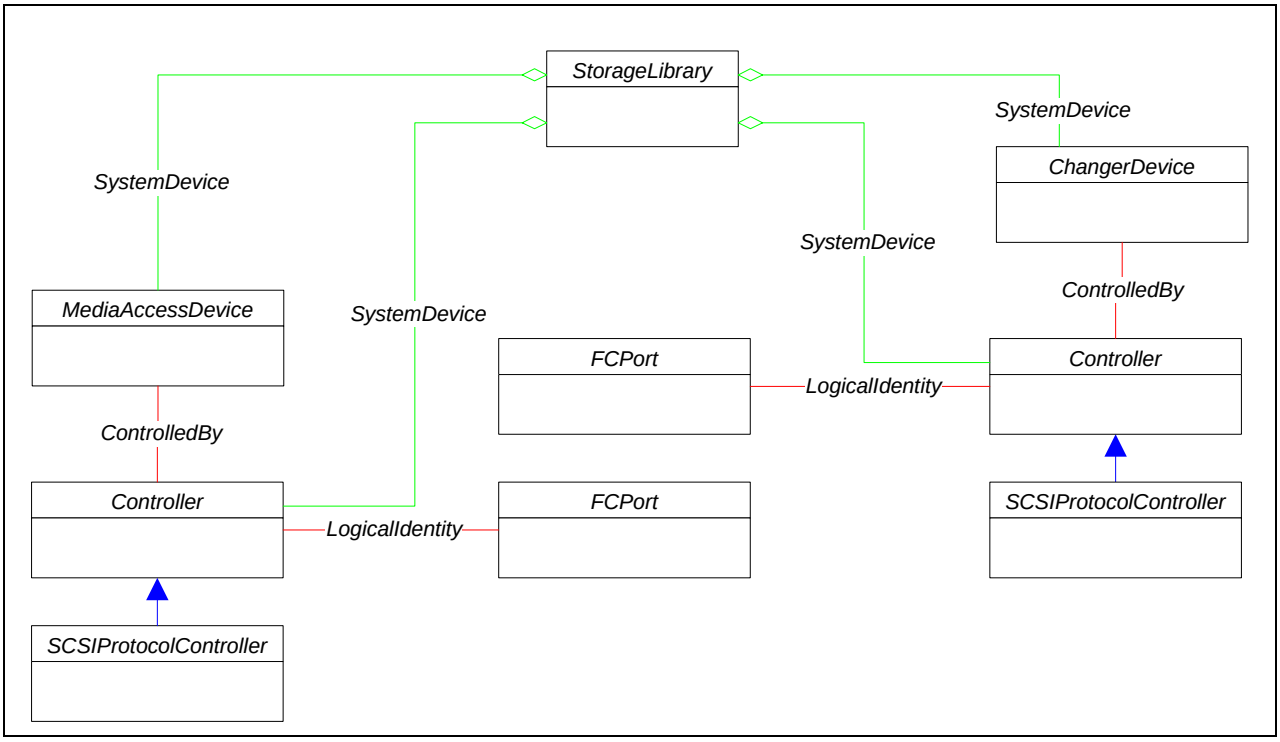


Figure B.4 - Instance Diagram for Fibre Channel Connection

B.3.3.6 Durable names and correlatable IDs of the Profile

B.3.3.6.1 Durable names exported

None.

B.3.3.6.2 Correlatable IDs used

FCPort: FCPort.PermanentAddress = Fibre Channel Port World-Wide Name. NameFormat should be set to “WWN”.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

B.3.3.7 Methods

None.

B.3.3.8 Client considerations

See parent profile.

B.3.3.9 Recipes

None.

B.3.3.10 Instrumentation requirements

None.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

B.3.3.11 Required CIM Elements

Table B.13 - Required CIM Elements

Profile Classes & Associations	Notes
FCPort (B.3.3.12.1)	The fibre channel port connected to a LogicalDevice's (SCSI) Controller.
ProtocolControllerForPort (B.3.3.12.2)	The connection that identifies the mapping between an FCPort and (SCSI) Controller.
Packages	
None.	
Associated indications	
None.	

B.3.3.12 Required properties for CIM Elements

B.3.3.12.1 FCPort

Table B.14 - Required Properties for FCPort

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
ElementName	**string**		Port Symbolic Name if available. Otherwise NULL.
OperationalStatus	uint16		See Table 132.
DeviceID	string	key, maxlen (64)	Opaque.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table B.14 - Required Properties for FCPort (Continued)

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
PortType	uint16	override	FC-GS Port.Type The specific mode currently enabled for the Port. The values: \"N\" = Node Port, \"NL\" = Node Port supporting FC arbitrated loop, \"E\" = Expansion Port connecting fabric elements (for example, FC switches), \"F\" = Fabric (element) Port, \"FL\" = Fabric (element) Port supporting FC arbitrated loop and \"B\" = Bridge Port. PortTypes are defined in the 14 165-414. When set to 1 (\"Other\"), the related property OtherPortType contains a string description of the port's type. PortType is defined to force consistent naming of the 'type' property in subclasses and to guarantee unique enum values for all instances of NetworkPort. A range of values, DMTF_Reserved, has been defined that allows subclasses to override and define their specific port types. Vendor Reserved = 16 000..65 535 can be used if the PortType is not one already defined in the above enumerations and a vendor subclass is defined specifying the appropriate “value” and “valuemap”.
OtherNetworkPortType	string		Describes the type of module, when PortType is set to 1 ("Other").
LinkTechnology	uint16		For FibreChannel, “FC”.
OtherLinkTechnology	string		A string value describing LinkTechnology when it is set to 1, "Other".
PermanentAddress	string	maxlen (64)	For FibreChannel, it is the Fibre Channel Port WWN.
NetworkAddresses[ ]	string	maxlen (64), arraytype ("indexed")	For Fibre Channel end device ports, it is the Fibre Channel ID. For Switches, it should be Null.
ActiveCOS	uint16[ ]		FC-GS Class Of Service An array of integers indicating the Classes of Service that are active. Not applicable for switches (e.g. NULL).
ActiveFC4Types	uint16[ ]		FC-GS FC4-TYPE An array of integers indicating the Fibre Channel FC-4 protocols currently running. Not applicable for switches (e.g. NULL).

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

B.3.3.12.2 ProtocolControllerForPort

Table B.15 - Required Properties for ProtocolControllerForPort

Property/ Method	Type	Qualifier/ Parameter	Description/ Notes
SystemElement	ref	key	The FCPort.
SameElement	ref	key	The Controller.

B.3.3.13 Optional Subprofile

None.

B.3.4 Partitioned/Virtual Library Subprofile

B.3.4.1 Description

NOTE This subprofile is experimental and is provided for information only. The contents of an experimental subprofile is likely to change as implementation experience is gained. Please provide any implementation feedback to Storage Management Initiative Technical Steering Group (td@snia.org).

Many libraries allow “partitioning”: the splitting up of library resources into pool used by different clients or hosts. Partitioning may also involve “virtualization”, used here to mean the representation of a single physical ChangerDevice as multiple logical ChangerDevices that can each be accessed or controlled independently. Each “virtual” ChangerDevice access its own group of StorageMediaLocations. No methods for the configuration of partitioning, virtualization or access control are provided in this subprofile. Instead, a simple model is given to allow multiple (virtual) ChangerDevices to exist within a single StorageLibrary, where each ChangerDevice can access a specific subset of pre-existing StorageMediaLocations within that StorageLibrary.

B.3.4.2 Standard dependencies

See parent profile.

B.3.4.3 Profile dependencies

See parent profile.

B.3.4.4 CIM Server requirements

B.3.4.4.1 Functional Profiles

See parent profile.

B.3.4.4.2 Extrinsic methods

None.

B.3.4.4.3 Discovery

See parent profile.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

B.3.4.5 Instance diagrams

In this example, three “virtual” ChangerDevices within a single StorageLibrary have orthogonal access to three sets of Magazines or StorageMediaLocations, all contined within the Chassis.

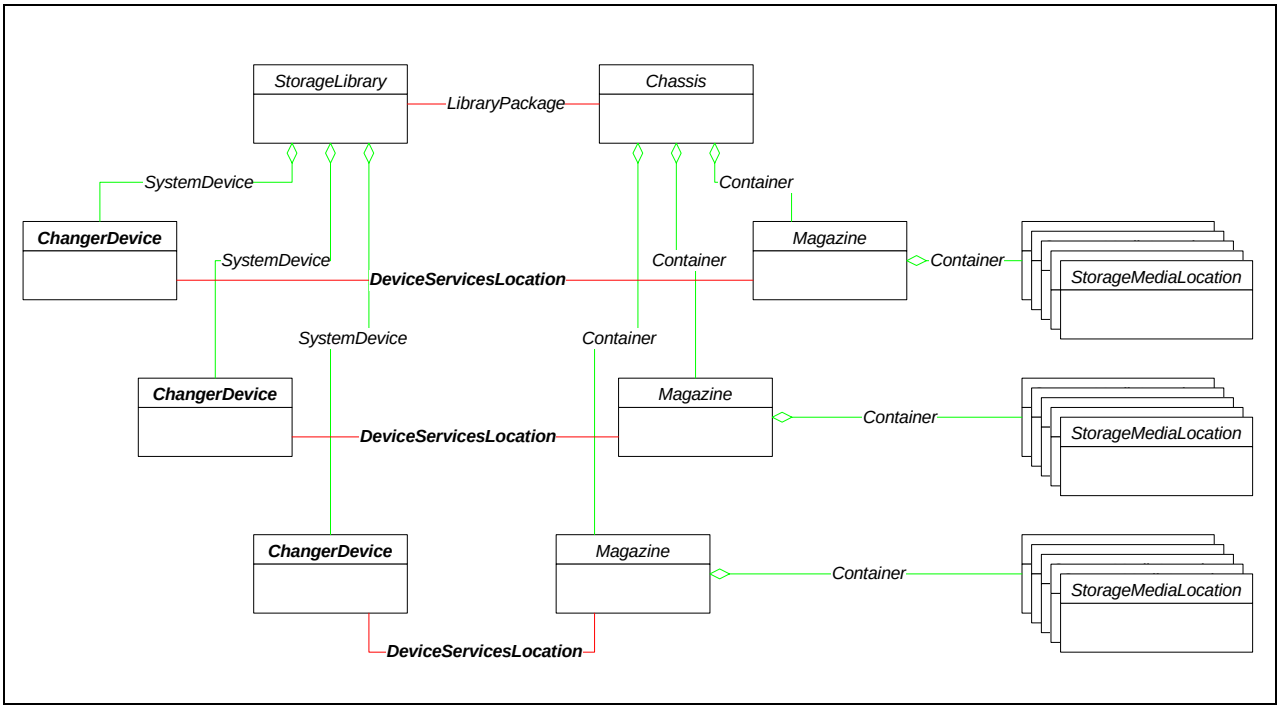


Figure B.5 - Virtual ChangerDevices Sharing a Chassis

B.3.4.6 Durable names and correlatable IDs of the Profile

B.3.4.6.1 Durable names exported

See parent profile.

B.3.4.6.2 Correlatable IDs used

See parent profile.

B.3.4.7 Methods

None.

B.3.4.8 Client considerations

See parent profile.

B.3.4.9 Recipes

See parent profile.

B.3.4.10 Instrumentation requirements

See parent profile.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

B.3.4.11 Required CIM Elements

Table B.16 - Required CIM Elements

Profile Classes & Associations	Notes
ChangerDevice (7.3.6.3.12.1)	
DeviceServicesLocation (B.3.4.12.1)	Used to specify which changers can access which sets of StorageMediaLocations or Magazines.
Packages	
None.	
Associated Indications	
None.	

B.3.4.12 Required properties for CIM Elements

B.3.4.12.1 DeviceServicesLocation

Table B.17 - Required Properties for DeviceServicesLocation

Property/ Method	Type	Qualifier/ Parameter	Description/ Notes
Antecedent	ref	key	The ChangerDevice.
Dependent	ref	key	The StorageMediaLocation or Magazine.

B.3.4.13 Optional Subprofiles

Table B.18 - Optional Profiles or Subprofiles

Name	Notes
None	

B.3.5 Library Capacity Subprofile

B.3.5.1 Description

By adding two classes (ConfigurationCapacity and ElementCapacity) servers can publish the minimum and maximum number of slots, drives, magazines, media changers and other elements associated with a given StorageLibrary.

B.3.5.2 Standard dependencies

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

B.3.5.3 Profile dependencies

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

B.3.5.4 CIM Server requirements

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

B.3.5.5 Instance diagrams

The following instance diagram illustrates the use of ConfigurationCapacity and ElementCapacity in conjunction with the basic storage library profile.

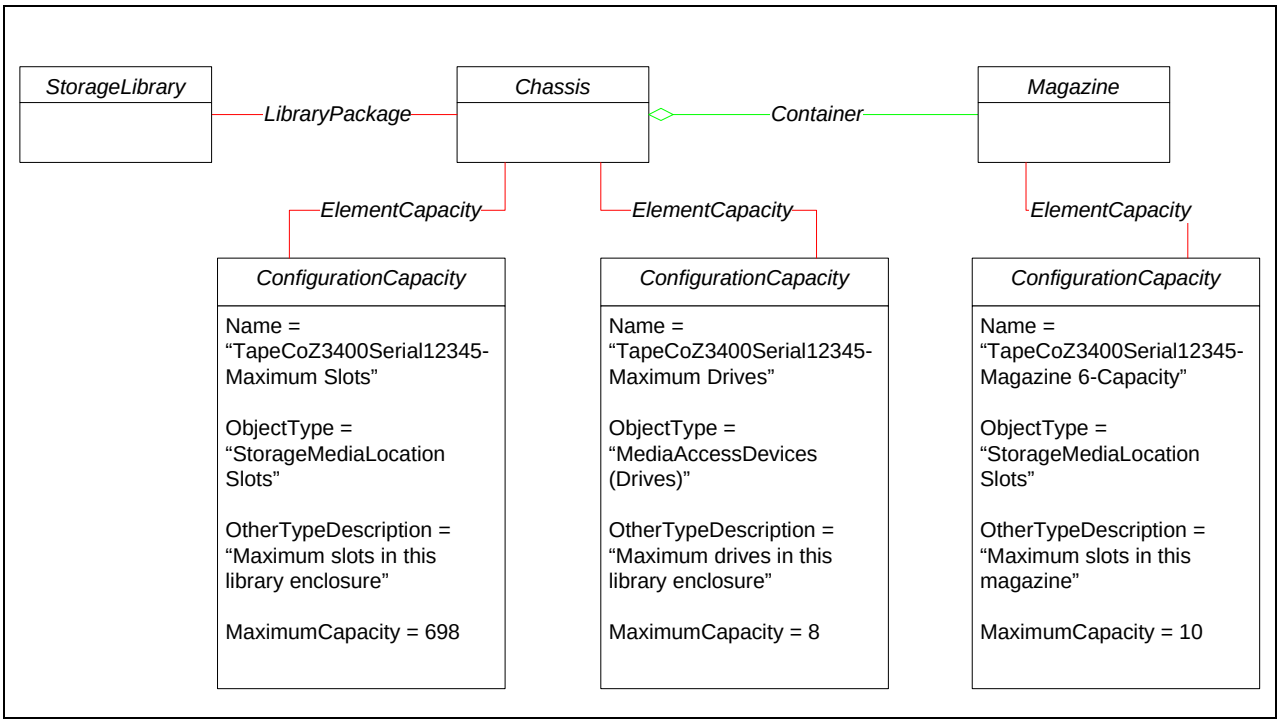


Figure B.6 - Library Capacity Instance Diagram

B.3.5.6 Durable names and correlatable IDs

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

B.3.5.7 Methods

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

B.3.5.8 Client considerations

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

B.3.5.9 Recipes

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

B.3.5.10 Instrumentation requirements

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

B.3.5.11 Required CIM Elements

Table B.19 - Required CIM Elements

Profile Classes & Associations	Notes
ConfigurationCapacity (B.3.5.12.1)	
ElementCapacity (B.3.5.12.2)	
Packages	
None	
Associated indications	
None	

B.3.5.12 Required properties for CIM Elements

B.3.5.12.1 ConfigurationCapacity

Table B.20 - Required Properties for ConfigurationCapacity

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Name	string	key, override	
ObjectType	uint16	key	See CIM schema for values.
OtherTypeDescription	string	maxlen (64)	
MinimumCapacity	uint64		
MaximumCapacity	uint64		

B.3.5.12.2 ElementCapacity

Table B.21 - Required Properties for ElementCapacity

Property/ Method	Type	Qualifier or Parameter	Description/Notes
Capacity	ref	key	PhysicalCapacity Reference
Element	ref	key	PhysicalElement Reference

B.3.5.13 Optional Subprofiles

Table B.22 - Optional Profiles or Subprofiles

Name	Notes
None	

B.3.6 “LibraryAlert” Events/Indications for Library Devices

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

B.3.6.1 Description

Historically, media libraries have been managed using both SCSI and SNMP interfaces. A number of library management standards have been defined based on these interfaces, including the “TapeAlert” error events flags. These events alert subscribing clients to current or pending error conditions related to a library, drives or media. The SCSI implementation of TapeAlert is described in the SCSI Stream Commands (SSC-2) and SCSI Media Changer Commands (SMC-2).

In order to carry these useful asynchronous events into the WBEM/CIM domain, the TapeAlert events have been mapped into instances of the CIM_AlertIndication class. This CIM class provides a general means for communicating asynchronous events to subscribing clients and TapeAlert events/indications, hereafter referred to more generally as “LibraryAlert” indications, should be specified by filling in standard values for the properties of a CIM_AlertIndication.

B.3.6.2 Standard dependencies

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

B.3.6.3 Profile dependencies

See 7.2.5, “Events – CIM indications” for a general discussion of events and indications in a SMI-S environment.

B.3.6.4 CIM Server requirements

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

B.3.6.5 Instance diagrams

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

B.3.6.6 Durable names and correlatable IDs

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

B.3.6.7 Methods

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

B.3.6.8 Client considerations

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

B.3.6.9 Recipes

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

B.3.6.10 Instrumentation requirements

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

B.3.6.11 Required properties for CIM Elements

B.3.6.11.1 AlertIndication

Table B.23 - Required Properties for AlertIndication

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Description	string	key, override	
AlertType	uint16	key	See CIM schema for values.
ProbableCause	string	maxlen (64)	
Trending	uint64		
SystemCreationClassName	string		
OtherSeverity			
EventID			
ProviderName			
SystemName			
AlertingManagedElement			
OtherAlertType			
PerceivedSeverity			
ProbableCauseDescription			
ProbableCauseDescription			

For all LibraryAlert indications, the following properties of CIM_AlertIndication shall be static and set to the values shown in Table B.24.

Table B.24 - LibraryAlert Property Settings

Property Name	Property type	Property Value
Description	string	“LibraryAlert Indication”
AlertType	uint16 (enumeration)	5 = “Device Alert”
ProabableCause	uint16 (enumeration)	1 = “other”
Trending	uint16 (enumeration)	1 = “Not Applicable”
SystemCreationClassName	string	“CIM_StorageLibrary”

Clients may identify a received CIM_AlertIndication as a LibraryAlert indication primarily by the value of “LibraryAlert Indication” in the Description property. The following Query attribute on an IndicationFilter instance should be provided by the agent for these alerts:

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
SELECT * FROM CIM_Alert
                                WHERE Description="LibraryAlert Indication"
```

The following CIM_AlertIndication properties for LibraryAlert indications shall be vendor-specific and no specification or restriction of values is made here.

Table B.25 - Vendor Specific Properties of LibraryAlert

Property Name	Property type	Property Value
OtherSeverity	string	specified by vendor.
EventID	string	specified by vendor.
ProviderName	string	specified by vendor.

A small number of CIM_AlertIndication properties for LibraryAlert indications shall have variable values that are restricted within a small range, as follows.

Table B.26 - Variable Alert Properties for LibraryAlert

Property Name	Property type	Property Value
SystemName	string	Name property value for the CIM_StorageLibrary instance that is associated with this unique indication.
AlertingManagedElement	string	CIMInstance in string format for element to which this indication applies: MediaAccessDevice, StorageLibrary or PhysicalMedia.

The remaining CIM_AlertIndication properties for LibraryAlert indications shall have values derived from the SCSI TapeAlert specifications: SCSI Stream Commands (SSC-2) and SCSI Media Changer Commands (SMC-2).

Note that a small number of indications apply only to Tape libraries, while all other indications apply generically to any library type. Those indications that are tape-specific may be identified by the following strings in the OtherAlertType property.

Table B.27 - SCSI TapeAlert-based Properties

Property Name	Property type	Property Value
OtherAlertType	string	"Tape snapped/cut in the drive where media can be de-mounted."
OtherAlertType	string	"Tape snapped/cut in the drive where media cannot be de-mounted."
OtherAlertType	string	"The drive is having severe trouble reading or writing, which will be resolved by a retension cycle."

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

The remaining CIM_AlertIndication properties and values for all LibraryAlert indications are show below. Note that the OtherAlertType property, in particular, serves to uniquely identify each of the LibraryAlert indications.

Table B.28 - LibraryAlert AlertIndication Properties

Event/Alert Summary	CIM_AlertIndication "Mapped" Properties from SSC-2 and SMC-2 Specifications			
	OtherAlert Type	Perceived Severity	ProbableCause Description	Recommended Action[ ]
	string	uint16	string	string
Read Warning	"The drive is having severe trouble reading."	"3" = "Degraded/ Warning"	"The drive is having problems reading data. No data has been lost, but there has been a reduction in the performance."	""
Write Warning	"The drive is having severe trouble writing."	"4" = "Warning"	"Worn out Media".	"1. Discard the worn out media" "2. Use a new cleaning media" .
Hard Error	"The drive had a hard read or write error."	"5" = "Warning"	"Bad Media or Drive. The operation has stopped because an error has occurred while reading or writing data that the drive cannot correct."	""
Media	"Media can no longer be written/read or performance is severely degraded."	"6" = "Critical"	"Bad Media".	"1. Copy any data you require from this media." "2. Do not use this media again." "3. Restart the operation with a different media."
Read Failure	"The drive can no longer read data from the storage media."	"6" = "Critical"	"Worn out media".	"1. Replace media." "2. Call the drive supplier help line."

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table B.28 - LibraryAlert AlertIndication Properties (Continued)

Event/Alert Summary	CIM_AlertIndication "Mapped" Properties from SSC-2 and SMC-2 Specifications			
	OtherAlert Type	Perceived Severity	ProbableCause Description	Recommended Action[ ]
	string	uint16	string	string
Write Failure	"The drive can no longer write data to the media."	"6" = "Critical"	"The media is from a faulty batch or the drive is faulty: "	"1. Use known-good media to test the drive. " "2. If the problem persists, call the media drive supplier".
Media Life	"The media has exceeded its specified life."	"3" = "Degraded/ Warning"	"The media has reached the end of its calculated useful life: "	"1. Copy any data you need to another media." 2. Discard the old media."
Not Data Grade	"The cartridge is not data-grade. Any data you write to the media is at risk. Replace the cartridge with a data-grade media."	"3" = "Degraded/ Warning"	"The cartridge is not data-grade. Any data you write to the media is at risk."	"1. Replace the cartridge with a data-grade media."
Write Protect	"Write command is attempted to a write protected media."	"6" = "Critical"	"Replace with writable media".	"You are trying to write to a write protected cartridge. Remove the write protection or use another media."
No Removal	"Manual or software unload attempted when prevent media removal is on."	"2" = "Information"	"Wait until drive is not in-use"	"You cannot eject the cartridge because the drive is in use. Wait until the operation is complete before ejecting the cartridge."
Cleaning Media	"Cleaning media loaded into drive"	"2" = "Information"	"The media in the drive is a cleaning cartridge."	"1. Replace this media with writeable media".

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table B.28 - LibraryAlert AlertIndication Properties (Continued)

Event/Alert Summary	CIM_AlertIndication "Mapped" Properties from SSC-2 and SMC-2 Specifications			
	OtherAlert Type	Perceived Severity	ProbableCause Description	Recommended Action[ ]
	string	uint16	string	string
Unsupported Format	"Attempted load of unsupported media format (e.g., DDS2 in DDS1 drive)."	"2" = "Information"	"You have tried to load a cartridge of a type that is not supported by this drive."	"1. Insert media of a type supported by this drive".
Recoverable Snapped Tape	"Tape snapped/cut in the drive where media can be de-mounted."	"6" = "Critical"	"The operation has failed because the tape in the drive has snapped."	"1. Discard the old tape." "2. Restart the operation with a different tape."
Unrecoverable Snapped Tape	"Tape snapped/cut in the drive where media cannot be de-mounted."	"6" = "Critical"	"The operation has failed because the tape in the drive has snapped."	"1. Do not attempt to extract the tape cartridge." "2. Call the tape drive supplier help line."
Memory Chip In Cartridge Failure	"Memory chip failed in cartridge."	"3" = "Degraded/ Warning"	"The memory in the media has failed, which reduces performance. "	"Do not use the cartridge for further write operations."
Forced Eject	"Manual or forced eject while drive actively writing or reading."	"6" = "Critical"	"The operation has failed because the media was manually de-mounted while the drive was actively writing or reading."	""
Read Only Format	"Media loaded that is read-only format."	"3" = "Degraded/ Warning"	"You have loaded a cartridge of a type that is read-only in this drive. The cartridge will appear as write protected."	""
Directory Corrupted On Load	"Drive powered down while loaded or permanent error prevented the directory being updated."	"3" = "Degraded/ Warning"	"The directory on the cartridge has been corrupted. File search performance will be degraded. "	"The directory can be rebuilt by reading all the data on the cartridge."

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table B.28 - LibraryAlert AlertIndication Properties (Continued)

Event/Alert Summary	CIM_AlertIndication "Mapped" Properties from SSC-2 and SMC-2 Specifications			
	OtherAlert Type	Perceived Severity	ProbableCause Description	Recommended Action[ ]
	string	uint16	string	string
Nearing Media Life	"Media may have exceeded its specified number of passes."	"2" = "Information"	"The storage media is nearing the end of its calculated life."	"1. Use another storage media for your next backup. "2. Store this storage media in a safe place in case you need to restore data from it."
Clean Now	"The drive thinks it has a head clog or needs cleaning."	"6" = "Critical"	"The drive needs cleaning."	"1. If the operation has stopped, eject the storage media and clean the drive." "2. If the operation has not stopped, wait for it to finish and then clean the drive. Check the drive user's manual for device specific cleaning."
Clean Periodic	"The drive is ready for a periodic cleaning."	"3" = "Degraded/ Warning"	"The drive is due for routine cleaning."	"1. Wait for the current operation to finish." "2. Then use a cleaning cartridge. Check the drive user's manual for device specific cleaning instructions."

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table B.28 - LibraryAlert AlertIndication Properties (Continued)

Event/Alert Summary	CIM_AlertIndication "Mapped" Properties from SSC-2 and SMC-2 Specifications			
	OtherAlert Type	Perceived Severity	ProbableCause Description	Recommended Action[ ]
	string	uint16	string	string
Expired Cleaning Media	"The cleaning media has expired."	"6" = "Critical"	"The last cleaning cartridge used in the drive has worn out."	"1. Discard the worn out cleaning cartridge." "2. Wait for the current operation to finish." "3. Then use a new cleaning cartridge."
Invalid Cleaning Media	"Invalid cleaning media type used."	"6" = "Critical"	"The last cleaning cartridge used in the drive was an invalid type."	"1. Do not use this cleaning cartridge in this drive." "2. Wait for the current operation to finish." "3. Then use a valid cleaning cartridge."
Retension Requested	"The drive is having severe trouble reading or writing, which will be resolved by a retension cycle."	"3" = "Information"	"The drive has requested a retension operation."	""
Dual-Port Interface Error	"Failure of one interface port in a dual-port configuration (i.e., Fibre Channel)".	"3" = "Degraded/ Warning"	"A redundant interface port on the drive has failed."	""
Cooling Fan Failure	"Fan failure inside drive mechanism or drive enclosure."	"3" = "Degraded/ Warning"	"A drive cooling fan has failed."	"Replace cooling fan or drive enclosure".
Power Supply Failure	"Redundant power supply unit failure inside the drive enclosure or rack subsystem."	"3" = "Degraded/ Warning"	"A redundant power supply has failed inside the drive enclosure."	"Check the enclosure user's manual for instructions on replacing the failed power supply."

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table B.28 - LibraryAlert AlertIndication Properties (Continued)

Event/Alert Summary	CIM_AlertIndication "Mapped" Properties from SSC-2 and SMC-2 Specifications			
	OtherAlert Type	Perceived Severity	ProbableCause Description	Recommended Action[ ]
	string	uint16	string	string
Power Consumption	"Power consumption of the drive is outside the specified range."	"3" = "Degraded/ Warning"	"The drive power consumption is outside the specified range."	""
Drive Maintenance	"The drive requires preventive maintenance (not cleaning)."	"3" = "Degraded/ Warning"	"Preventive maintenance of the drive is required."	Check the drive user's manual for device specific preventive maintenance tasks or call the drive supplier help line."
Hardware A	"The drive has a hardware fault that requires reset to recover."	"6" = "Critical"	"The drive has a hardware fault".	"1. Eject the media or magazine." "2. Reset the drive." "3. Restart the operation."
Hardware B	"The drive has a hardware fault that is not read/write related or requires a power cycle to recover."	"6" = "Critical"	"The drive has a hardware fault".	"1. Turn the drive off and then on again." "2. Restart the operation." "3. If the problem persists, call the drive supplier help line."
Interface	"The drive has identified an interface fault."	"3" = "Degraded/ Warning"	"Bad cable or drive interface."	"1. Check the cables and cable connections." "2. Restart the operation."

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table B.28 - LibraryAlert AlertIndication Properties (Continued)

Event/Alert Summary	CIM_AlertIndication "Mapped" Properties from SSC-2 and SMC-2 Specifications			
	OtherAlert Type	Perceived Severity	ProbableCause Description	Recommended Action[ ]
	string	uint16	string	string
Eject Media	"Error recovery action: Media Ejected"	"6" = "Critical"	""	"1. Eject the media or magazine." "2. Insert the media or magazine again." "3. Restart the operation."
Download Failure	"Firmware download failed."	"3" = "Degraded/ Warning"	"The firmware download has failed because you have tried to use the incorrect firmware for this drive."	"Obtain the correct firmware and try again."
Drive Humidity	"Drive humidity limits exceeded."	"3" = "Degraded/ Warning"	"Bad drive fan"	"Replace fan or drive enclosure".
Drive Temperature	"Drive temperature limits exceeded."	"3" = "Degraded/ Warning"	"Bad cooling fan"	"Replace fan or drive enclosure".
Drive Voltage	"Drive voltage limits exceeded."	"3" = "Degraded/ Warning"	"Bad drive power supply"	"Check the drive users manual for device specific preventive maintenance tasks or call the drive supplier help line."
Predictive Failure	"Predictive failure of drive hardware."	"6" = "Critical"	""	"A hardware failure of the drive is predicted. Call the drive supplier help line."

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table B.28 - LibraryAlert AlertIndication Properties (Continued)

Event/Alert Summary	CIM_AlertIndication "Mapped" Properties from SSC-2 and SMC-2 Specifications			
	OtherAlert Type	Perceived Severity	ProbableCause Description	Recommended Action[ ]
	string	uint16	string	string
Diagnostics Required	"The drive may have a hardware fault that may be identified by extended diagnostics (i.e., SEND DIAGNOSTIC command)."	"3" = "Degrading/ Warning"	"The drive may have a hardware fault."	"1. Run extended diagnostics to verify and diagnose the problem. Check the drive user's manual for device specific instructions on running extended diagnostic tests."
Loader Hardware A	"Loader mechanism is having trouble communicating with the drive."	"6" = "Critical"	"The changer mechanism is having difficulty communicating with the drive."	"1. Turn the autoloader off then on." "2. Restart the operation." "3. If a problem persists, call the drive supplier help line."
Loader Stray Media	"Stray media left in loader after previous error recovery."	"6" = "Critical"	"A media has been left in the autoloader by a previous hardware fault."	"1. Insert an empty magazine to clear the fault." "2. If the fault does not clear, turn the autoloader off and then on again." "3. If the problem persists, call the drive supplier help line."
Loader Hardware B	"Loader mechanism has a hardware fault."	"3"= "Degrading/ Warning"	"There is a problem with the autoloader mechanism."	""

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table B.28 - LibraryAlert AlertIndication Properties (Continued)

Event/Alert Summary	CIM_AlertIndication "Mapped" Properties from SSC-2 and SMC-2 Specifications			
	OtherAlert Type	Perceived Severity	ProbableCause Description	Recommended Action[ ]
	string	uint16	string	string
Loader Door	"Changer door open."	"6" = "Critical"	"The operation has failed because the autoloader door is open."	"1. Clear any obstructions from the autoloader door." "2. Eject the magazine and then insert it again." "3. If the fault does not clear, turn the autoloader off and then on again." "4. If the problem persists, call the drive supplier help line."
Loader Hardware C	"The loader mechanism has a hardware fault that is not mechanically related."	"6" = "Critical"	"The autoloader has a hardware fault."	"1. Turn the autoloader off and then on again." "2. Restart the operation." "3. If the problem persists, call the drive supplier help line. Check the autoloader user's manual for device specific instructions on turning the device power on and off."

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table B.28 - LibraryAlert AlertIndication Properties (Continued)

Event/Alert Summary	CIM_AlertIndication "Mapped" Properties from SSC-2 and SMC-2 Specifications			
	OtherAlert Type	Perceived Severity	ProbableCause Description	Recommended Action[ ]
	string	uint16	string	string
Loader Magazine	"Loader magazine not present."	"6" = "Critical"	"The autoloader cannot operate without the magazine. "	"1. Insert the magazine into the autoloader." "2. Restart the operation."
Loader Predictive Failure	"Predictive failure of loader mechanism hardware".	"3" = "Degrading/ Warning"		"A hardware failure of the changer mechanism is predicted. Call the drive supplier help line."
Load Statistics	"Drive or library powered down with media loaded."	"3" = "Degrading/ Warning"	"Media statistics have been lost at some time in the past."	""
Media Directory Invalid at Unload	"Error preventing the media directory being updated on unload."	"3" = "Degrading/ Warning"	"The directory on the media just unloaded has been corrupted."	"The directory can be rebuilt by reading all the data."
Media System area Write Failure	"Write errors while writing the system area on unload."	"6" = "Critical"	"The media just unloaded could not write its system area successfully."	"1. Copy data to another cartridge." "2. Discard the old cartridge."
Media System Area Read Failure	"Read errors while reading the system area on load."	"6" = "Critical"	"The media system area could not be read successfully at load time."	"1. Copy data to another cartridge."
No Start of Data	"Media damaged, bulk erased or incorrect format."	"6" = "Critical"	"The start of data could not be found on the media."	"1. Check that you are using the correct format media." "2. Discard the media or return the media to your supplier."

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table B.28 - LibraryAlert AlertIndication Properties (Continued)

Event/Alert Summary	CIM_AlertIndication "Mapped" Properties from SSC-2 and SMC-2 Specifications			
	OtherAlert Type	Perceived Severity	ProbableCause Description	Recommended Action[ ]
	string	uint16	string	string
Loading Failure	"The drive is unable to load the media".	"6" = "Critical"	"The operation has failed because the media cannot be loaded and threaded."	"1. Remove the cartridge, inspect it as specified in the product manual and retry the operation." "2. If the problem persists, call the drive supplier help line."
Library Hardware A	"Changer mechanism is having trouble communicating with the internal drive".	"6" = "Critical"	"The library mechanism is having difficulty communicating with the drive."	"1. Turn the library off then on." "2. Restart the operation." "3. If the problem persists, call the library supplier help line."
Library Hardware B	"Changer mechanism has a hardware fault".	"3" = "Degrading/ Warning"	""	"There is a problem with the library mechanism. If the problem persists, call the library supplier help line."
Library Hardware C	"The changer mechanism has a hardware fault that requires a reset to recover."	"6" = "Critical"	"The library has a hardware fault".	"1. Reset the library." "2. Restart the operation. Check the library user's manual for device specific instructions on resetting the device."

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table B.28 - LibraryAlert AlertIndication Properties (Continued)

Event/Alert Summary	CIM_AlertIndication "Mapped" Properties from SSC-2 and SMC-2 Specifications			
	OtherAlert Type	Perceived Severity	ProbableCause Description	Recommended Action[ ]
	string	uint16	string	string
Library Hardware D	"The changer mechanism has a hardware fault that is not mechanically related or requires a power cycle to recover."	"6" = "Critical"	"The library has a hardware fault."	"1. Turn the library off then on again." "2. Restart the operation." "3. If the problem persists, call the library supplier help line. Check the library user's manual for device specific instructions on turning the device power on and off."
Library Diagnostic Required	"The changer mechanism may have a hardware fault which would be identified by extended diagnostics."	"3" = "Degrading/ Warning"	"The library mechanism may have a hardware fault."	Run extended diagnostics to verify and diagnose the problem. Check the library user's manual for device specific instructions on running extended diagnostic tests."
Library Interface	"The library has identified an interface fault".	"6" = "Critical"	"Bad cable"	"1. Check the cables and connections." "2. Restart the operation."
Failure Prediction	"Predictive failure of library hardware".	"3" = "Degrading/ Warning"	""	"A hardware failure of the library is predicted. Call the library supplier help line."

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table B.28 - LibraryAlert AlertIndication Properties (Continued)

Event/Alert Summary	CIM_AlertIndication "Mapped" Properties from SSC-2 and SMC-2 Specifications			
	OtherAlert Type	Perceived Severity	ProbableCause Description	Recommended Action[ ]
	string	uint16	string	string
Library Maintenance	"Library preventative maintenance required."	"3" = "Degrading/ Warning"	""	"Preventive maintenance of the library is required. Check the library user's manual for device specific preventative maintenance tasks or call your library supplier help line."
Library Humidity Limits	"Library humidity limits exceeded".	"6" = "Critical"	"Library humidity range is outside the operational conditions".	""
Library Temperature Limits	"Library temperature limits exceeded".	"6" = "Critical"	"Library temperature is outside the operational conditions".	""
Library Voltage Limits	"Library voltage limits exceeded".	"6" = "Critical"	"Potential problem with a power supply."	""
Library Stray Media	"Stray cartridge left in library after previous error recovery".	"6" = "Critical"	"Cartridge left in picker or drive".	"1. Insert an empty magazine to clear the fault." "2. If the fault does not clear, turn the library off and then on again." "3. If the problem persists, call the library supplier help line."

--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table B.28 - LibraryAlert AlertIndication Properties (Continued)

Event/Alert Summary	CIM_AlertIndication "Mapped" Properties from SSC-2 and SMC-2 Specifications			
	OtherAlert Type	Perceived Severity	ProbableCause Description	Recommended Action[ ]
	string	uint16	string	string
Library Pick Retry	"Operation to pick a cartridge from a slot had to perform an excessive number of retries before succeeding".	"3" = "Degrading/ Warning"	"There is a potential problem with the drive ejecting cartridges or with the library mechanism picking a cartridge from a slot."	"1.Run diagnostics to determine the health of the Library." "2. If the problem persists, call the library supplier help line."
Library Place Retry	"Operation to place a cartridge in a slot had to perform an excessive number of retries before succeeding".	"3" = "Degrading/ Warning"	"Worn cartridge or bad storage slot/ magazine".	"1. No action needs to be taken at this time." "2. If the problem persists, call the library supplier help line."
Library Load Retry	"Operation to load a cartridge in a drive had to perform an excessive number of retries before succeeding".	"3" = "Degrading/ Warning"	"Worn cartridge or picker".	"1. Run diagnostics to determine the health of the library."
Library Door	"Library door open is preventing the library from functioning".	"6" = "Critical"	"The library has failed because the door is open."	"1. Clear any obstructions from the library door." "2. Close the library door." "3. If the problem persists, call the library supplier help line."

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table B.28 - LibraryAlert AlertIndication Properties (Continued)

Event/Alert Summary	CIM_AlertIndication "Mapped" Properties from SSC-2 and SMC-2 Specifications			
	OtherAlert Type	Perceived Severity	ProbableCause Description	Recommended Action[ ]
	string	uint16	string	string
Library Mailslot	"Mechanical problem with import/export mailslot".	"6" = "Critical"	"There is a mechanical problem with the library media mailslot."	"1. Check for wedged storage media in import/export mailslot".
Library Magazine	"Library magazine not present".	"6" = "Critical"	"Administrator has removed the library's magazine".	"1. Insert the magazine into the library." "2. Restart the operation."
Library Security	"Library door opened then closed during operation".	"3" = "Degrading/ Warning"	"Administrator is trying to remove or insert a storage media".	""
Library Security Mode	"Library security mode changed".	"2" = "Information"	"Administrator changed security mode".	"The library security mode has been changed. The library has either been put into secure mode, or the library has exited the secure mode. This is for information purposes only. No action is required."
Library Offline	"Library manually turned offline"	"2" = "Information"	"The library has been manually turned offline and is unavailable for use."	""
Library Drive Offline	"Library turned internal drive offline."	"2" = "Information"	"Drive failure".	"A drive inside the library has been taken offline. This is for information purposes only. No action is required."

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table B.28 - LibraryAlert AlertIndication Properties (Continued)

Event/Alert Summary	CIM_AlertIndication "Mapped" Properties from SSC-2 and SMC-2 Specifications			
	OtherAlert Type	Perceived Severity	ProbableCause Description	Recommended Action[ ]
	string	uint16	string	string
Library Scan Retry	"Operation to scan the bar code on a cartridge had to perform an excessive number of retries before succeeding".	"3" = "Degrading/ Warning"	"There is a potential problem with the bar code label or the scanner hardware in the library mechanism."	"1. No action needs to be taken at this time." "2. If the problem persists, call the library supplier help line."
Library Inventory	"Inconsistent media inventory".	"6" = "Critical"	"Media label has changed or there is bad bar code scanner subsystem problem."	"1. Redo the library inventory to correct inconsistency." "2. Restart the operation. Check the applications user's manual or the hardware user's manual for specific instructions on redoing the library inventory."
Library Illegal Operation	"Illegal operation detected".	"3" = "Degrading/ Warning"	"A library operation has been attempted that is invalid at this time."	""
Dual-Port Interface Error	"Failure of one interface port in a dual-port configuration"	"3" = "Degrading/ Warning"	"A redundant interface port on the library has failed."	""
Cooling Fan Failure	"One or more fans inside the library have failed. Internal flag state only cleared when all flags are working again".	"3" = "Degrading/ Warning"	"Bad cooling Fan".	""

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table B.28 - LibraryAlert AlertIndication Properties (Continued)

Event/Alert Summary	CIM_AlertIndication "Mapped" Properties from SSC-2 and SMC-2 Specifications			
	OtherAlert Type	Perceived Severity	ProbableCause Description	Recommended Action[ ]
	string	uint16	string	string
Power Supply	"Redundant power supply failure inside the library subsystem".	"3" = "Degrading/ Warning"	"Bad Power Supply"	"A redundant power supply has failed inside the library. Check the library user's manual for instructions on replacing the failed power supply."
Power Consumption	"Power consumption of one or more devices inside the library is outside the specified range".	"3" = "Degrading/ Warning"	"The library power consumption is outside the specified range."	""
Pass Through Mechanism Failure	"Error occurred in pass-through mechanism during self-test or while attempting to transfer a cartridge between library modules".	"6" = "Critical"	"A failure has occurred in the cartridge pass-through mechanism between two library modules."	""
Cartridge in Pass-through Mechanism	"Cartridge left in the pass-through mechanism between two library modules"	"6" = "Critical"	""	"A cartridge has been left in the pass-through mechanism from a previous hardware fault. Check the library user's guide for instructions on clearing this fault."
Unreadable barcode Labels	"Unable to read a bar code label on a cartridge during library inventory/ scan".	"2" = "Information"	"Bad Bar Code Labels or Scanner".	"The library was unable to read the bar code on a cartridge."

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

B.3.6.12 Optional Subprofiles

Table B.29 - Optional Profiles or Subprofiles

Name	Notes
None	

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

B.4 Extender profile

B.4.1 Description

NOTE This profile is experimental and is provided for information only. The contents of an experimental profile is likely to change as implementation experience is gained. Please provide any implementation feedback to Storage Management Initiative Technical Steering Group (td@snia.org).

A FC Extender is a network segment, consisting of 2 or more FC Extender node devices and network pipes that connect them. A FC Extender is dedicated to connect two fabric islands across a WAN or any other extension medium. A FC Extender node is a device that supports Fibre Channel communication over different communication technologies.

The domain of the components of the extender is defined by Network, a subclass of an AdminDomain and is necessary to host the Networking.

B.4.2 FC Extender profile as a topology

The ComputerSystem class is the core of the model. It is identified as an Extender node by the dedicated attribute being set to Extender. It has a group of services, FCExtenderNodeService consisting of IPService and TCPService or SRService subclassed from the ForwardingService. These services represent the status/ configuration of the FC Extender transport layer.

The Port group of classes contains the following classes: FCPort, EthernetPort and ATMPort. The FCPort class represents the connection of a FC Extender to a SAN. This class connects to other FCPort classes to represent Fibre channel connections. This class could be replaced with other port types to represent SANs based on other interconnect technology. The ATMPort class represents ATM link between FC Extender nodes and the EthernetPort represents an Ethernet link.

B.4.3 FC Extender profile as a connection

The intent to view FC Extender as a connection or collection of connections is to facilitate the connection manageability including capabilities to create, activate or delete a connection.

B.4.4 Standard dependencies

The Extender profile is based on the following standards.

Table B.30 - Extender Standard Dependencies

Standard	Version	Organization
CIM Specification	2.2	DMTF
CIM Operations over HTTP	1.1	DMTF
CIM Schema	2.8 Preliminary	DMTF

B.4.5 Profile dependencies

The Extender profile requires the Server Profile (7.3.7).

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

B.4.6 CIM Server Requirements

Table B.31 - Required Functional Profiles

Profile Required	Functional Group	Dependency
YES	Basic Read	None
NO	Basic Write	Basic Read
NO	Instance Manipulation	Basic Write
NO	Schema Manipulation	Instance Manipulation
YES	Association Traversal	Basic Read
NO	Query Execution	Basic Read
NO	Qualifier Declaration	Schema Manipulation
YES	Indication	None

B.4.6.1 Extrinsic methods

The CIM Server shall support extrinsic methods for the Server profile.

B.4.6.2 Discovery

The CIM Server shall support SLP discovery as defined in the CIM Operations over HTTP specification.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

B.4.7 Instance diagrams

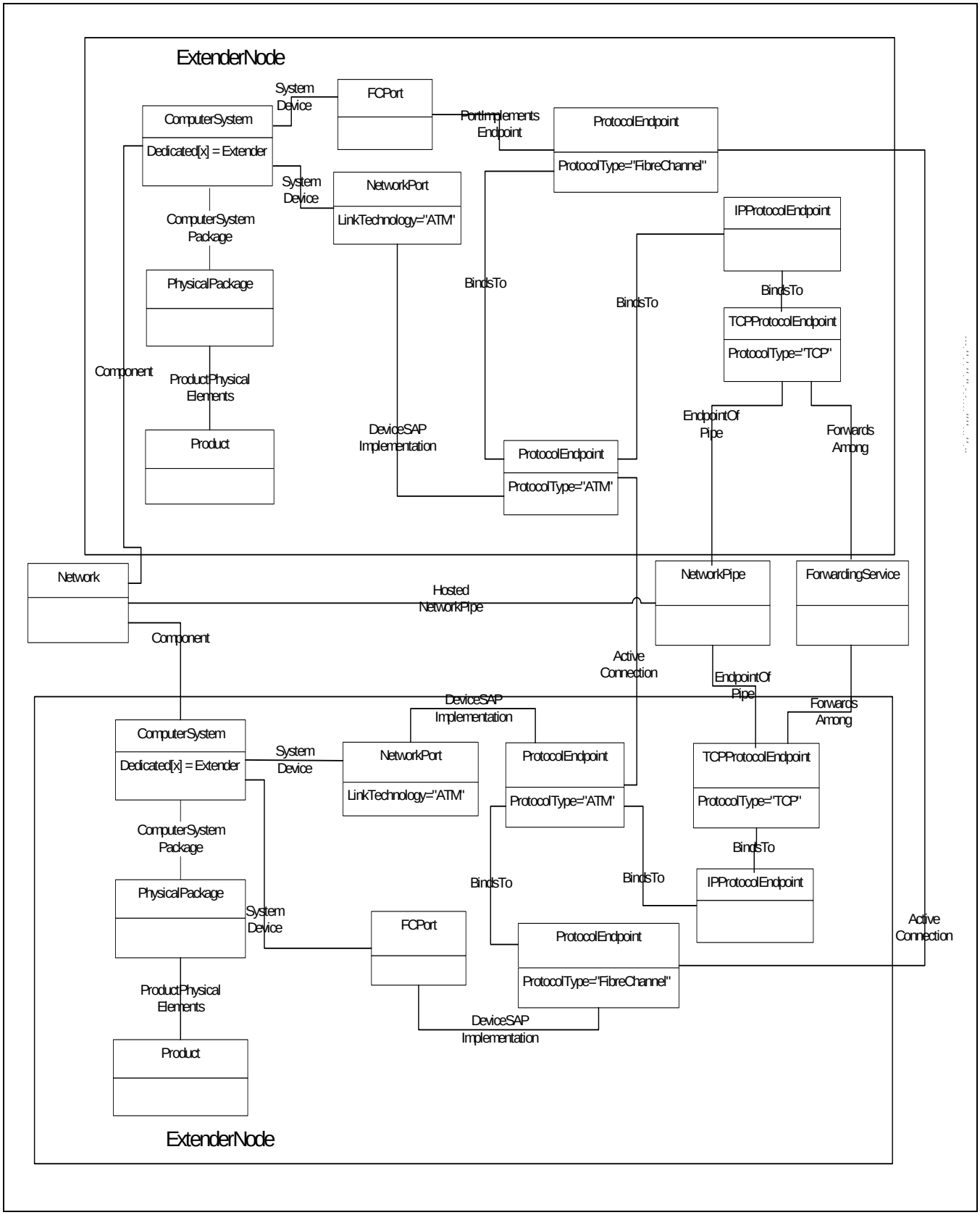


Figure B.7 - Extender Instance Diagram

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

B.4.8 Durable names and correlatable IDs of the Profile

B.4.8.1 Durable names exported

There are no durable name exports defined for this profile.

B.4.8.2 Correlatable IDs used

There are no correlatable IDs defined for this profile.

B.4.9 Methods

There are no methods defined for this profile.

B.4.10 Client considerations

There are no implementation requirements defined for this profile.

B.4.11 Recipes

There are no recipes defined for this profile.

B.4.12 Instrumentation requirements

There are no implementation requirements defined for this profile.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

B.4.13 Required CIM Elements

Table B.32 - Required CIM Elements

Profile Classes & Associations	Notes
ActiveConnetction for FC (B.4.14.2)	For FC
ActiveConnetction for ATM (B.4.14.1)	For ATM
BindsTo (B.4.14.3)	
BindsTo (B.4.14.3)	For the EntenderNodes
FCPort (B.4.14.6)	
ForwardsAmong (B.4.14.8)	
ForwardingService (B.4.14.7)	
HostedNetworkPipe (B.4.14.9)	
IPProtocolEndpoint (B.4.14.10)	
Network (B.4.14.12)	
Network (B.4.14.12)	
IPProtocolEndpoint (B.4.14.10)	For ATM Port
DeviceSAPImplementation (B.4.14.5)	
ProtocolEndPoint (B.4.14.14)	For FC
ProtocolEndPoint (B.4.14.14)	For ATM
TCPPrototocolEndpoint (B.4.14.15)	
TCPPrototocolEndpoint (B.4.14.15)	
Packages	
Physical Package “package” (7.3.2.2) Software Package (7.3.2.3)	
Associated Indications	
Creation/Deletion of FCPort	SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_FCPort SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_FCPort.
Creation/Deletion of ComputerSystem	SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_ComputerSystem SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_ComputerSystem.
Creation/Deletion of ActiveConnection	SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_ActiveConnection SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_ActiveConnection.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table B.32 - Required CIM Elements (Continued)

Profile Classes & Associations	Notes
Creation/Deletion of ATM Port	SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_Port and SourceInstance.LinkTechnology = 'ATM' SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_Port and SourceInstance.LinkTechnology = 'ATM'.
Change in status of ComputerSystem	SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_ComputerSystem AND SourceInstance.OperationalStatus <> PreviousInstance.OperationalStatus.
Change in status of FCPort	SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_Port AND SourceInstance.OperationalStatus <> PreviousInstance.OperationalStatus.
Change in status of ATM Port	SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_FCPort AND SourceInstance.OperationalStatus <> PreviousInstance.OperationalStatus and SourceInstance.LinkTechnology = 'ATM'.

--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

B.4.14 Required Properties for CIM Elements

B.4.14.1 ActiveConnetction for ATM

Table B.33 - Required Properties for ActiveConnection for ATM

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Antecedent	ref	key	ProtocolEndpoint reference to ProtocolEndpoint.PrototcolType=”ATM”.
Dependent	ref	key	ProtocolEndpoint reference to ProtocolEndpoint.PrototcolType=”ATM”.

B.4.14.2 ActiveConnetction for FC

Table B.34 - Required Properties for ActiveConnection for FC

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Antecedent	ref	key	ProtocolEndpoint reference to ProtocolEndpoint.PrototcolType=”ATM”.
Dependent	ref	key	ProtocolEndpoint reference to ProtocolEndpoint.PrototcolType=”ATM”.

B.4.14.3 BindsTo

Table B.35 - Required Properties for BindsTo

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Antecedent	ref	key	ProtocolEndpoint reference.
Dependent	ref	key	ProtocolEndpoint reference.

B.4.14.4 ComputerSystem

Table B.36 - Required Properties for ComputerSystem

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
ElementName	string		Switch Symbolic Name. For Platform it is the Platform Label.
CreationClassName	string	maxlen(256), key	Name of Class.
Name	string	maxlen(256), key	For Switches, it is the FC WWN. For Platforms, it is the Platform Name if available.
NameFormat	string	override	
Dedicated	int16[ ]		“Extender”.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table B.36 - Required Properties for ComputerSystem (Continued)

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Name	string	maxlen(256), key	For Switches, it is the FC WWN. For Platforms, it is the Platform Name if available.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

B.4.14.5 DeviceSAPImplementation

Table B.37 - Required Properties for DeviceSAPImplementation

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Antecedent	ref	key	ProtocolEndpoint reference
Dependent	ref	key	ProtocolEndpoint reference

B.4.14.6 FCPort

Table B.38 - Required Properties for FCPort

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
ElementName	**string**		Port 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.
OperationalStatus	uint16		See Table 132.
DeviceID	string	key, maxlen (64)	Opaque.
PortType	uint16	override	FC-GS Port.Type The specific mode currently enabled for the Port. The values: \"N\" = Node Port, \"NL\" = Node Port supporting FC arbitrated loop, \"E\" = Expansion Port connecting fabric elements (for example, FC switches), \"F\" = Fabric (element) Port, \"FL\" = Fabric (element) Port supporting FC arbitrated loop, and \"B\" = Bridge Port. PortTypes are defined in ISO/IEC 14165-414. When set to 1 (\"Other\"), the related property OtherPortType contains a string description of the port's type. PortType is defined to force consistent naming of the 'type' property in subclasses and to guarantee unique enum values for all instances of NetworkPort. A range of values, DMTF_Reserved, has been defined that allows subclasses to override and define their specific port types. Vendor Reserved = 16 000..65 535 can be used if the PortType is not one already defined in the above enumerations and a vendor subclass is defined specifying the appropriate “value” and “valuemap”.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table B.38 - Required Properties for FCPort (Continued)

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
OtherNetworkPortType	string		Describes the type of module, when PortType is set to 1 (\"Other\").
LinkTechnology	uint16		For FibreChannel, “FC”.
OtherLinkTechnology	string		A string value describing LinkTechnology when it is set to 1, \"Other\".
PermanentAddress	string	maxlen (64)	For FibreChannel, it is the Fibre Channel Port WWN.
NetworkAddresses[ ]	string	maxlen (64), arraytype ("indexed")	For Fibre Channel end device ports, it is the Fibre Channel ID. For Switches, it should be Null.
ActiveCOS	uint16[ ]		FC-GS Class Of Service An array of integers indicating the Classes of Service that are active. Not applicable for switches (e.g. NULL).
ActiveFC4Types	uint16[ ]		FC-GS FC4-TYPE An array of integers indicating the Fibre Channel FC-4 protocols currently running. Not applicable for switches (e.g., NULL).

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

B.4.14.7 ForwardingService

Table B.39 - Required Properties for ForwardingService

Property/ Method	Type	Qualifier/Parameter	Notes
SystemCreationClassName	string	maxlen (256),key,propagated	
SystemName;	string	maxlen (256),key,propagated	
CreationClassName	string	maxlen (256),key	
Name	string	maxlen (256),key,override	
ProtocolType	unit16		
OtherProtocolType	string		

B.4.14.8 ForwardsAmong

Table B.40 - Required Properties for ForwardsAmong

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Antecedent	ref	key	ProtocolEndpoint reference.
Dependent	ref	key	ForwardingService reference.

B.4.14.9 HostedNetworkPipe

Table B.41 - Required Properties for HostedNetworkPipe

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Antecedent	ref	key	ProtocolEndpoint reference to TCPProtocolEndpoint.
Dependent	ref	key	ProtocolEndpoint reference to TCPProtocolEndpoint.

B.4.14.10 IPProtocolEndpoint

Table B.42 - Requited Properties for IPProtocolEndpoint

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Name	string	maxlen (256)	
CreationClassName	string	key, maxlen (256)	
SystemCreationClassName	string	key, maxlen (256)	
SystemName	string	key, maxlen (256)	

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table B.42 - Requited Properties for IPProtocolEndpoint (Continued)

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
NameFormat	string	maxlen (256)	heuristic that ensures unique name.
ProtocolType	string	maxlen (64), valuemap {} values {}	"IPv4" or “IPv6”.
OtherTypeDescription	string	maxlen (64)	used when ProtocolType = "Other".
IPv4Address	string		
IPv6Address	string		

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

B.4.14.11 NetworkPort

Table B.43 - Required Properties for NetworkPort

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
ElementName	**string**		Port 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.
OperationalStatus	uint16		Table 132.
DeviceID	string	key, maxlen (64)	Opaque.
PortType	uint16	override	FC-GS Port.Type The specific mode currently enabled for the Port. The values: \"N\" = Node Port, \"NL\" = Node Port supporting FC arbitrated loop, \"E\" = Expansion Port connecting fabric elements (for example, FC switches), \"F\" = Fabric (element) Port, \"FL\" = Fabric (element) Port supporting FC arbitrated loop, and \"B\" = Bridge Port. PortTypes are defined in ISO/IEC 14165-414. When set to 1 (\"Other\"), the related property OtherPortType contains a string description of the port's type. PortType is defined to force consistent naming of the 'type' property in subclasses and to guarantee unique enum values for all instances of NetworkPort. A range of values, DMTF_Reserved, has been defined that allows subclasses to override and define their specific port types. Vendor Reserved = 16 000..65 535 can be used if the PortType is not one already defined in the above enumerations and a vendor subclass is defined specifying the appropriate “value” and “valuemap”.
OtherNetworkPortType	string		Describes the type of module, when PortType is set to 1 (\"Other\").
LinkTechnology	uint16		For FibreChannel, “FC”.
OtherLinkTechnology	string		A string value describing LinkTechnology when it is set to 1, \"Other\".
PermanentAddress	string	maxlen (64)	For FibreChannel, it is the Fibre Channel Port WWN.
NetworkAddresses[ ]	string	maxlen (64), arraytype ("indexed")	For Fibre Channel end device ports, it is the Fibre Channel ID. For Switches, it should be Null.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

B.4.14.12 Network

Table B.44 - Required Properties for Network

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
CreationClassName	string	maxlen(256), key	Name of Class.
Name	string	maxlen(256), key, override	
NameFormat	string	maxlen(64)	

B.4.14.13 NetworkPipe

Table B.45 - Required Properties for NetworkPipe

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
InstanceID	string		
OperationalStatus	unint16 (enum)		

B.4.14.14 ProtocolEndPoint

Table B.46 - Requited Properties for ProtocolEndpoint

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Name	string	maxlen (256)	
CreationClassName	string	key, maxlen (256)	
SystemCreationClassName	string	key, maxlen (256)	
SystemName	string	key, maxlen (256)	
NameFormat	string	maxlen (256)	heuristic that ensures unique name.
ProtocolType	string	maxlen (64), valuemap {} values {}	"IPv4", "Fibre Channel","InfiniBand", …
OtherTypeDescription	string	maxlen (64)	used when ProtocolTYpe = "Other".

B.4.14.15 TCPPrototocolEndpoint

Table B.47 - Requited Properties for TCPPrototocolEndpoint

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Name	string	maxlen (256)	
CreationClassName	string	key, maxlen (256)	

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table B.47 - Requited Properties for TCPPrototocolEndpoint (Continued)

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
SystemCreationClassName	string	key, maxlen (256)	
SystemName	string	key, maxlen (256)	
NameFormat	string	maxlen (256)	heuristic that ensures unique name
ProtocolType	string	maxlen (64), valuemap {} values {}	"IPv4" or “IPv6”
OtherTypeDescription	string	maxlen (64)	used when ProtocolTYpe = "Other"

B.4.14.16 SystemDevice

Table B.48 - Required Properties for SystemDevice

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
GroupComponent	ref	override	System Reference.
PartComponent	ref	override	LogicalDevice Reference.

B.4.15 Optional Subprofiles

Table B.49 - Optional Profiles or Subprofiles

Name	Notes
None	

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

B.5 Management Appliance profile

B.5.1 Description

NOTE This profile is experimental and is provided for information only. The contents of an experimental [profile/subprofile] is likely to change as implementation experience is gained. Please provide any implementation feedback to Storage Management Initiative Technical Steering Group (td@snia.org).

A **Management Appliance** is a computer system dedicated to running management software. In most cases the management software is accessed remotely through Web interfaces. This model is designed to allow for the discovery of the appliance and the applications running on it.

B.5.2 Standard dependencies

The Management Appliance profile is based on the following standards.

Table B.50 - Management Appliance Standard Dependencies

Standard	Version	Organization
CIM Specification	2.2	DMTF
CIM Operations over HTTP	1.1	DMTF
CIM Schema	2.8 Preliminary	DMTF

B.5.3 Profile dependencies

The Management Appliance profile relies on the following profiles.

- Server Profile (7.3.7)

B.5.4 CIM Server requirements

B.5.4.1 Functional profiles

Table B.51 - Required Functional Profiles

Profile Required	Functional Group	Dependency
YES	Basic Read	None
NO	Basic Write	Basic Read
NO	Instance Manipulation	Basic Write
NO	Schema Manipulation	Instance Manipulation
YES	Association Traversal	Basic Read
NO	Query Execution	Basic Read
NO	Qualifier Declaration	Schema Manipulation
NO	Indication	None

B.5.4.2 Extrinsic methods

The CIM Server shall support extrinsic methods for the Server profile.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

B.5.4.3 Discovery

The CIM Server shall support SLP discovery as defined in the CIM Operations over HTTP specification.

B.5.5 Instance diagrams

B.5.5.1 Overview

The basic Management appliance profile provides a read-only view of the system. The various subprofiles indicated in Figure B.8 extend this description. Refer to B.5.13, “Optional Subprofiles” for more information on these optional extensions. This profile also includes the mandatory Physical Package package, which describes the physical layout of the system and includes product identification information.

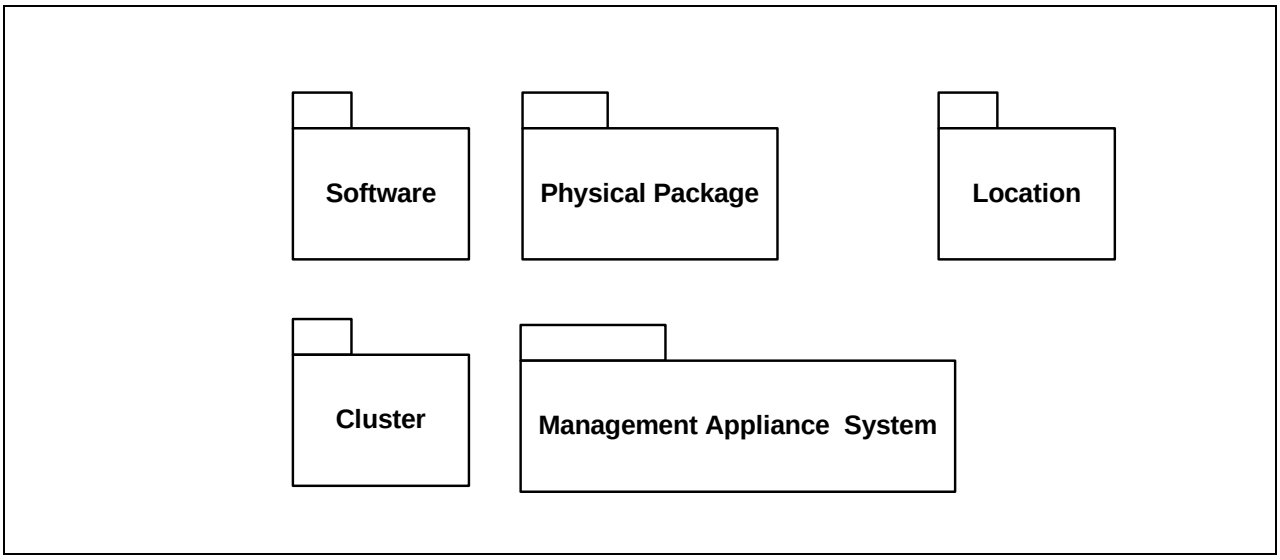


Figure B.8 - Management Appliance Subprofile Diagram

B.5.5.2 Basic Management Appliance Instance

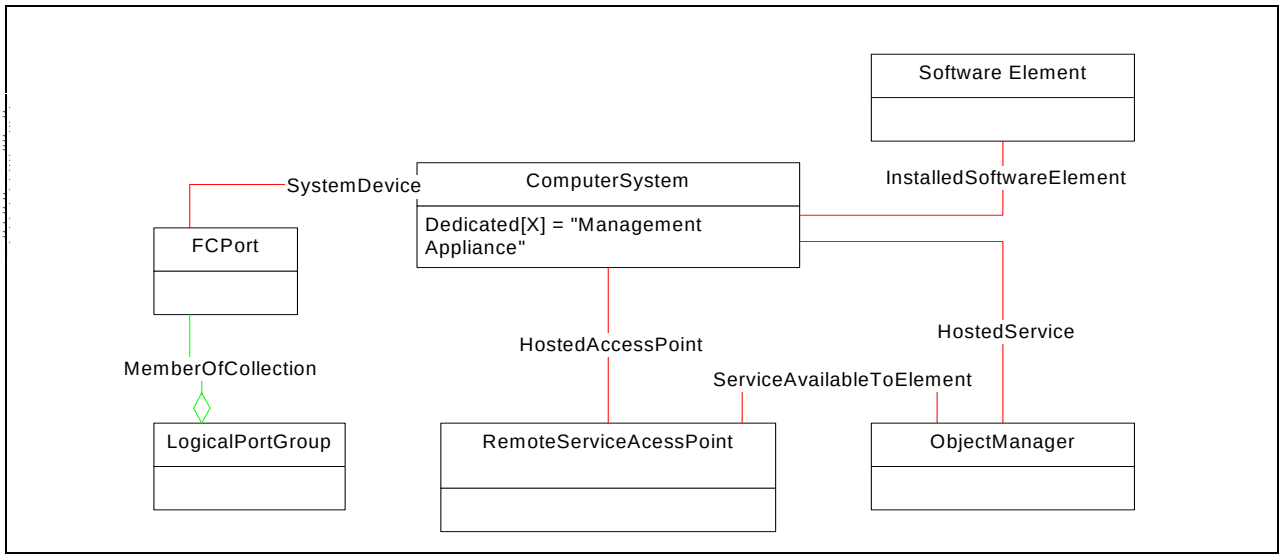


Figure B.9 - Management Appliance Instance Diagram

The Management Appliance model consists of 4 major groups of classes (Core, Port, Physical and Software).

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

The ComputerSystem class is the core of the model. It is identified as a Management Appliance by the Dedicated attribute being set to “Management Appliance”.

The FCPort class and LogicalPortGroup class represents the connection of the Management Appliance to the SAN. The FcPort class connects to other FCPort classes to represent fibre channel connections. This class could be replaced with other port types to represent SANs based on other interconnect technology.

The SoftwareElement class represents the management utilities that are running on the appliance. The instance diagram shows the SoftwareElement class subclassed to represent a CIMOM. The ObjectManager class is a subclass of CIM_SoftwareElement and represents the CIMOM. The ObjectManagerCommunicationMechanism class contains the URL to access the CIMOM.

The Client can get a list of available applications by locating the ComputerSystem class and enumerating the group of SoftwareElement classes by traversing the InstalledSoftwareElement associations.

B.5.5.3 Controller software

Information on the installed operating system is represented by the common subprofile Software Package. This is linked to the controller using an InstalledSoftwareElement association.

B.5.5.4 Physical modelling

The physical aspects of the metadata controller are represented by the Common Package “Physical Package” and the optional subprofile “Location”. See these common sections for more details.

B.5.5.5 Cluster

The Management Appliance can be implemented by a fault tolerate cluster. This part of the model is described by the optional Cluster common Subprofile. See these common sections for more details.

B.5.6 Durable names and correlatable IDs of the Profile

B.5.6.1 Durable names exported

For Fibre Channel port, the durable name is the Port WWN. It is found in FCPort.PermanentAddress.

For the appliance itself (the computer system), the Name property contains a durable name. The format of this name is defined by the NameFormat property.

B.5.6.2 Correlatable IDs used

B.5.7 Methods

There are no methods defined for this profile.

B.5.8 Client considerations

There are no client considerations defined for this profile.

B.5.9 Recipes

There are no recipes defined for this profile.

B.5.10 Instrumentation requirements

The purpose of the Management Appliance is to be a platform to run management applications on. Each of the applications running on the appliance should be modelled with both a SoftwareElement class and a RemoteServiceAccessPoint. The SoftwareElement describes the application including name, version and other. This class if often subclassed for each type of application. The RemoteServiceAccessPoint or

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

ObjectManagerCommunicationMechanism class contains a network address or URL to access the application.

B.5.11 Required CIM Elements

Table B.52 - Required CIM Elements

Profile Classes & Associations	Notes
ComputerSystem (B.5.12.1)	
FCPort (B.5.12.2)	
HostedService (B.5.12.3)	
InstalledSoftwareElement (B.5.12.4)	
LogicalPortGroup (B.5.12.5)	
MemberOfCollection (B.5.12.6)	
ObjectManager	
RemoteServiceAccessPoint (B.5.12.7)	
ServiceAvailableToElement (B.5.12.8)	
SoftwareElement (B.5.12.9)	
Packages	
Physical Package “package” (7.3.2.2)	
Associated Indications	
None	

B.5.12 Required Properties for CIM Elements

B.5.12.1 ComputerSystem

Table B.53 - Required Properties for ComputerSystem

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
ElementName	string		User Friendly name.
OperationalStatus	uint16		Status of array.
Name	string	key	The identifier for the Array (e.g., IP address or FC world-wide name).
NameFormat	string		The format of the Name property.
Dedicated	int16[ ]	“blockserver”, “metadatacontroller”	The use of this ComputerSystem.
PrimaryOwnerContact	string		Optional.
PrimaryOwnerName	string		Optional.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

B.5.12.2 FCPort

Table B.54 - Required Properties for FCPort

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
ElementName	**string**		User friendly name/caption for port.
OperationalStatus	uint16		Status of device.
DeviceID	string	key	Opaque.
PortType	uint16		FC-GS Port.Type The specific mode currently enabled for the Port. The values: \"N\" = Node Port, \"NL\" = Node Port supporting FC arbitrated loop, \"E\" = Expansion Port connecting fabric elements (for example, FC switches), \"F\" = Fabric (element) Port, \"FL\" = Fabric (element) Port supporting FC arbitrated loop and \"B\" = Bridge Port. PortTypes are defined in ISO/IEC 14165-414. When set to 1 (\"Other\"), the related property OtherPortType contains a string description of the port's type. PortType is defined to force consistent naming of the 'type' property in subclasses and to guarantee unique enum values for all instances of NetworkPort. A range of values, DMTF_Reserved, has been defined that allows subclasses to override and define their specific port types. Vendor Reserved = 16 000..65 535 can be used if the PortType is not one already defined in the above enumerations and a vendor subclass is defined specifying the appropriate “value” and “valuemap”.
PermanentAddress	string		The WWN of the port.
NetworkAddresses[ ]	string		The Fibre Channel address of the port (optional).
Speed	uint64		Speed of zero represents a link not established. 1 Gbit is 1 062 500 000 bit/s 2 Gbit is 2 125 000 000 bit/s 4 Gbit is 4 250 000 000 bit/s) 10 Gbit single channel variants are 10 518 750 000 bit/s 10 Gbit four channel variants are 12 750 000 000 bit/s This is the raw bit rate.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

B.5.12.3 HostedService

Table B.55 - Required Properties from HostedService

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Antecedent	ref		The Computer System.
Dependent	ref		The Service.

B.5.12.4 InstalledSoftwareElement

Table B.56 - Required Properties from Installed SoftwareElement

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Antecedent	ref		
Dependent	ref		

B.5.12.5 LogicalPortGroup

Table B.57 - Required Properties for LogicalPortGroup

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
InstanceName	string		Node Symbolic Name
SystemCreationClassName	string	propagated,key	
SystemName	string	propagated, key	
InstanceID	string	key	Node WWN||FC-GS InterconnectElement.Name, REQUIRED

B.5.12.6 MemberOfCollection

Table B.58 - Required Properties for MemberOfCollection

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Collection	ref	override	The reference to the collection.
Member	ref	override	The reference to the member.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

B.5.12.7 RemoteServiceAccessPoint

Table B.59 - Required Properties for RemoteServiceAccessPoint

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
SystemName	string	key	
SystemCreationClassName	string	key	
CreationClassName	string	key	
Name	string	key	
AccessInfo	string		FCGS||InterconnectElement.Management Address
InfoFormat	uint16		
OtherInfoFormatDescription	string		

B.5.12.8 ServiceAvailableToElement

ServiceAvailableToElement is not subclassed from anything.

Table B.60 - Required Properties for ServiceAvailableToElement

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Service	ref	override	The Service that is available.
ManagedElement	ref	override	The ManagedElement that may use the Service.

B.5.12.9 SoftwareElement

SoftwareElement is subclassed from LogicalElement.

Table B.61 - Required Properties for SoftwareElement

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Description	string		Longer description.
ElementName	string		User Friendly name.
OperationalStatus[ ]	uint16		
StatusDescriptions[ ]	string		
Name	string	key, maxlen (256), override	The name used to identify this SoftwareElement.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table B.61 - Required Properties for SoftwareElement (Continued)

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Version	string	key, maxlen (64)	Software Version should be in the form <Major>.<Minor>.<Revision> or <Major>.<Minor><letter><revision>.
SoftwareElementState	uint16	key	The SoftwareElementState is defined in this model to identify various states of a SoftwareElement's life cycle.
SoftwareElementID	string	key, maxlen(256)	This is an identifier for the SoftwareElement and is designed to be used in conjunction with other keys to create a unique representation of the element.
Manufacturer	string	maxlen(256)	Manufacturer of this SoftwareElement.
BuildNumber	string	maxlen(64)	The internal identifier for this compilation of SoftwareElement.
SerialNumber	string	maxlen(64)	The assigned serial number of this SoftwareElement.
IdentificationCode	string	maxlen(64)	The manufacturer's identifier for this SoftwareElement. Often this is a stock keeping unit (SKU) or a part number.

B.5.13 Optional Subprofiles

Table B.62 - Optional Profiles or Subprofiles

Name	Notes
Location subprofile (7.3.3.6)	
Software Subprofile (7.3.3.7)	
Cluster Subprofile (7.3.3.3)	

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

B.6 Out of Band Virtualizer Profile

B.6.1 Description

NOTE This profile is experimental and is provided for information only. The contents of an experimental profile is likely to change as implementation experience is gained. Please provide any implementation feedback to Storage Management Initiative Technical Steering Group (td@snia.org).

The Out-of-Band Virtualization system uses storage provided by array systems to create a seamless pool of storage. The virtualization system in turn allocates volumes from the pool for host systems to use. The system uses a control element called a “Metadata Controller” to control the system and maintain the mapping of array storage to host volumes. The system uses a separate component called a “Translation engine” to execute I/Os and perform the translation in real time. The Translation Engine may be implement as a stand-alone package or as part of other packages (i.e., host software, HBA or Fibre channel switch).

Below is a block diagram of an out-of-band virtualization system. At the top of the diagram are the host systems. They perform I/O to volumes presented to them by the Translation Engines. The host may communicate with one or more Translation Engines. The same Storage volume can be instantiated on any or all the Translation Engines at the same time. The next component is the Translation Engine. It is controlled by and receives translation information from the Metadata Controller. It uses this information to redirect and perform I/Os for a host system. Off to the side, the metadata Controller controls the virtualization system and provides translation information to the translation engines but isn’t actively involved in I/O. The last component is the array system. It is represented in its own namespace and would be supported by its own model and

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

profile. The virtualization model interfaces with the array and host system. Durable names are used to connect them together.

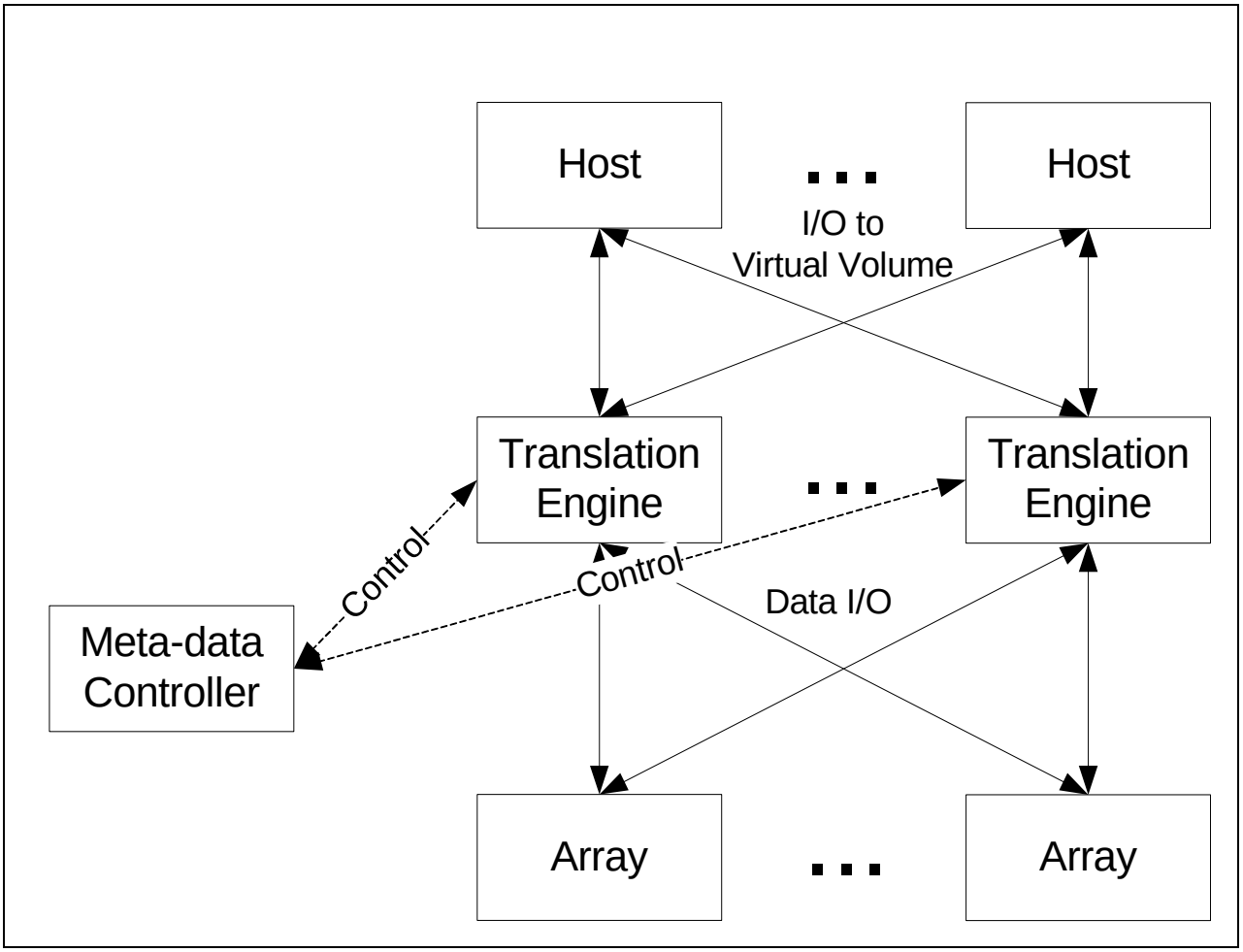


Figure B.10 - Out-of-band Virtualization Block Diagram

B.6.2 Standard dependencies

The Out-of-band Virtualizer profile is based on the following standards.

Table B.63 - OutofBand Virtualizer Standard Dependencies

Standard	Version	Organization
CIM Specification	2.2	DMTF
CIM Operations over HTTP	1.1	DMTF
CIM Schema	2.8 Preliminary	DMTF

B.6.3 Profile dependencies

The Out-of-band Virtualizer profile requires the Server Profile (7.3.7)

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

B.6.4 CIM Server requirements

B.6.4.1 Functional Profiles

Table B.64 - Required Functional Profiles

Profile Required	Functional Group	Dependency
YES	Basic Read	None
NO	Basic Write	Basic Read
NO	Instance Manipulation	Basic Write
NO	Schema Manipulation	Instance Manipulation
YES	Association Traversal	Basic Read
NO	Query Execution	Basic Read
NO	Qualifier Declaration	Schema Manipulation
YES	Indication	None

B.6.4.2 Extrinsic methods

Although there are some extrinsic methods defined within classes in this profile, they are not needed for this Profile. However, subprofiles do require the use of Extrinsic methods.

B.6.4.3 Discovery

The CIM Server shall support SLP discovery as defined in the CIM Operations over HTTP specification.

B.6.5 Instance diagrams

B.6.5.1 Overview

The modelling in this standard is split into various clauses that describe how to model particular elements of a virtualization system. The diagrams used in this document are 'Instance' diagrams implying the actual classes that you implement rather than the class hierarchy diagrams often used to show CIM models. This is felt to be easier to understand. Please refer to the DMTF CIM Schema for information on class inheritance information and full information on the properties and methods used.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

The subprofile diagram below is split into two parts, the metadata Controller and the Translation Engine. A virtualization model has one metadata Controller and one or more Translation Engines.

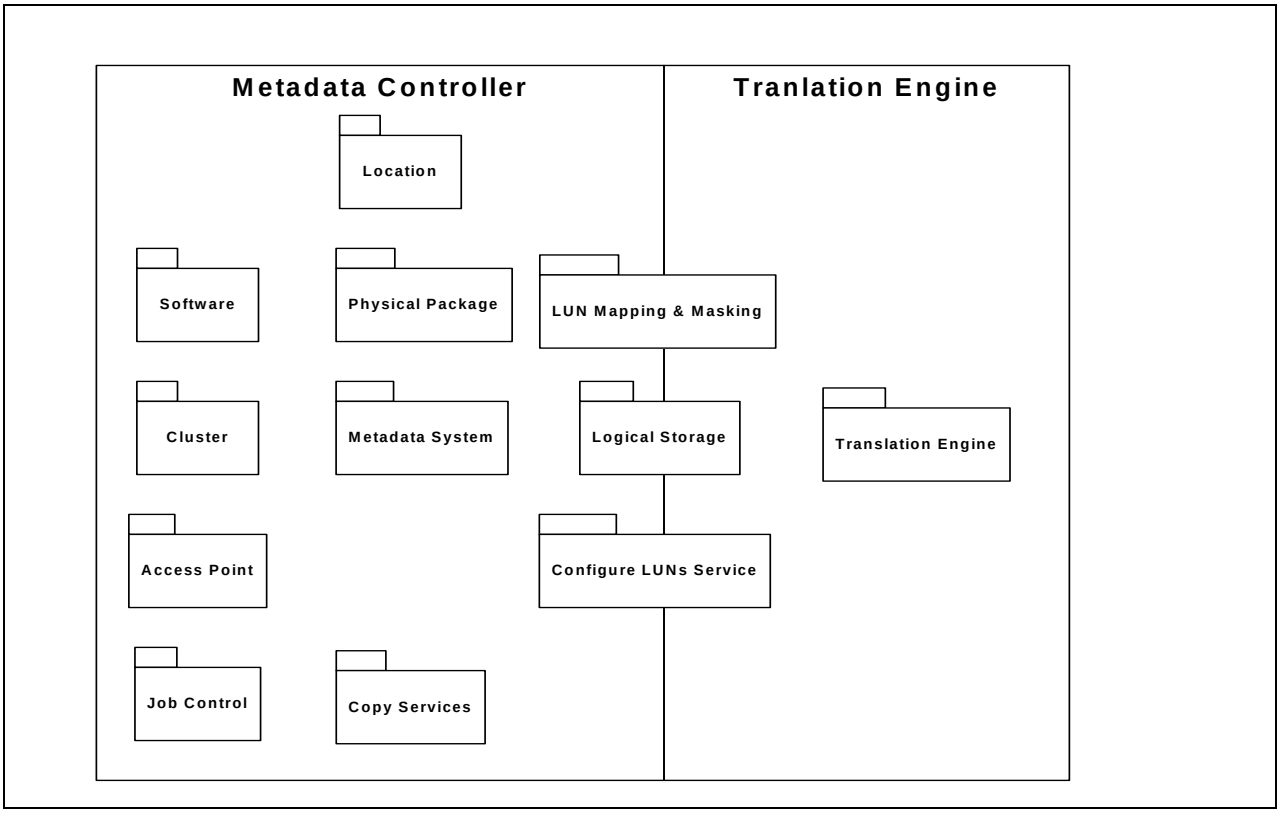


Figure B.11 - Out-of-Band Virtualization Subprofile Diagram

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

B.6.5.2 Metadata System Instance

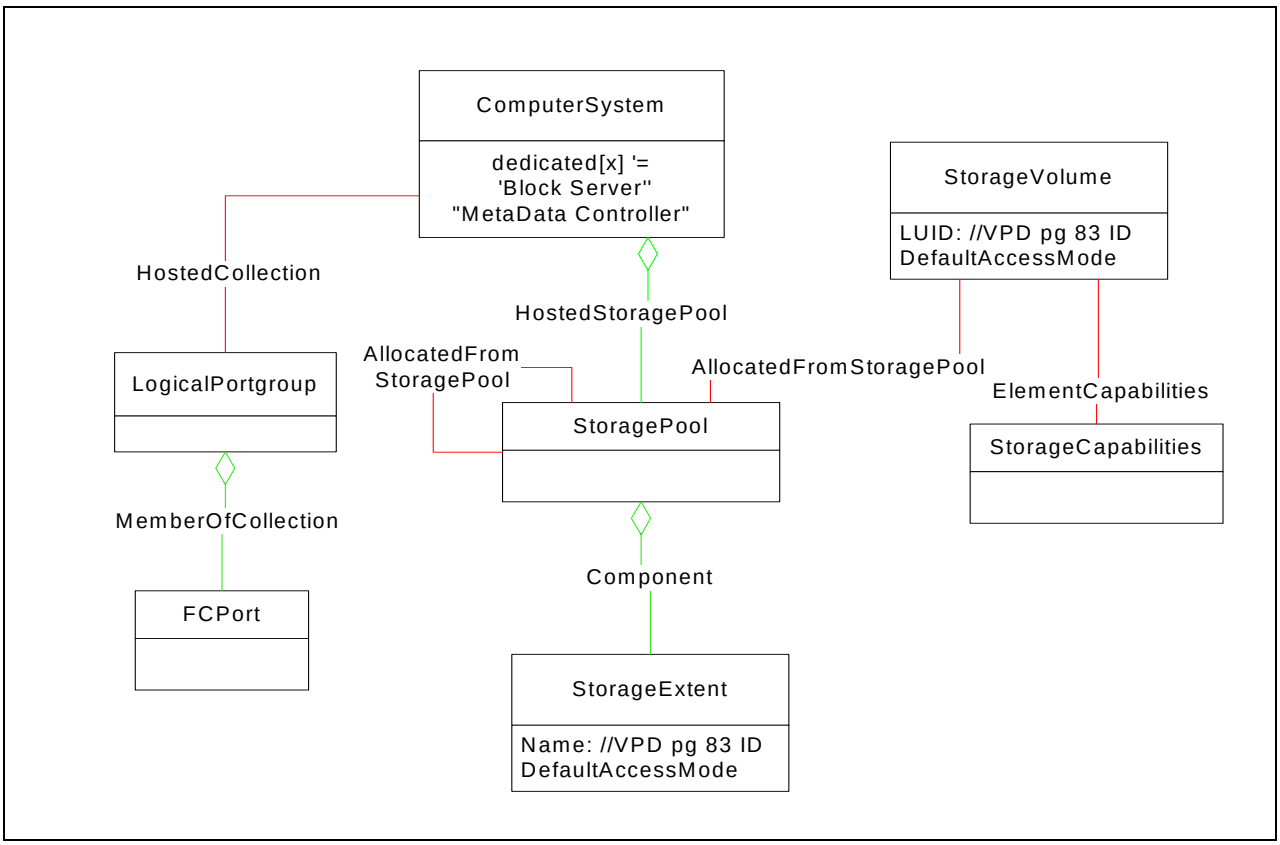


Figure B.12 - Metadata System Instance Diagram

B.6.5.2.1 Overview

The metadata Controller controls the virtualization system. It holds the definition of how the array storage is mapped to host volumes and controls the translation engines. The model allows the metadata controller to be a cluster or contain redundant components but the components act as a single system. The ComputerSystem class and common Cluster Subprofile model this Internal Communication.

The FCPort near the bottom-left represents the Fibre Channel port used by the Metadata Controllers to communicate with each other or the translation engines. This port is not used for user data transfers.

The StoragePool classes in the center of the diagram represents the mapping from array storage to Volumes for host access. The pool is hosted on the metadata controller and services to control it are host on the same controller. The StorageVolume at the bottom of the screen represents the storage from external arrays used by the mapping. These Storage Volumes are connected to the pool using the Component association.

StorageVolumes at the upper right are the volumes created from the StoragePool and are accessible from hosts. These Volumes are connected to ports on the translation engines. The mapping to hosts is also shown on the Metadata controller. The StorageVolumes are described by the StorageCapabilities class connected to them by the ElementsCapabilities association.

B.6.5.2.2 Controller software

Information on the installed controller software is represented by the common Software Package. This is linked to the controller using an InstalledSoftwareElement association.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

B.6.5.2.3 Device Management Tool Access

Most devices now have a web GUI to allow device specific configuration. This is modelled using the common subprofile “Access Point”.

B.6.5.2.4 Physical Modelling

The physical aspects of the metadata controller are represented by the Common Package “Physical Package” and the optional subprofile “Location”. See these sections for more details.

B.6.5.2.5 Services

The metadata controller hosts the services used to control the virtualization system. These services are optional and modelled by “Configuration LUNs Service”, “Copy Services” and “Job Control” subprofiles. The hosted services are accessed through the metadata controller, but affect objects on both the metadata controller and translation engine.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

B.6.5.3 Translation Engine Instance

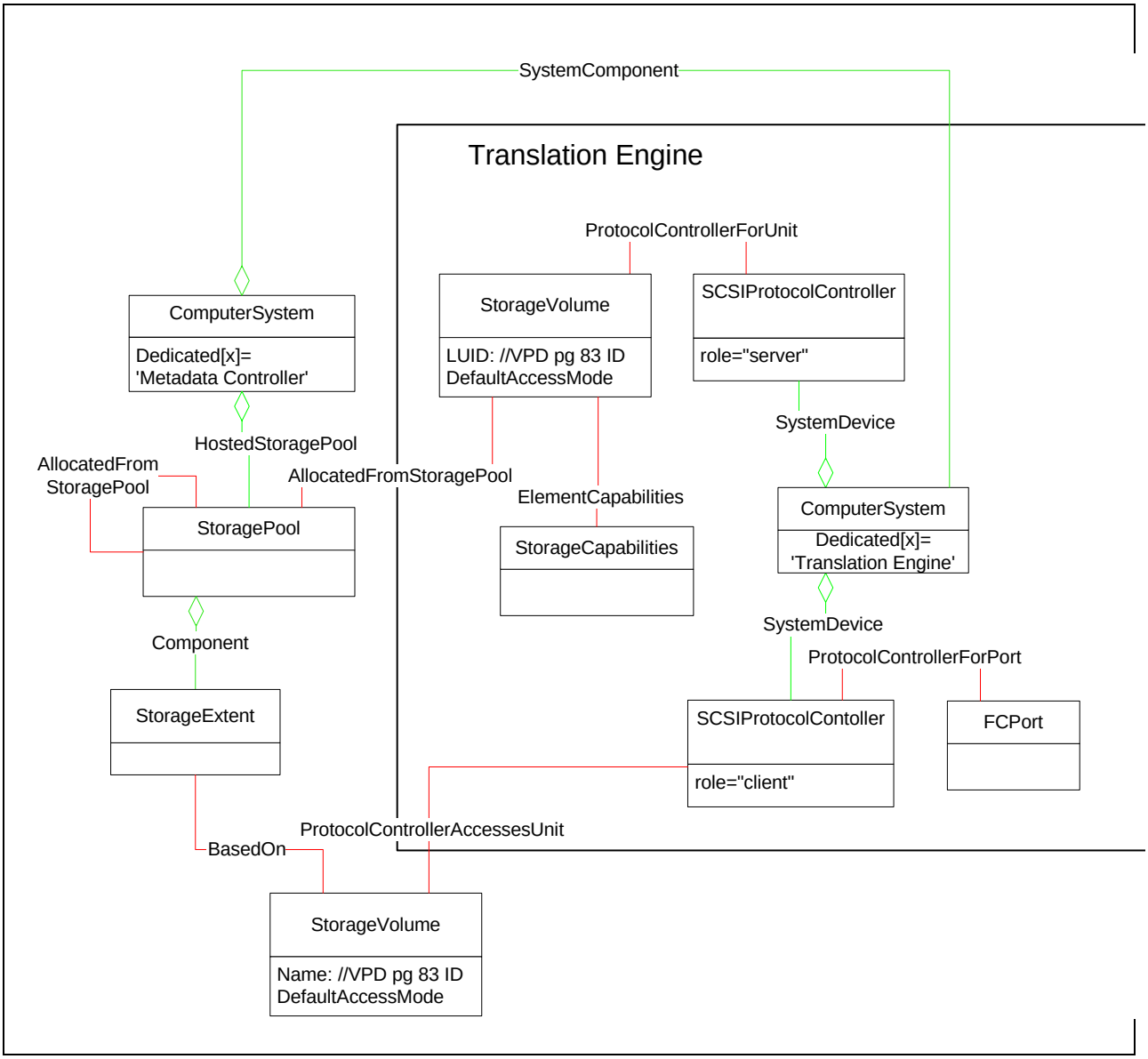


Figure B.13 - Translation Engine Instance Diagram

The ComputerSystem is linked to the two sets of SCSIProtocolControllers. The upper SCSIProtocolController represents the interface to the host. Volumes linked to this SCSIProtocolController by ProtocolControllerForUnit associations represent the storage the translation engine is presenting to the host system. Details of the associations is described by the “LUN Masking and Mapping” subprofile”. The StorageVolume is also linked to the StoragePools they are allocated from. The StoragePool is defined and

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

owned by the metadata Controller. The translation engine uses the pool definition to map array storage to host volumes. Multiple translation engines share the pools owned by the metadata Controller.

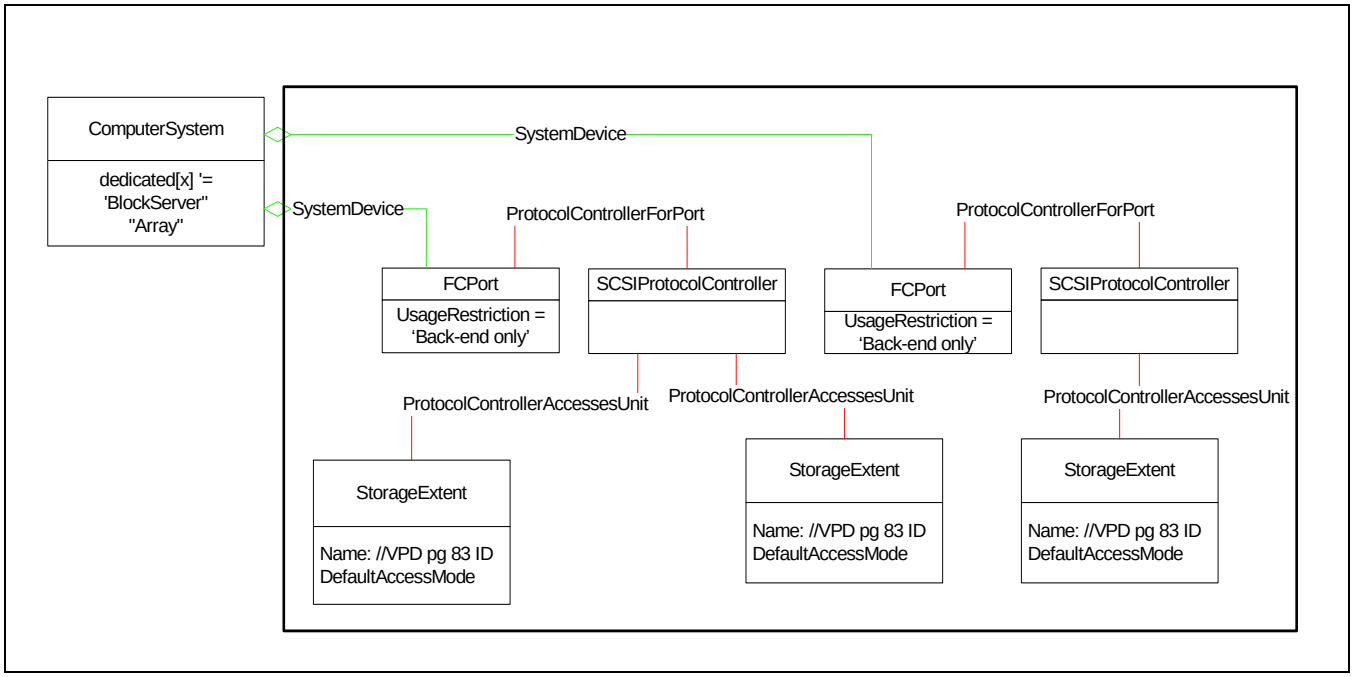


Figure B.14 - Translation Engine Back-end Ports Instance

The Lower SCSIProtocolController with its FCPort class represent the physical port the translation engine uses to access the array's storage. The SystemDevice associations link the ports to the translation engine. The SCSIProtocolController represents the logical SCSI controller that uses the port. The PotocolControllerAccessesUnit association links the SCSIProtocolControllers to the volumes presented by the array. This association contains information about this SCSI connection. Durable names in the volume links this model to the volumes presented by the array model.

B.6.6 Durable names and correlatable IDs of the Profile

B.6.6.1 Durable names exported

The StorageVolume class is used to represent storage that is used by hosts and storage from arrays that are used by the Virtualization system. The StorageVolumes have a durable name to link these objects across profiles.

The FCPort class is used to show both communication and SCSI ports. To model the physical links in a SAN, these objects have a durable name.

B.6.6.2 Correlatable IDs used

None.

B.6.7 Methods

The methods needed by this model are part of the common subprofiles for the services and are described there.

B.6.8 Client considerations

None.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

B.6.9 Recipes

There are no recipes defined for this profile.

B.6.10 Instrumentation requirements

None.

B.6.11 Required CIM Elements

Table B.65 - Required CIM Elements

Profile Classes & Associations	Notes
AllocatedFromStoragePool (B.6.12.1)	
BasedOn (B.6.12.2)	
Component (B.6.12.3)	
ComputerSystem - Metadata Controller (B.6.12.4)	(metadata Controller)
ComputerSystem - Translation Engine (B.6.12.5)	(translation Engine)
ElementCapabilities (B.6.12.6)	
FCPort (B.6.12.7)	
HostedCollection (B.6.12.8)	
HostedStoragePool (B.6.12.9)	
LogicalPortGroup (B.6.12.10)	
MemberOfCollection (B.6.12.11)	
ProtocolControllerForPort (B.6.12.12)	
ProtocolControllerForUnit (B.6.12.13)	
SCSIProtocolController (B.6.12.14)	
StorageCapabilities (B.6.12.15)	
StoragePool (B.6.12.16)	
StorageSetting (B.6.12.17)	
StorageExtent (B.6.12.18)	
StorageVolume (B.6.12.19)	
SystemDevice (B.6.12.20)	(port)
SystemDevice (B.6.12.20)	(volume)
Packages	
Physical Package “package” (7.3.2.2)	
Associated Indications	

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table B.65 - Required CIM Elements (Continued)

Profile Classes & Associations	Notes
Creation/Deletion of a StoragePool	SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_StoragePool SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_StoragePool.
Creation/Deletion of an FCPort	SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_FCPort SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_FCPort .
Creation/Deletion of a ComputerSystem	SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_ComputerSystem SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_ComputerSystem.
Creation/Deletion of a StorageVolume	SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_StorageVolume SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_StorageVolume.
Change in the status of a StoragePool	SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_StoragePool AND SourceInstance.OperationalStatus <> PreviousInstance.OperationalStatus.
Change in the status of a StorageVolume	SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_StorageVolume AND SourceInstance.Operationalstatus <> PreviousInstance.Operationalstatus.
Change in the status of a FCPort	SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_FCPort AND SourceInstance.Operationalstatus <> PreviousInstance.Operationalstatus.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

B.6.12 Required properties for CIM Elements

B.6.12.1 AllocatedFromStoragePool

Table B.66 - Required Properties for AllocatedFromStoragePool

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Antecedent	ref	override	The reference to The StoragePool.
Dependent	ref	override	The reference to the logical element that is the subsidiary element.
SpaceConsumed	uint64		Space Consumed from this Pool (in megabytes).

B.6.12.2 BasedOn

Table B.67 - Required Properties for BasedOn

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Antecedent	ref	override	The reference to the source.
Dependent	ref	override	The reference to the destination.

B.6.12.3 Component

Table B.68 - Required Properties for Component

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
GroupComponent	ref	override, key	The reference to the collection.
PartComponent	ref	override, key	The reference to the member.

B.6.12.4 ComputerSystem - Metadata Controller

Table B.69 - Required Properties for ComputerSystem - Metadata Controller

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
ElementName	string		User Friendly name.
OperationalStatus	uint16		Status of array.
Name	string	key	The identifier for the Array (e.g., IP address or FC world-wide name).
NameFormat	string		The format of the Name property.
Dedicated	int16[ ]	“blockserver”, “metadatacontroller”	The use of this ComputerSystem.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table B.69 - Required Properties for ComputerSystem - Metadata Controller

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
PrimaryOwnerContact	string		Optional.
PrimaryOwnerName	string		Optional.

B.6.12.5 ComputerSystem - Translation Engine

Table B.70 - Required Properties for ComputerSystem - Translation Engine

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
ElementName	string		User Friendly name.
OperationalStatus	uint16		Status of array.
Name	string	key	The identifier for the Array (e.g., IP address or FC world-wide name).
NameFormat	string		The format of the Name property.
Dedicated	int16[ ]	“blockserver”, “translationengine”	The use of this ComputerSystem.
PrimaryOwnerContact	string		Optional.
PrimaryOwnerName	string		Optional.

B.6.12.6 ElementCapabilities

Table B.71 - Required Properties for ElementCapabilities

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
MangedElement	ref	override, key	The reference to the source.
Capabilities	ref	override, key	The reference to the destination.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

B.6.12.7 FCPort

Table B.72 - Required Properties for FCPort

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
ElementName	**string**		User friendly name/caption for port.
OperationalStatus	uint16		Status of device.
DeviceID	string	key	Opaque.
PortType	uint16		FC-GS Port.Type The specific mode currently enabled for the Port. The values: \"N\" = Node Port, \"NL\" = Node Port supporting FC arbitrated loop, \"E\" = Expansion Port connecting fabric elements (for example, FC switches), \"F\" = Fabric (element) Port, \"FL\" = Fabric (element) Port supporting FC arbitrated loop and \"B\" = Bridge Port. PortTypes are defined in ISO/IEC 14165-414. When set to 1 (\"Other\"), the related property OtherPortType contains a string description of the port's type. PortType is defined to force consistent naming of the 'type' property in subclasses and to guarantee unique enum values for all instances of NetworkPort. A range of values, DMTF_Reserved, has been defined that allows subclasses to override and define their specific port types. Vendor Reserved = 16 000..65 535 can be used if the PortType is not one already defined in the above enumerations and a vendor subclass is defined specifying the appropriate “value” and “valuemap”.
PermanentAddress	string		The WWN of the port.
NetworkAddresses[ ]	string		The Fibre Channel address of the port (optional).
Speed	uint64		Speed of zero represents a link not established. 1 Gbit is 1 062 500 000 bit/s 2 Gbit is 2 125 000 000 bit/s 4 Gbit is 4 250 000 000 bit/s) 10 Gbit single channel variants are 10 518 750 000 bit/s 10 Gbit four channel variants are 12 750 000 000 bit/s This is the raw bit rate.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table B.72 - Required Properties for FCPort (Continued)

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
UsageRestriction	uint32		Role of this controller (e.g., Initiator/ Target).

B.6.12.8 HostedCollection

Table B.73 - Required Properties from HostedCollection

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Antecedent	ref	key	The portgroup.
Dependent	ref	key	The scoping system.

B.6.12.9 HostedStoragePool

Table B.74 - Required Properties from HostedStoragePool

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
PartComponent	ref		The storage pool.
GroupComponent	ref		The scoping system.

B.6.12.10 LogicalPortGroup

Table B.75 - Required Properties from LogicalPortGroup

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
InstanceID	key		

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

B.6.12.11 MemberOfCollection

Table B.76 - Required Properties for MemberOfCollection

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Collection	ref	override	The reference to the collection.
Member	ref	override	The reference to the member.

B.6.12.12 ProtocolControllerForPort

Table B.77 - Required Properties from ProtocolControllerForPort

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Antecedent	ref		The SCSIProtocolController for this port.
Dependent	ref		The port.
AccessPriority	unit16		The priority of access through this port for this ProtocolController. This property is OPTIONAL.

B.6.12.13 ProtocolControllerForUnit

Table B.78 - Required Properties from ProtocolControllerForUnit

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Antecedent	ref		The protocol controller.
Dependent	ref		The exposed logical unit.
DeviceNumber	unit16		The Logical Unit number for this Volume through this controller. Formatted as unseparated uppercase hexadecimal digits, with no leading “0x”.

B.6.12.14 SCSIProtocolController

Table B.79 - Required Properties for SCSIProtocolController

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
ElementName	**string**		User friendly name/caption for port.
OperationalStatus	uint16		Status of device.
DeviceID	string	key	Opaque.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table B.79 - Required Properties for SCSIProtocolController (Continued)

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
MaxUnitsControlled	uint32		Maximum number of units controlled by this controller.
ControllerNumber[ ]	string		The number by which the Controller is known in the system.

B.6.12.15 StorageCapabilities

Table B.80 - Required Properties from StorageCapabilities

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
ElementName	string		User friendly name/caption.
ElementType	uint16		Type of element this capability applies to.
InstanceID	uint16	key	
NoSinglePointOfFailure	boolean		
NoSinglePointOfFailureDefault	boolean		
DataRedundancyMin	uint16		
DataRedundancyMax	uint16		
DataRedundancyDefault	uint16		
PackageRedundancyMin	uint16		
PackageRedundancyMax	uint16		
PackageRedundancyDefault	uint16		

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

B.6.12.16 StoragePool

Table B.81 - Required Properties for StoragePool

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
ElementName	string		User Friendly name.
InstanceID	string	key	Opaque identifier.
PoolID	string	req, maxlen(256)	A unique name in the context of the System that identifies this pool.
TotalManagedSpace	uint64		
AvailableManagedSpace	unit64		
Primordial	boolean		Defaults to false, true for the primordial pools.
GetSupportedSizes ()	uint32		Method to get a list of supported sizes for pool or volume creation.
GetSupportedSizeRange ()	uint32		Method to get a range of supported sizes for pool or volume creation.

B.6.12.17 StorageSetting

Table B.82 - Required Properties from StorageSetting

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
ElementName	string	req	User friendly name/caption.
InstanceID	uint16	key, req	
DataRedundancyMin	uint16		
DataRedundancyMax	uint16		
DataRedundancyGoal	uint16		
PackageRedundancyMin	uint16		
PackageRedundancyMax	uint16		
PackageRedundancyGoal	unit16		

B.6.12.18 StorageExtent

Table B.83 - Required Properties for StorageExtent

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Caption	string	maxlen(64)	Short (one line) description.
Description	string		Longer description.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table B.83 - Required Properties for StorageExtent (Continued)

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
ElementName	string		User Friendly name.
InstallDate	datetime		
Name	string	maxlen (256)	
SystemCreationClassName	string	maxlen(256), key	Scoping System's CreationClassName.
SystemName	string	maxlen(256), key	Scoping System's Name.
CreationClassName	string	maxlen(256), key	Name of the concrete subclass.
DeviceID	string	maxlen(64), key	Unique identifying information.
Availability	uint16		
LastErrorCode	uint32		
ErrorDescription	string		
ErrorCleared	boolean		
OtherIdentifyingInfo	string[ ]		
PowerOnHours	uint64		
TotalPowerOnHours	uint64		
IdentifyingDescriptions	string[ ]		
AdditionalAvailability	uint16[ ]		
MaxQuiesceTime	uint64		
DataOrganization	uint16		Type of data organization used.
Purpose	string		Free form string describing the media and/or its use.
Access	uint16		Access describes whether the media is readable (value=1), writable (value=2) or both (value=3). \"Unknown\" (0) and \"Write Once\" (4) can also be defined.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table B.83 - Required Properties for StorageExtent (Continued)

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
ErrorMethodology	string		ErrorMethodology is a free-form string describing the type of error detection and correction supported by this StorageExtent.
BlockSize	uint64		Size in bytes of the blocks that form this StorageExtent. If variable block size, then the maximum block size in bytes should be specified. If the block size is unknown or if a block concept is not valid (for example, for AggregateExtents, Memory or LogicalDisks), enter a 1.
NumberOfBlocks	uint64		Total number of logically contiguous blocks, of size BlockSize, that form this Extent. The total size of the Extent can be calculated by multiplying BlockSize by NumberOfBlocks. If the BlockSize is 1, this property is the total size of the Extent.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table B.83 - Required Properties for StorageExtent (Continued)

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
ConsumableBlocks	uint64		The maximum number of blocks, of size BlockSize, that are available for consumption when layering StorageExtents using the BasedOn association. This property only has meaning when this StorageExtent is an Antecedent reference in a BasedOn relationship. For example, a StorageExtent could be composed of 120 blocks. However, the Extent itself may use 20 blocks for redundancy data. If another StorageExtent is BasedOn this Extent, only 100 blocks would be available to it. This information ('100 blocks is available for consumption') is indicated in the ConsumableBlocks property.
IsBasedOnUnderlyingRedundancy	boolean		True indicates that the underlying StorageExtent(s) participate in a StorageRedundancyGroup.
SequentialAccess	boolean		Boolean set to TRUE if the Storage is sequentially accessed by a MediaAccessDevice. A TapePartition is an example of a sequentially accessed StorageExtent. StorageVolumes, DiskPartitions and LogicalDisks represent randomly accessed Extents.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table B.83 - Required Properties for StorageExtent (Continued)

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
ExtentStatus[ ]	uint16		StorageExtents have additional status information beyond that captured in the OperationalStatus and other properties, inherited from ManagedSystem-Element. This additional information (for example, \"Protection Disabled\", value=9) is captured in the VolumeStatus property.
NoSinglePointOfFailure	boolean		Indicates whether or not there exists no single point of failure.
DataRedundancy	uint16		Number of complete copies of data maintained.
SpindleRedundancy	uint16		How many disk spindles can fail without data loss.
DeltaReservation	uint16		Current value for Delta reservation.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

B.6.12.19 StorageVolume

Table B.84 - Required Properties for StorageVolume

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
ElementName	string		User Friendly name.
Name	string		VPD Page 83 ID.
NameFormat	uint16		Format of Name property.
ExtentStatus	uint16[ ]		Status of volume (Rebuild, spare in use, etc.).
OperationalStatus	unit[ ]		Current general status of volume.
DeviceID	string		Opaque ID.
BlockSize	uint64		Block size of Volume.
NumberOfBlocks	uint64		Number of Blocks (not size of volume is BlockSize* NumberOfBlocks).
IsBasedOnUnderlyingRedundancy	boolean		
NoSinglePointOfFailure	boolean		Current value of StorageSetting.
DataRedundancy	uint16		Current value of StorageSetting.
PackageRedundancy	uint16		Current value of StorageSetting.
DeltaReservation	uint16		Current value of StorageSetting.

B.6.12.20 SystemDevice

Table B.85 - Required Properties for SystemDevice

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
GroupComponent	ref	override	System Reference.
PartComponent	ref	override	LogicalDevice Reference.

B.6.13 Optional Subprofiles

Table B.86 - Optional Profiles or Subprofiles

Name	Notes
Cluster Subprofile (7.3.3.3)	
Extra Capacity Set Subprofile (B.8)	
Access Points subprofile (7.3.3.2)	
Software Subprofile (7.3.3.7)	

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table B.86 - Optional Profiles or Subprofiles (Continued)

Name	Notes
Location subprofile (7.3.3.6)	
Pool Manipulation Capabilities and Settings Subprofile (7.3.3.10)	
Job Control Subprofile (7.3.3.9)	
Copy Services Subprofile (7.3.3.8)	
LUN Masking and Mapping (7.3.3.14)	
LUN Creation Subprofile (7.3.3.11)	

B.6.14 Cross-client considerations

B.6.14.1 Mapping Volumes to disks (FC HBA, Virtualizer, Array)

The HBA provides volumes to host systems. The volumes are based on volumes being served by other devices and finally can be mapped to physical disks. Between the host and the physical disk, storage systems can reorganize the data and add redundancy. This example describes how to find the redundancy and physical disks behind a host volume.

The first step is to locate the host volume in the FC HBA profile. SLP can be used to find the namespace for the HBA. A simple enumeration of volumes locates the volume instance. The durable name for the volume can be determined and used to locate the source. To locate the source SLP can be used to find all “block servers” (arrays, virtualization appliances). The source volume can be found by enumerating the volumes on the block servers and matching the durable name to the host volume. The AllocatedFromStoragePool and BasedOn associations can be used to determine where the volume came from. In the case of a virtualization appliance, the AllocatedFromStoragePool leads to a pool. In turn, all StorageExtents connected to the pool with StoragePoolComponent could contain data from the host volume. Each Storage extent should be followed back to the physical disks it is made of. The pool components are Volumes or StorageExtents. If they are not volumes, they can be traced back using BasedOn associations to a volume, set of volumes or (in the case of an array) to physical drives. The associations lead back to Volumes. Durable names can be used again to find the namespace of the sources and the associations to be used to lead back to their source.

--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

B.6.14.2 Virtualizer and Fabric interaction

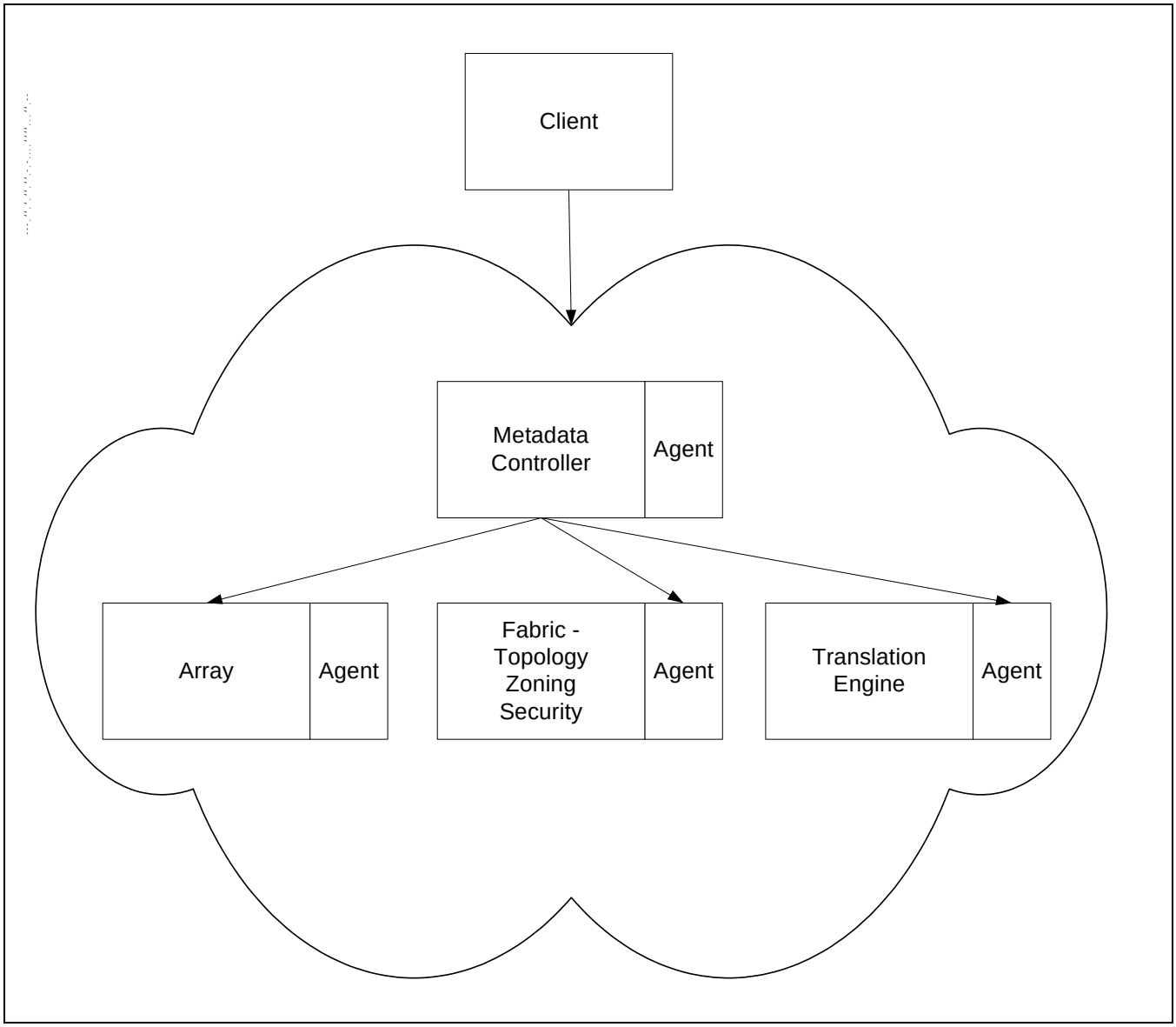


Figure B.15 - Virtualizer/Fabric Interaction Overview

B.6.14.3 Overview

In the case of fabric based out of band virtualization, the Virtualization Appliance may be connected to a fabric using any media (e.g., Fibre Channel or IP) and translation engines may be distributed in the fabric (running on fabric switches). As translation engines come up, they register themselves with a directory service. The Virtualization Appliance can use SLP to lookup this information to discover all the fabric hosted translation engines. The Metadata Controller would typically be hosted on the Virtualization Appliance. Fabric management operations in support of virtualization (such as zoning) can be performed by a management application either through the Metadata Controller or through the fabric. In the case where the management access point is the Metadata Controller, the Virtualization Appliance that hosts the Metadata Controller could register its address with a directory service using SLP and with other mechanisms such as the fabric's platform database.

This subclause describes the role of fabric based zoning and fabric route discovery for virtualization. It also discusses the approaches that can be used to associate objects from different providers and to understand

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

the relationships between the different providers. HBAs, switches and storage devices (e.g., storage arrays) will typically have their own agents supporting their respective profiles and fabric services such as fabric route discovery, zoning and security. These require a client management application to understand the cross provider associations and to traverse through them.

For example, during fabric route discovery it is important for clients to know the logical connectivity between switches, HBAs and storage devices. In the case of zoning, the client may wish to know which HBA and storage device ports or LUNs are zoned together and to be able to set up zones between HBAs and Translation Engines and between Translation Engines and physical storage.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

B.7 JBOD Profile

B.7.1 Description

The JBOD profile is intended to cover simple arrays of disk storage systems that consist only of internal disk drives with a common power and chassis. The key classes are storage volumes (visible to consumers), ports and controllers.

The JBOD model is distinguished from other “Block Server” profiles by the ”JBOD” value in the Dedicated property of ComputerSystem. That is, for JBODs the dedicated values would be “Block Server” and “JBOD”.

B.7.2 Standard dependencies

The JBOD profile is based on the following standards:

Table B.87 - JBOD Standard Dependencies

Standard	Version	Organization
CIM Specification	2.2	DMTF
CIM Operations over HTTP	1.1	DMTF
CIM Schema	2.8 Preliminary	DMTF

B.7.3 Profile dependencies

The JBOD profile requires the Server Profile (7.3.7).

B.7.4 CIM Server requirements

B.7.4.1 Functional Profiles

Table B.88 - Required Functional Profiles

Profile Required	Functional Group	Dependency
YES	Basic Read	None
NO	Basic Write	Basic Read
NO	Instance Manipulation	Basic Write
NO	Schema Manipulation	Instance Manipulation
YES	Association Traversal	Basic Read
NO	Query Execution	Basic Read
NO	Qualifier Declaration	Schema Manipulation
YES	Indication	None

B.7.4.2 Extrinsic methods

The CIM Server should support extrinsic methods for the Server profile.

B.7.4.3 Discovery

The CIM Server should support SLP discovery as defined in the CIM Operations over HTTP specification.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

B.7.5 Instance diagrams

None.

B.7.6 Durable names and correlatable IDs of the Profile

B.7.6.1 Durable names exported

There are no durable names exported by this profile.

B.7.6.2 Correlatable IDs used

There are no correlatable IDs defined for this profile.

B.7.7 Methods

There are no methods defined for this profile.

B.7.8 Client considerations

There are no client considerations defined for this profile.

B.7.9 Recipes

There are no recipes defined for this profile.

B.7.10 Instrumentation requirements

There are no instrumentation requirements defined for this profile.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

B.7.11 Required CIM Elements

Table B.89 - Required CIM Elements

Profile Classes & Associations	Notes
ComputerSystem (B.7.12.1)	
ComputerSystemPackage (B.7.12.2)	
FCPort (B.7.12.3)	
ConcreteIdentity (B.7.12.4)	
ProtocolControllerForUnit (B.7.12.6)	
SCSIProtocolController (B.7.12.5)	(target)
ProtocolControllerForUnit (B.7.12.6)	(port)
SystemDevice (B.7.12.7)	
Packages	
Physical Package “package” (7.3.2.2)	
Associated Indications	
Change in status of ComputerSystem	SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_ComputerSystem AND SourceInstance.OperationalStatus <> PreviousInstance.OperationalStatus.
Creation/Deletion of FC ports	SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_FCPort SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_FCPort.
Creation/Deletion of ProtocolControllerForUnit	SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_ProtocolControllerForUnit SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_ProtocolControllerForUnit.

B.7.12 Required properties for CIM Elemenets

B.7.12.1 ComputerSystem

Table B.90 - Required Properties for ComputerSystem

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Caption	string	maxlen(64)	Short (one line) description.
Description	string		Longer description.
ElementName	string		User Friendly name.
OperationalStatus	uint16		
CreationClassName	string	maxlen(256), key	Name of Class.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table B.90 - Required Properties for ComputerSystem (Continued)

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Name	string	maxlen(256), key	
NameFormat	string	(override “nameformat”)	The ComputerSystem object and its derivatives are Top-level Objects of CIM. They provide the scope for numerous components. Having unique System keys is required. A heuristic is defined to create the ComputerSystem Name to attempt to always generate the same Name, independent of discovery protocol. This prevents inventory and management problems where the same asset or entity is discovered multiple times, but cannot be resolved to a single object. Use of the heuristic is optional, but recommended. The NameFormat property identifies how the ComputerSystem Name is generated, using a heuristic. The heuristic is outlined, in detail, in the CIM V2 System Model specification. It assumes that the documented rules are traversed in order to determine and assign a Name. The NameFormat Values list defines the precedence order for assigning the ComputerSystem Name. Several rules do map to the same Value. Note that the ComputerSystem Name calculated using the heuristic is the System's key value. Other names can be assigned and used for the ComputerSystem, which suit a business better, using Aliases.
OtherIdentifyingInfo	string[ ]		An array of free-form strings providing explanations and details behind the entries in the OtherIdentifying Info array. Note, each entry of this array is related to the entry in OtherIdentifyingInfo that is located at the same index.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table B.90 - Required Properties for ComputerSystem (Continued)

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
IdentifyingDescription	string[ ]		An array of free-form strings providing explanations and details behind the entries in the OtherIdentifying Info array. Note, each entry of this array is related to the entry in OtherIdentifyingInfo that is located at the same index.
Dedicated	int16[ ]	“blockserver”	Enumeration indicating whether the ComputerSystem is a special-purpose System (i.e., dedicated to a particular use), versus being 'general purpose'. For example, one could specify that the System is dedicated to \"Print\" (value=11) or acts as a \"Hub\" (value=8). ||A clarification is needed with respect to the value 17 (\"Mobile User Device\"). An example of a dedicated user device is a mobile phone or bar code scanner in a store that communicates via radio frequency. These systems are quite limited in functionality and programmability, and are not considered 'general purpose' computing platforms. Alternately, an example of a mobile system that is 'general purpose' (i.e., is not dedicated) is a hand-held computer. Although limited in its programmability, new software can be downloaded and its functionality expanded by the user.
OtherDedicatedDescription	string		A string describing how or why the system is dedicated when the Dedicated array includes the value 2, \"Other\".
ResetCapability	uint16		If enabled (value = 4), the ComputerSystem can be reset via hardware (e.g., the power and reset buttons). If disabled (value = 3), hardware reset is not allowed. In addition to Enabled and Disabled, other Values for the property are also defined - \"Not Implemented\" (5), \"Other\" (1) and \"Unknown\" (2).

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

B.7.12.2 ComputerSystemPackage

Table B.91 - Required Properties for ComputerSystemPackage

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Antecedent	ref	key	The reference to the PhysicalPackage(s) that realize a UnitaryComputerSystem.
Dependent	ref	key	The reference to the UnitaryComputerSystem.
PlatformGUID	string		A Gloabally Unique Identifier for the System's Package.

B.7.12.3 FCPort

Table B.92 - Required Properties for FCPort

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
ElementName	**string**		
OperationalStatus	uint16		
DeviceID	string	key, maxlen (64)	
Speed	uint64	units ("bits per second")	Speed of zero represents a link not established. 1 Gbit is 1 062 500 000 bit/s 2 Gbit is 2 125 000 000 bit/s 4 Gbit is 4 250 000 000 bit/s) 10 Gbit single channel variants are 10 518 750 000 bit/s 10 Gbit four channel variants are 12 750 000 000 bit/s This is the raw bit rate.
MaxSpeed	uint64		The maximum speed of the Port in bits per second. FC-FS Port Speed Capabilities.
PortType	uint16	override	
OtherNetworkPortType	string		Describes the type of module, when PortType is set to 1 (\"Other\").
PortNumber	uint64		NetworkPorts are often numbered relative to either a logical module or a network element.
LinkTechnology	uint16		An enumeration of the types of links. When set to 1 (\"Other\"), the related property OtherLinkTechnology contains a string description of the link's type.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table B.92 - Required Properties for FCPort (Continued)

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
OtherLinkTechnology	uint16		A string value describing LinkTechnology when it is set to 1, \"Other\".
PermanentAddress	string	maxlen (64)	PermanentAddress defines the network address hardcoded into a port. This 'hardcoded' address may be changed via firmware upgrade or software configuration. If so, this field should be updated when the change is made. PermanentAddress should be left blank if no 'hardcoded' address exists for the NetworkAdapter.||Port WWN (InfiniBand: Port GUID).
NetworkAddresses[ ]	string	maxlen (64), arraytype ("indexed")	An array of strings indicating the network addresses for the port. FCID (InfiniBand: LIDs) FC-FS Address Identifier.
Speed	uint64	override	Speed of zero represents a link not established. 1 Gbit is 1 062 500 000 bit/s 2 Gbit is 2 125 000 000 bit/s 4 Gbit is 4 250 000 000 bit/s) 10 Gbit single channel variants are 10 518 750 000 bit/s 10 Gbit four channel variants are 12 750 000 000 bit/s This is the raw bit rate.
FullDuplex	boolean		Boolean indicating that the port is operating in full duplex mode.
AutoSense	boolean		A boolean indicating whether the NetworkPort is capable of automatically determining the speed or other communications characteristics of the attached network media.
SupportedMaximumTransmissionUnit	uint64		The maximum transmission unit (MTU) that can be supported."), Units ("Bytes").
ActiveMaximumTransmissionUnit	uint64		The active or negotiated maximum transmission unit (MTU) that can be supported.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table B.92 - Required Properties for FCPort (Continued)

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
PortType	string		FC-GS Port.Type The specific mode currently enabled for the Port. The values: \"N\" = Node Port, \"NL\" = Node Port supporting FC arbitrated loop, \"E\" = Expansion Port connecting fabric elements (for example, FC switches), \"F\" = Fabric (element) Port, \"FL\" = Fabric (element) Port supporting FC arbitrated loop and \"B\" = Bridge Port. PortTypes are defined in ISO/IEC 14165-414. When set to 1 (\"Other\"), the related property OtherPortType contains a string description of the port's type. PortType is defined to force consistent naming of the 'type' property in subclasses and to guarantee unique enum values for all instances of NetworkPort. A range of values, DMTF_Reserved, has been defined that allows subclasses to override and define their specific port types.
SupportedCOS	uint16[ ]		FC-GS Class Of Service An array of integers indicating the Fibre Channel Classes of Service that are supported. The active COS are indicated in ActiveCOS.
ActiveCOS	uint16[ ]		FC-GS Class Of Service An array of integers indicating the Classes of Service that are active. The Active COS is indicated in ActiveCOS.
SupportedFC4Types	uint16[ ]		FC-GS FC4-TYPEs An array of integers indicating the Fibre Channel FC-4 protocols supported. The protocols that are active and running are indicated in the ActiveFC4Types property.
ActiveFC4Types	uint16[ ]		FC-GS FC4-TYPE An array of integers indicating the Fibre Channel FC-4 protocols currently running. A list of all protocols supported is indicated in the SupportedFC4Types property.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

B.7.12.4 ConcreteIdentity

Table B.93 - Required Properties for ConcreteIdentity

Property/ Method	Type	Qualifier/ Parameter	Description/ Notes
SystemElement	ref	key, req	
SameElement	ref	key, req	

B.7.12.5 SCSIProtocolController

Table B.94 - Required Properties for SCSIProtocolController

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Caption	string	maxlen(64)	Short (one line) description.
Description	string		Longer description.
ElementName	string		User Friendly name.
InstallDate	datetime		
OperationalStatus	uint16		
SystemCreationClassName	string	maxlen(256), key	The scoping System's CreationClassName.
SystemName	string	maxlen(256), key	The scoping System's Name.
CreationClassName	string	maxlen(256), key	The name of the concrete subclass.
DeviceID	string	maxlen(64), key	Unique identifying information.
PowerManagementSupported	boolean		
PowerManagementCapabilities	int16[ ]		
Availability	int16		
EnabledState	int16		
LastErrorCode	uint32		
ErrorDescription	string		
ErrorCleared	boolean		
OtherIdentifyingInfo	string[ ]		
PowerOnHours	uint64		
TotalPowerOnHours	uint64		
IdentifyingDescriptions	string[ ]		
AdditionalAvailability	uint16[ ]		

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table B.94 - Required Properties for SCSIProtocolController (Continued)

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
MaxQuiesceTime	uint64		
TimeOfLastReset	datetime		Time of last reset of the Controller.
ProtocolSupported	uint16		The protocol used by the Controller to access 'controlled' Devices.
MaxNumberControlled	uint32		Maximum number of directly addressable entities supported by this Controller. A value of 0 should be used if the number is unknown or unlimited.
ProtocolDescription	string		A free form string providing more information related to the ProtocolSupported by the Controller.
ProtectionManagement	uint16		An integer enumeration indicating whether or not the SCSIProtocolController provides redundancy or protection against device failures.
MaxDataWidth	uint32		Maximum data width (in bits) supported by the SCSIProtocolController.
MaxTransferRate	uint64		Maximum transfer rate (in bits per second) supported by the SCSIProtocolController.
ControllerTimeouts	uint32		Number of SCSIProtocolController timeouts that have occurred since the TimeOfLastReset.
SignalCapabilities[ ]	uint16		Signal capabilities that can be supported by the SCSIProtocolController. For example, the Controller may support \"Single Ended\" and \"Differential\". In this case, the values 3 and 4 would be written to the SignalCapabilities array.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

B.7.12.6 ProtocolControllerForUnit

Table B.95 - Required Properties for ProtocolControllerForUnit

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
NegotiatedSpeed	unit64		
NegotiatedDataWidth	unit32		
Dependent	ref	override	LogicalDevice Reference.
AccessState	unit16		
TimeOfDeviceReset	datetime		
NumberOfHardResets	unit32		
NumberOfSoftResets	unit32		
Antecedent	ref	override	SCSIProtocolController Reference.
DeviceNumber	string		Formatted as unseparated uppercase hexadecimal digits, with no leading “0x”.

B.7.12.7 SystemDevice

Table B.96 - Required Properties for SystemDevice

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
GroupComponent	ref	override	System Reference.
PartComponent	ref	override	LogicalDevice Reference.

B.7.13 Optional Subprofiles

Table B.97 - Optional Profiles or Subprofiles

Name	Notes
Cluster Subprofile (7.3.3.3)	
Physical Package “package” (7.3.2.2)	
Access Points subprofile (7.3.3.2)	
Location subprofile (7.3.3.6)	
Disk Drive subprofile (7.3.3.4)	

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

B.8 Extra Capacity Set Subprofile

B.8.1 Description

NOTE This profile is experimental and is provided for information only. The contents of an experimental profile is likely to change as implementation experience is gained. Please provide any implementation feedback to Storage Management Initiative Technical Steering Group (td@snia.org).

The Extra Capacity Set Subprofile is an optional subprofile for the Array, Out-of-Band Virtualization and In-band Virtualization Profiles.

While the Cluster subprofile defines component computer systems that make up the top-level computer system, it does not specify the relationship among the systems. The Extra Capacity Set subprofile is for defining specific redundancy offered by a collection of computer systems in the configuration.

Some of the component computer systems also typically provide redundancy. This can be of several kinds: Load balancing, fail-over, load balancing/fail-over with redundancy.

- Load balancing.
  This typically means that each path to a LUN through the participating computer systems has equal functionality and priority. This type of redundancy is shown using the ExtraCapacitySet class with the LoadBalancedSet property set to true. Each computer system is associated with a ExtraCapacitySet instance with a MemberOfCollection association. If Controllers operate as redundant pairs then there would be multiple ExtraCapacitySet instances – one for each pair.
- Fail over.
  This typically means that the paths are asymmetrical. One path will be the primary one, the other(s) are for fail-over only and have lower access speed or whatever. In this case, the LoadBalancedSet property is set to false. The priority of different access paths is shown by the AccessPriority property on the ProtocolControllerForUnit associations between the StorageVolume and ProtocolController.
- Load Balancing and Fail over
  This typically means that load balancing is occurring across the ExtraCapacitySet and in the event of a failure another computer system in the set can take over the work of the failing computer system.

B.8.2 Standard dependencies

The Extra Capacity Set subprofile is defined using the CIM Schema 2.7 final. As such, it can be used in profiles at 2.7 and later. It does not require that Profiles be on a later schema. It will operate within profiles that are at the CIM schema 2.7 final or later. The subprofile will operate correctly with CIM Specification 2.2 (or later) and CIM Operations over HTTP 1.1 (or later).

B.8.3 Profile dependencies

The Extra Capacity Set subprofile introduces no Profile dependencies.

B.8.4 CIM Server requirements

For the SMI-S uses of the Extra Capacity Set subprofile, support for Basic Read, Indications and Association Traversal functional profiles shall be supported by the CIM server for the Profile.

The Extra Capacity Set subprofile does not require support for extrinsic methods.

The Extra Capacity Set subprofile is not advertised.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

B.8.5 Instance diagram

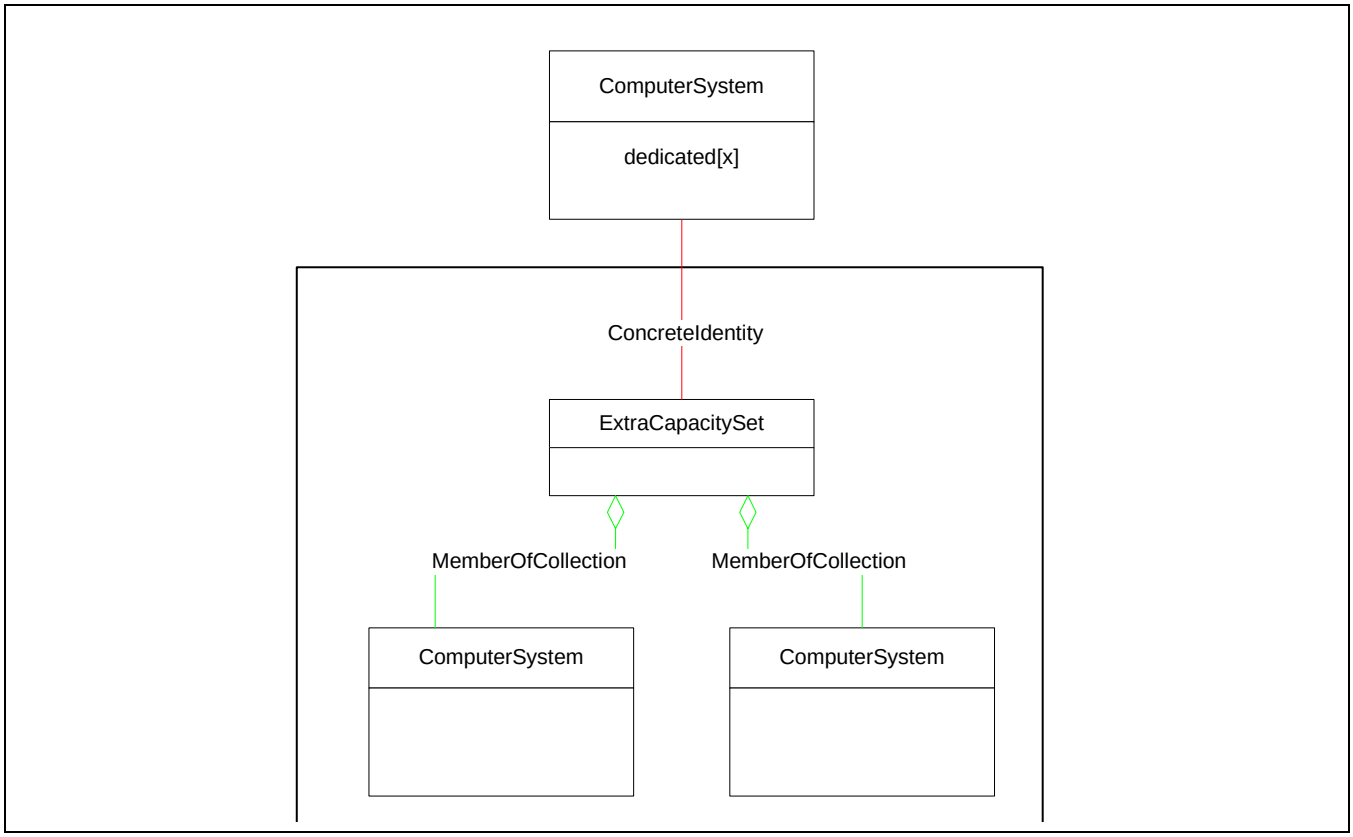


Figure B.16 - Extra Capacity Set Instance Diagram

B.8.6 Durable names and correlatable IDs

The Extra Capacity Set subprofile does not add any durable names or correlatable IDs to the profiles (or subprofiles) in which it is used. While a Durable Name is defined for the top-level ComputerSystem, durable names are not required for the collected computer systems.

B.8.7 Methods

The Extra Capacity Set subprofile is populated by providers and is accessible to clients using basic read and association traversal.

No extrinsics are specified on the Extra Capacity Set subprofile.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

B.8.8 Client considerations

B.8.8.1 Find System Status of a Member Computer System

The ‘OperationStatus’ property is available on most objects in the model and is used to indicate its status. For member computer systems, the ComputerSystem instance should have one of the following Main Operational Statuses and possibly one of the Subsidiary statuses.

Table B.98 - OperationStatus for Component ComputerSystem

Main Operational Status	Possible Subsidiary Operational Status	Description
OK		The computer system has a good status.
OK	Stressed	The computer system is stressed, for example the temperature is over limit or there is too much IO in progress.
OK	Predictive Failure	The computer system will probably fail sometime soon.
Degraded		The computer system is operational but not at 100 % redundancy. A component has suffered a failure or something is running slow.
Error		An error has occurred causing the computer system to stop. This error may be recoverable with operator intervention.
Error	Non-recoverable error	A severe error has occurred. Operator intervention is unlikely to fix it.
Error	Supporting entity in error	A modelled element has failed.
No contact		The provider knows about the computer system but has not talked to it since last reboot.
Lost communication		The provider used to be able to communicate with the computer system, but has now lost contact.
Starting		The computer system is starting up.
Stopping		The computer system is shutting down.
Stopped		The computer system is OK but shut down, the management channel is still working.

A client has the option of subscribing for Asynchronous notification of changes in status through InstModification Indications. More details on indications are in “Events - CIM Indications”.

B.8.9 Recipes

See details in related profile section.

B.8.10 Instrumentation Requirements

See details in related profile section.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

B.8.11 Required CIM Elements

Table B.99 - Required CIM Elements

Profile Classes & Associations	Notes
ComputerSystem (B.8.12.1)	For the processors that make up the ExtraCapacitySet.
ExtraCapacitySet (B.8.12.2)	The collection of processors.
ConcreteIdentity	Associates the ExtraCapacitySet to the “top” level Storage System.
MemberOfCollection (B.8.12.4)	The association that ties the processors in the collection to the ExtraCapacitySet.
Packages	
None	
Methods	
None	
SubProfile Indications	
To indicate the creation of a Component ComputerSystem.	SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_ComputerSystem.
To indicate the deletion of a Component ComputerSystem.	SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_ComputerSystem.
To receive an indication on a change in operational status of a Component ComputerSystem.	SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_ComputerSystem AND SourceInstance.OperationalStatus <> PreviousInstance.OperationalStatus.
To receive an indication on a change to the redundancy status of the ExtraCapacitySet.	SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_ExtraCapacitySet AND SourceInstance.RedundancyStatus <> PreviousInstance.RedundancyStatus.

B.8.12 Required properties for CIM Elements

B.8.12.1 ComputerSystem

The ComputerSystem(s) in the ExtraCapacitySet are processors that support the storage system. They are not the ‘top’ level system.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

ComputerSystem is subclassed from System.

Table B.100 - Required Properties for ComputerSystem

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
ElementName	string		User Friendly name.
OperationalStatus[ ]	uint16		
CreationClassName	string	maxlen(256), key	Name of Class.
Name	string	maxlen(256), key	Not required for the collected computer systems, but may be supplied.
NameFormat	string	override	If supplied, use the same formats as defined for the top-level computer system.
Dedicated[ ]	int16		Since this is a collected computer system, this property is not required. However, if is it used, it should refer to one or more of the values used in the top-level computer system.
OtherDedicatedDescriptions	string		This is not required for SNIA profiles, but it shall be filled in if “other” is specified in the Dedicated array.

B.8.12.2 ExtraCapacitySet

A class that describes that the aggregated elements have more capacity or capability than is needed.

Table B.101 - Required Properties for ExtraCapacitySet

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
InstanceID	string	key	
ElementName	string	override, required	User Friendly name.
RedundancyStatus	uint16		Values {"Unknown", "Other", "Fully Redundant", "Degraded Redundancy", "Redundancy Lost"}.
LoadBalancedSet	boolean		Boolean indicating whether load balancing is supported by the ExtraCapacitySet.

B.8.12.3 ConcreteIdentity

ConcreteIdentity associates two elements representing different aspects of the same underlying entity. ConcreteIdentity is limited in its use as a concrete form of a general identity relationship.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

In the context of the Extra Capacity Set subprofile, this association is used to equate the ExtraCapacitySet instance to the top-level computer system.

ConcreteIdentity is subclassed from LogicalIdentity.

Table B.102 - Required Properties for ConcreteIdentity

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
SystemElement	ref	key	The ManagedElement (e.g., System) that is the basis of the identity (The top-level computer system).
SameElement	ref	key	SameElement represents an alternate aspect of the ManagedElement (System). That is, the ExtraCapacitySet.

B.8.12.4 MemberOfCollection

MemberOfCollection is an aggregation used to establish membership of ManagedElements in a Collection. In the context of the Extra Capacity Set subprofile, this association aggregates the collected computer systems under the ExtraCapacitySet.

MemberOfCollection is not subclassed from anything.

Table B.103 - Required Properties of MemberOfCollection

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Collection	ref	key	The Collection (ExtraCapacitySet) that aggregates members (ComputerSystems).
Member	ref	key	The aggregated member (ComputerSystem) of the Collection.

B.8.13 Optional Subprofiles

Table B.104 - Optional Profiles or Subprofiles

Name	Notes
None	

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

B.9 Host Discovered Resources Profile (EXPERIMENTAL)

B.9.1 Description

NOTE This subprofile is experimental and is provided for information only. The contents of an experimental subprofile is likely to change as implementation experience is gained. Please provide any implementation feedback to Storage Management Initiative Technical Steering Group (td@snia.org).

Among the primary functions of a Fibre Channel Host Bus Adapter (HBA) and its supporting software is discovery of SAN resources and presentation of these resources to the Host Operating System. A description of the results of these functions is useful for some aspects of SAN management, as follows:

- determination of discrepancies between resources discovered by HBAs and the resources provided by other SAN elements is valuable for diagnostics;
- the information discovered by HBAs can provide information about SAN resources not themselves supported by SMI-S agents;
- in SANs that lack an agent for the Fabric profile, e.g., Private Arbitrated Loops and FC Direct Attach, a client can construct a view of the fabric by integrating the discovered resources from any available hosts;
- discovered resource information includes the identification of SAN resources as they are presented to the Host OS.

The Host Discovered Resource agent uses the HBA API Phase 1 to create a generic model of the logical SAN and attached storage. HBA API Phase 1 is included ISO/IEC 14165-511. This agent models elements also exposed by HBA, storage and switch agents. A client can use durable names to equate objects from different agents.

This profile is restricted to FCP (SCSI over FibreChannel) discovery. A similar approach can be used for other protocols (such as IP over FC), but this is not described in this profile. Note that no physical objects are represented by this profile. Since the objects in this profile are discovered remotely through an HBA, only their logical aspects are available. In general, the objects exposed by this agent duplicate those exposed by canonical HBA, storage or switch agents that provide the physical model.

The Host SAN Resources are independently instantiated for each HBA FCPort on a host. They include its discovered (remote) FCP ports and SCSI Targets.

The discovering FCPort and each discovered FCPort are associated by DeviceSAPImplementation to a ProtocolEndpoint representing its FCP support (ProtocolType=other, OtherProtocolType="SCSIOverFC"). An instance of LogicalNetwork is created to aggregate the FCP ProtocolEndpoint for the discovering FCPort and all its discovered FCPorts.

SCSI Targets are modelled by FCPorts with a ProtocolControllerForPort to a SCSIProtocolController that, in turn, has at least one ProtocolControllerForUnit association to a LogicalDevice. The SCSIProtocolController / FCPort combination represents a SCSI Port with Target capability. The LogicalDevice in such associations represent SCSI Target Logical Units.

SCSI Initiators (HBA ports) are also modelled with a SCSIProtocolController / FCPort combination. Initiator SCSIProtocolControllers have ProtocolControllerAccessesUnit associations to logical units (LogicalDevice subclasses) that are mapped to the HBA host.

Target Mappings are a pairing of an OS SCSI ID and an FCPID for a Logical Unit that represents a Logical Unit as presented to a Host Operating System. They are modelled by a LogicalDevice with both a ProtocolControllerForUnit and a ProtocolControllerAccessesUnit association. The OS SCSI ID is represented as attributes of the ProtocolControllerAccessesUnit association. The paired FCPID is derived from the attributes of the ProtocolControllerForUnit association and the FCPort to which it (indirectly) associates.

CIM requires that all LogicalDevices (including SCSIProtocolController and FCPort) be weak to a System via a SystemDevice aggregation. It does not, in general, have means to discover the containing systems for

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

discovered FCPorts, so for each LogicalNetwork, this profile provides an AdminDomain to aggregate discovered objects that shall be weak to a System.

B.9.2 Standard dependencies

The Host Discovered Resources profile is based on the following standards:

Table B.105 - HostDiscoveredResources Standard Dependencies

Standard	Version	Organization
CIM Specification	2.2	DMTF
CIM Operations over HTTP	1.1	DMTF
CIM Schema	2.8 Preliminary	DMTF

B.9.3 Profile dependencies

The Host Discovered Resources profile requires the Server Profile (7.3.7).

B.9.4 CIM Server requirements

B.9.4.1 Functional Profiles

Table B.106 - Required Functional Profiles

Profile Required	Functional Group	Dependency
YES	Basic Read	None
NO	Basic Write	Basic Read
NO	Instance Manipulation	Basic Write
NO	Schema Manipulation	Instance Manipulation
YES	Association Traversal	Basic Read
NO	Query Execution	Basic Read
NO	Qualifier Declaration	Schema Manipulation
YES	Indication	None

B.9.4.2 Extrinsic methods

The CIM Server shall support extrinsic methods for the Server profile.

B.9.4.3 Discovery

The CIM Server shall support SLP discovery as defined in the CIM Operations over HTTP specification.

B.9.5 Instance diagrams

The first instance diagram depicts two logical networks, each contains an HBA and one of two FCPorts in a multi-port array. Three volumes are depicted. Note that volume 2 has no ProtocolControllerAccessesUnit

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

associations, indicating that it is not mapped to the OS hosting this agent. Due to the complexity of this example, some associations are missing and some are unlabeled.

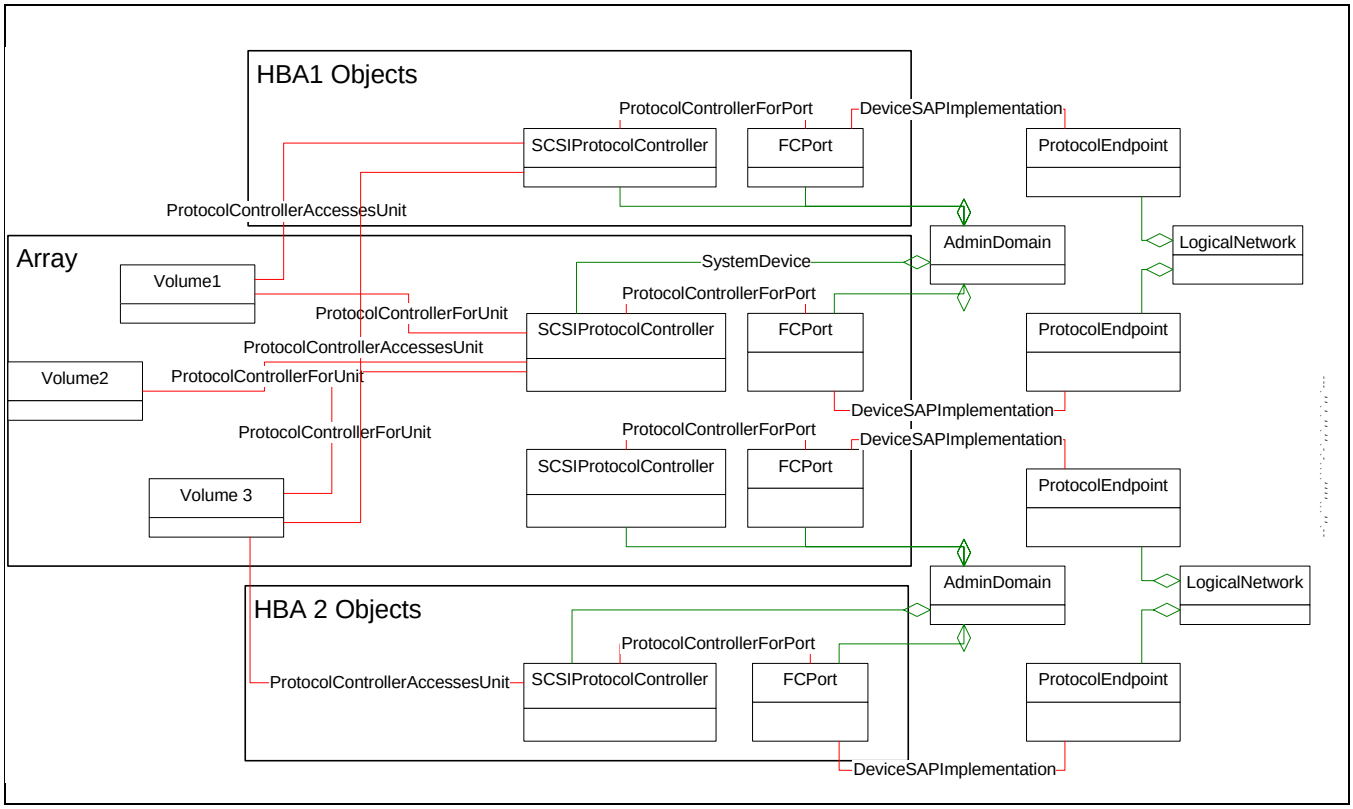


Figure B.17 - Host Discovered Resources Instance Diagram 1

Note that all the depicted objects are instantiated by the Discovered Resources agent. The dashed rectangles represent groups of objects that duplicate objects from other agents. For example, an HBA agent also exposes all the objects in the “HBA1 Objects” rectangle. A client can use durable names to “stitch together” these duplicates.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

The second diagram consists of just a single HBA port, single array port and single volume. All associations are included and labeled.

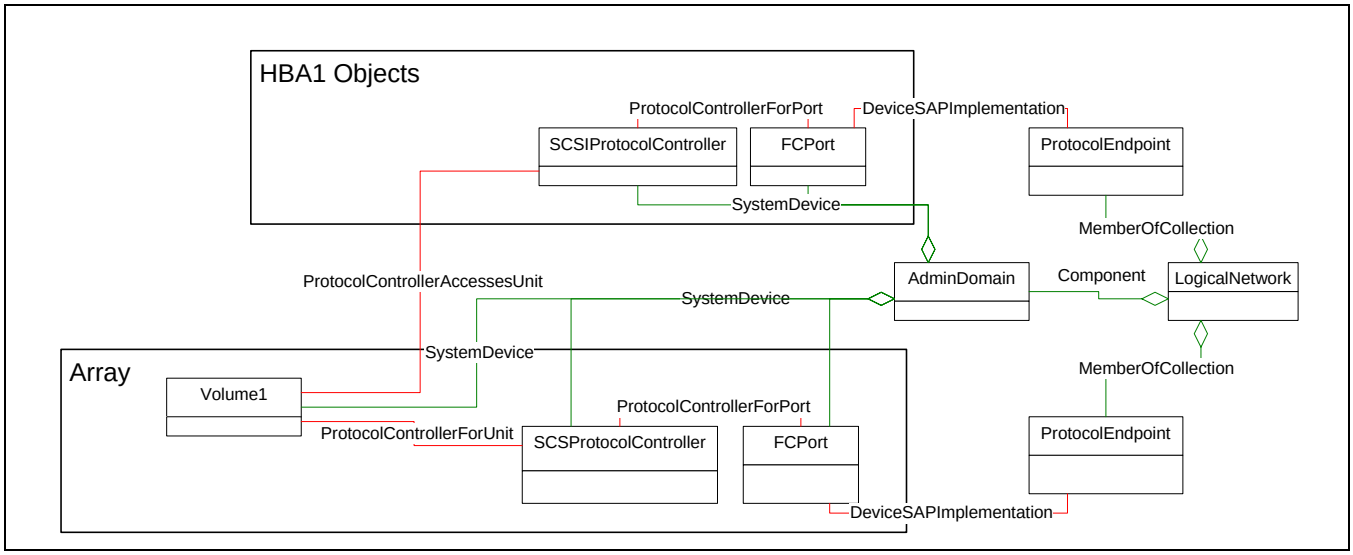


Figure B.18 - Host Discovered Resources Instance Diagram 2

B.9.6 Durable names and correlatable IDs of the Profile

B.9.6.1 Durable names exported

For the Fibre Channel Port, the durable name is the Port WWN in FCPort.PermanentAddress.

B.9.6.2 Correlatable IDs used

The Host Discovered Resources profile uses FC Port WWNs and SCSI Logical Unit (volume or tape drive) IDs.

B.9.7 Methods

There are no methods defined for this profile

B.9.8 Client considerations

In typical configurations, the ports and logical units provided by this model duplicates those found in storage (array or tape library) and switch agents. Although this profile has information about storage systems and the storage network, the information is not complete. For example, this agent may model several small arrays that are actually separate targets (ports) on a single array (or virtual targets resulting from LUN masking/mapping). Where available, the client should use information from an array agent to get a complete model for the array. Similarly, the logical networks modelled by this agent may actually be zones in fabrics. The client should use information from switch agents to get a complete fabric model.

In a non-fabric storage network (Loop or Direct Attached Storage) there is likely to be no agent for the Fabric profile. A client may derive similar information by integrating the models presented by Host Discovered Resource agents running on multiple hosts.

Since the storage system topology cannot be accurately inferred, storage system objects are associated to an AdminDomain, a “virtual” ComputerSystem that represents the collection of objects in the ConnectivityCollection. In particular, SystemDevice associates all logical units to the AdminDomain and the AdminDomain Name property is used as the SystemName property for all LogicalDevice subclasses.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

A client associates objects between profiles using durable identifiers (as described in other profiles). If no storage system agent is available, the model from this agent may suffice, but some details may not be available.

Discovered storage system resources can be partitioned into two groups, objects related to a port (FCPort, SCSIProtocolController and ProtocolEndpoint) and logical units (StorageVolume, TapeDrive and the ProtocolControllerForUnit association). Discovery of logical units can be resource intensive and disruptive to the host system (consider arrays with thousands of logical units). The agent should not allocate resources on logical unit discovery unless requested by a client; this request is communicated by following the ProtocolControllerForUnit associations from a SCSIProtocolController to its logical units. The client algorithm for Discovered Resources for a specific FCPort would be the following.

```
Enumerate AdminDomains
Consider just those with “HBA Discovered Resources” in Roles[ ] and the
                        WWPN of the specific FCPort in Name
Follow the Component association to the LogicalNetwork
For each MemberOfCollection association, follow it to a ProtocolEndpoint
     Follow the DeviceSAPImplementation association to a FCPort
     // This gives the client a list of PortWWNs on the network.
     // If these all map to PortWWNs from array/storage agents, the
     // client may opt to stop probing.
     If the client wishes to also discover LUNs
        Follow the ProtocolControllerForPort Association to a SCSIProtocolController
        Follow each ProtocolControllerForUnit association to logical
                        units
        If no ProtocolControllerForUnit associations are found,
           This is an initiator (another HBA port)
        Else
           Get Instance of LogicalDevices from the
                        ProtocolControllerForUnit association
```

This algorithm allows the agent to dedicate resources to LUN discovery only when requested by a client. Note that if LUNs are not discovered, the model does not include ProtocolControllerForUnit or ProtocolControllerAccessesUnit associations. The client determines target/initiator roles and host/storage system topology by matching durable names with FCPorts in HBA and storage profiles.

A client may discover more complex multipathing by integrating the HBA profiles and Host Discovered Resources profiles from their respective agents, as given in the following examples. If the client found two FCPorts that were SystemDevices of the same ComputerSystem, and found among the Discovered Resources of both, the same FCPort that was associated by ProtocolControllerForPort to a SCSIProtocolController in turn associated by ProtocolControllerForUnit to a LogicalDevice, the client would have demonstrated that the host represented by the ComputerSystem had multipathing via two HBA ports to a single Target port. If it found two FCPorts that were SystemDevices of the same ComputerSystem, and found among the Discovered Resources of each a different FCPort that was associated by ProtocolControllerForPort to a SCSIProtocolController in turn associated by ProtocolControllerForUnit with the same LogicalDevice, the client would have demonstrated that the host represented by the ComputerSystem had multipathing via two HBA ports and two Target ports to a single Target Logical Unit.

B.9.9 Recipes

There are no recipes defined for this profile.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

B.9.10 Instrumentation requirements

The Host SAN Resources profile is based on information available through the HBA API Phase 1 discovery interfaces. Therefore, its implementation may be HBA vendor independent.

The AdminDomain for the LogicalNetwork has no underlying identification. The agent should set the AdminDomain Name property to the Port WWN of the discovering HBA port and the NameFormat property to “FC”. This allows a client to determine which port was used to discover the particular LogicalNetwork. The AdminDomain Roles[ ] array shall contain a “HBA Discovered Resources” entry. This allows the client to determine which AdminDomains are related to this profile.

The agent shall set ProtocolEndpoint properties ProtocolType=other and OtherTypeDescription = ”SCSIOverFC”. The ProtocolEndpoints for the local HBA port and its attached remote ports are all aggregated into a LogicalNetwork. The agent shall set LogicalNetwork properties NetworkType=other, OtherProtocolType=”SCSIOverFC”, Name=discovering FCPort WWN with “:SCSIOverFC” appended and NameFormat=”Discovering FCPort WWN with:SCSIOverFC appended”.

The agent should separate LUN discovery so that a client can limit resources as described in the algorithm under “Client Considerations”, above. In particular, the agent should not issue SCSI “Report LUNs”, “Inquiry”, or “Read Capacity” unless a client follows ProtocolControllerForUnit associations.

If the client asks for ProtocolControllerForUnit associations, LogicalDevice subclasses are instantiated for all the SCSI Logical Units reported by the “Report LUNs” command, then SCSI Inquiry. The agent chooses the LogicalDevice subclass based in the SCSI Inquiry device type.

Table B.107 - SCSI Device Type Mapping

Peripheral Device Type	LogicalDevice Subclass
Direct-access SCSI type	StorageVolume
Sequential-access	TapeDrive
All others	LogicalDevice

Note that this agent cannot determine whether a Direct Access device is a physical disk or a virtualized volume. For consistency the agent always instantiates a StorageVolume. Other than disks and tapes, there are many vendor-specific implementations, so that generic LogicalDevice is instantiated.

SCSI Inquiry VPD commands are issued to get LogicalDevice durable names as described in the array and tape library profiles. These names can be used to identify multi-path configurations. This is modelled with multiple ProtocolControllerForUnit associations from FCPort/ProtocolController pairs to a common LogicalDevice.

If the same logical unit is discovered on multiple LogicalNetworks, the agent can create a single instance and use ProtocolControllerForUnit associations to ProtocolControllers. Logical unit objects can have ProtocolControllerForUnit associations to SCSIProtocolControllers that are associated to different AdminDomains (because they are in different LogicalNetworks). A logical unit object shall be associated to a single AdminDomain. The agent should pick one of the AdminDomains and use it for SystemDevice associations and determination of the SystemName property of the logical unit objects.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

B.9.11 Required CIM Elements

Table B.108 - Required CIM Elements

Profile Classes & Associations	Notes
AdminDomain (7.3.4.1.12.2)	The “virtual ComputerSystem” for LogicalDevice SystemName attributes and SystemDevice associations.
Component (7.3.4.1.12.3)	LogicalNetwork to AdminDomain.
DeviceSAPImplementation (7.3.4.1.12.6)	Associates ProtocolEndpoint and FCPort.
FCPort (7.3.4.1.12.8)	Used on both the initiator and target sides.
HostedCollection (7.3.4.1.12.12)	LogicalPortGroup (Node) to ComputerSystem.
LogicalDevice subclasses	StorageVolume, TapeDrive, MediaAccessDevice depending on SCSI Inquiry responses for whatever is discovered.
LogicalNetwork (7.3.4.1.12.14)	
MemberOfCollection (7.3.4.1.12.15)	ProtocolEndpoint to LogicalNetwork.
ProtocolControllerAccessesUnit (7.3.3.13.12.3)	Associates unit and initiator side protocol controller.
ProtocolControllerForPort (7.3.3.14.12.18)t	FCPort to SCSIProtocolController, if any. Optional, mandatory for FCPorts.
ProtocolControllerForUnit (7.3.3.14.12.17)	Associates unit and target side protocol controller.
ProtocolEndpoint (7.3.4.1.12.16)	Network aspects of an FC Port.
SCSIProtocolController (7.3.3.13.12.4)	Used on both the initiator and target sides.
SystemDevice (7.3.4.1.12.17)	Any LogicalDevice subclass to ComputerSystem (Device FCPort).
Packages	
None.	
Associated Indications	
Creation/Deletion/Modification of StorageVolumes (similar for other logical units)	SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_StorageVolume SELECT * from CIM_InstModification WHERE SourceInstance ISA CIM_StorageVolume SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_StorageVolume.
Creation/Deletion of ports	SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_FCPort SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_FCPort.

B.9.12 Required properties for CIM Elements

The model exposed by this profile is discovered remotely and does not provide canonical information about the elements. Hence, most properties are omitted.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

B.9.12.1 AdminDomain

Table B.109 - Required Properties for AdminDomain

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Caption	string	maxlen(64)	Short (one line) description.
Description	string		Longer description.
ElementName	string		User Friendly name.
OperationalStatus	uint16		
CreationClassName	string	maxlen(256), key	“whatever_AdminDomain”.
Name	string	maxlen(256), key, override	The Port WWN of the discovering port.
NameFormat	string	maxlen(64)	Set to “FC” in this Profile.
PrimaryOwnerName			
PrimaryOwnerContact			
Roles	string[ ]		Should contain an “HBA DIscovered REsources” entry.

B.9.12.2 Component

Table B.110 - Required Properties for Component

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
GroupComponent	ref	key	In this profile, the LogicalNetwork.
PartComponent	ref	key	In this profile, an AdminDomain.

B.9.12.3 DeviceSAPImplementation

Table B.111 - Required Properties for DeviceSAPImplementation

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Antecedent	ref	key	FCPort reference.
Dependent	ref	key	ProtocolEndpoint reference.

B.9.12.4 FCPort

Table B.112 - Required Properties for FCPort

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
ElementName	**string**		

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table B.112 - Required Properties for FCPort (Continued)

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
OperationalStatus	uint16		
DeviceID	string	key, maxlen (64)	
Speed	uint64	units ("bits per second")	Speed of zero represents a link not established. 1 Gbit is 1 062 500 000 bit/s 2 Gbit is 2 125 000 000 bit/s 4 Gbit is 4 250 000 000 bit/s) 10 Gbit single channel variants are 10 518 750 000 bit/s 10 Gbit four channel variants are 12 750 000 000 bit/s This is the raw bit rate.
MaxSpeed	uint64		The maximum speed of the Port in bits per second. FC-FS Port Speed Capabilities.
PortType	uint16	override	FC-GS Port.Type The specific mode currently enabled for the Port. The values: \"N\" = Node Port, \"NL\" = Node Port supporting FC arbitrated loop, \"E\" = Expansion Port connecting fabric elements (for example, FC switches), \"F\" = Fabric (element) Port, \"FL\" = Fabric (element) Port supporting FC arbitrated loop, and \"B\" = Bridge Port. PortTypes are defined in ISO/IEC 14165-414. When set to 1 (\"Other\"), the related property OtherPortType contains a string description of the port's type. PortType is defined to force consistent naming of the 'type' property in subclasses and to guarantee unique enum values for all instances of NetworkPort. A range of values, DMTF_Reserved, has been defined that allows subclasses to override and define their specific port types. Vendor Reserved = 16 000..65 535 can be used if the PortType is not one already defined in the above enumerations and a vendor subclass is defined specifying the appropriate “value” and “valuemap”.
OtherNetworkPortType	string		Describes the type of module, when PortType is set to 1 (\"Other\").

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table B.112 - Required Properties for FCPort (Continued)

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
PortNumber	uint64		NetworkPorts are often numbered relative to either a logical module or a network element.
LinkTechnology	uint16		An enumeration of the types of links. When set to 1 (\"Other\"), the related property OtherLinkTechnology contains a string description of the link's type.
OtherLinkTechnology	uint16		A string value describing LinkTechnology when it is set to 1, \"Other\".
PermanentAddress	string	maxlen (64)	PermanentAddress defines the network address hardcoded into a port. This 'hardcoded' address may be changed via firmware upgrade or software configuration. If so, this field should be updated when the change is made. PermanentAddress should be left blank if no 'hardcoded' address exists for the NetworkAdapter.||Port WWN (InfiniBand: Port GUID).
NetworkAddresses[ ]	string	maxlen (64), arraytype ("indexed")	An array of strings indicating the network addresses for the port. FCID (InfiniBand: LIDs) FC-FS Address Identifier.
Speed	uint64	override	Speed of zero represents a link not established. 1 Gbit is 1 062 500 000 bit/s 2 Gbit is 2 125 000 000 bit/s 4 Gbit is 4 250 000 000 bit/s) 10 Gbit single channel variants are 10 518 750 000 bit/s 10 Gbit four channel variants are 12 750 000 000 bit/s This is the raw bit rate.
FullDuplex	boolean		Boolean indicating that the port is operating in full duplex mode.
AutoSense	boolean		A boolean indicating whether the NetworkPort is capable of automatically determining the speed or other communications' characteristics of the attached network media.
SupportedMaximumTransmissionUnit	uint64		The maximum transmission unit (MTU) that can be supported."), Units ("Bytes").

--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table B.112 - Required Properties for FCPort (Continued)

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
ActiveMaximumTransmissionUnit	uint64		The active or negotiated maximum transmission unit (MTU) that can be supported.
PortType	uint16		FC-GS Port.Type The specific mode currently enabled for the Port. The values: \"N\" = Node Port, \"NL\" = Node Port supporting FC arbitrated loop, \"E\" = Expansion Port connecting fabric elements (for example, FC switches), \"F\" = Fabric (element) Port, \"FL\" = Fabric (element) Port supporting FC arbitrated loop and \"B\" = Bridge Port. PortTypes are defined in ISO/IEC 14165-414. When set to 1 (\"Other\"), the related property OtherPortType contains a string description of the port's type. PortType is defined to force consistent naming of the 'type' property in subclasses and to guarantee unique enum values for all instances of NetworkPort. A range of values, DMTF_Reserved, has been defined that allows subclasses to override and define their specific port types. Vendor Reserved = 16 000..65 535 can be used if the PortType is not one already defined in the above enumerations and a vendor subclass is defined specifying the appropriate “value” and “valuemap”.
SupportedCOS	uint16[ ]		FC-GS Class Of Service An array of integers indicating the Fibre Channel Classes of Service that are supported. The active COS are indicated in ActiveCOS.
ActiveCOS	uint16[ ]		FC-GS Class Of Service An array of integers indicating the Classes of Service that are active. The Active COS is indicated in ActiveCOS.
SupportedFC4Types	uint16[ ]		FC-GS FC4-TYPEs An array of integers indicating the Fibre Channel FC-4 protocols supported. The protocols that are active and running are indicated in the ActiveFC4Types property.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table B.112 - Required Properties for FCPort (Continued)

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
ActiveFC4Types	uint16[ ]		FC-GS FC4-TYPE An array of integers indicating the Fibre Channel FC-4 protocols currently running. A list of all protocols supported is indicated in the SupportedFC4Types property.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

B.9.12.5 HostedCollection

Table B.113 - Required Properties for HostedCollection

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Antecedent	ref	key, min(1), max(1)	ComputerSystem
Dependent	ref	key, weak	LogicalPortGroup

B.9.12.6 ProtocolControllerForPort

Table B.114 - Required Properties of ProtocolControllerForPort

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Dependent	ref		The Port.
Antecedent	ref		The protocol controller.

B.9.12.7 LogicalNetwork

Table B.115 - Required Properties for LogicalNetwork

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
InstanceName	string		Node Symbolic Name
CollectionID	string	key	Node WWN
SystemCreationClassName	string	propagated, key	
SystemName	string	propagated, key	
Name	string		Discovering FC Port WWN with ":SCSIOverFC" appended
Name Format	string		Should be set to "Discovering FCPort WWN with :SCSIOverFC appended".
NetworkType	string		Should be set to "Other".
OtherProtocolType	string		

B.9.12.8 MemberOfCollection

Table B.116 - Required Properties of MemberOfCollection

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Collection	ref	key	The Collection that aggregates members.
ManagedElement	ref	key	The aggregated member of the Collection.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

B.9.12.9 ProtocolEndpoint

Table B.117 - Requited Properties for ProtocolEndpoint

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Name	string	maxlen (256)	
CreationClassName	string	key, maxlen (256)	
SystemCreationClassName	string	key, maxlen (256)	
SystemName	string	key, maxlen (256)	
NameFormat	string	maxlen (256)	Heuristic that ensures unique name.
ProtocolType	string	maxlen (64)	“Other” for this profile.
OtherTypeDescription	string	maxlen (64)	“SCSIOverFC” for this profile.

B.9.12.10 SystemDevice

Table B.118 - Required Properties for SystemDevice

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
GroupComponent	ref	override	System Reference.
PartComponent	ref	override	LogicalDevice Reference.

B.9.12.11 StorageVolume

Table B.119 - Required Properties for StorageVolume

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Caption	string	maxlen(64)	Short (one line) description.
Description	string		Longer description.
ElementName	string		User Friendly name.
InstallDate	datetime		
Name	string	maxlen (256)	
Status	string	maxlen (10)	
SystemCreationClassName	string	maxlen(256), key	The scoping System's CreationClassName.
SystemName	string	maxlen(256), key	The scoping System's Name.
CreationClassName	string	maxlen(256), key	The name of the concrete subclass

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table B.119 - Required Properties for StorageVolume (Continued)

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
DeviceID	string	maxlen(64), key	Unique identifying information.
Availability	uint16		
LastErrorCode	uint32		
ErrorDescription	string		
ErrorCleared	boolean		
OtherIdentifyingInfo	string[ ]		
PowerOnHours	uint64		
TotalPowerOnHours	uint64		
IdentifyingDescriptions	string[ ]		
AdditionalAvailability	uint16[ ]		
MaxQuiesceTime	uint64		
DataOrganization	uint16		
Purpose	string		
Access	uint16		
ErrorMethodology	string		
BlockSize	uint64		
NumberOfBlocks	uint64		
ConsumableBlocks	uint64		
IsBasedOnUnderlyingRedundancy	boolean		
SequentialAccess	boolean		
ExtentStatus[ ]	uint16		
NoSinglePointOfFailure	boolean		
DataRedundancy	uint16		
SpindleRedundancy	uint16		
DeltaReservation	uint16		

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

B.9.13 Optional Subprofiles

Table B.120 - Optional Profiles or Subprofiles

Name	Notes
Initiator Subprofile (B.9.14)	
Target Subprofile (B.9.15)	

B.9.14 Initiator Subprofile

B.9.14.1 Description

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

B.9.14.2 Standard dependencies

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

B.9.14.3 Profile dependencies

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

B.9.14.4 CIM Server requirements

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

B.9.14.5 Instance diagrams

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

B.9.14.6 Durable names and correlatable IDs

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

B.9.14.7 Methods

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

B.9.14.8 Client considerations

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

B.9.14.9 Recipes

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

B.9.14.10 Instrumentation requirements

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

B.9.14.11 Required CIM Elements

Table B.121 - Required CIM Elements

Profile Classes & Associations	Notes
ProtocolControllerForPort (7.3.3.13.12.2)	Optional; mandatory for FCPorts..
SCSIProtocolController (Initiator) (7.3.3.13.12.4)	SCSI aspects of an FC Port..
ProtocolControllerAccessesUnit (7.3.3.13.12.3)	Initiator ProtocolController to LogicalDevice or subclass that is mapped into host OS.
Packages	
None	
Associated indications	
None	

B.9.14.12 Required Properties for CIM Elements

B.9.14.12.1ProtocolControllerForPort

Table B.122 - Required Properties for ProtocolControllerForPort

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Dependent	ref	override	LogicalDevice Reference.
Antecedent	ref	override	SCSIProtocolController Reference.

B.9.14.12.2SCSIProtocolController (Initiator)

Table B.123 - Required Properties for SCSIProtocolController

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
SystemCreationClassName	string	maxlen(256), key	The scoping System's CreationClassName.
SystemName	string	maxlen(256), key	The scoping System's Name.
CreationClassName	string	maxlen(256), key	The name of the concrete subclass.
DeviceID	string	maxlen(64), key	Port WWN of associated FCPort is commonly used.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

B.9.14.12.3ProtocolControllerAccessesUnit

Table B.124 - Required Properties for ProtocolControllerAccessesUnit

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Dependent	ref		LogicalDevice Reference.
Antecedent	ref		SCSIProtocolController Reference.
TargetID	uint32		The Target ID as exposed to drivers and applications that use the HBA driver.
DeviceNumber	string		The SCSI Logical Unit Number as seen by this initiator. Formatted as unseparated uppercase hexadecimal digits, with no leading “0x”.

B.9.14.13 Optional Subprofiles

Table B.125 - Optional Profiles or Subprofiles

Name	Notes
None	

B.9.15 Target Subprofile

B.9.15.1 Description

No additional information is required to implement this subprofile.

B.9.15.2 Standard dependencies

No additional information is required to implement this subprofile.

B.9.15.3 Profile dependencies

No additional information is required to implement this subprofile.

B.9.15.4 CIM Server requirements

No additional information is required to implement this subprofile.

B.9.15.5 Instance diagrams

No additional information is required to implement this subprofile.

B.9.15.6 Durable names and correlatable IDs

No additional information is required to implement this subprofile.

B.9.15.7 Methods

No additional information is required to implement this subprofile.

B.9.15.8 Client considerations

No additional information is required to implement this subprofile.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

B.9.15.9 Recipes

No additional information is required to implement this subprofile.

B.9.15.10 Instrumentation requirements

No additional information is required to implement this subprofile.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

B.9.15.11 Required CIM Elements

Table B.126 - Required CIM Elements

Profile Classes & Associations	Notes
SCSIProtocolController (7.3.6.1.12.9)	SCSI aspects of an FC Port.
ProtocolControllerForUnit (7.3.6.1.12.8)	Target ProtocolController to LogicalDevice or subclass, if any.
Packages	
None	
Associated Indications	
None	

B.9.15.12 Required properties for CIM_Elements

B.9.15.12.1SCSIProtocolController)

Table B.127 - Required Properties for SCSIProtocolController

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
SystemCreationClassName	string	maxlen(256), key	The scoping System's CreationClassName.
SystemName	string	maxlen(256), key	The scoping System's Name.
CreationClassName	string	maxlen(256), key	The name of the concrete subclass.
DeviceID	string	maxlen(64), key	Port WWN of associated FCPort is commonly used.

B.9.15.13 ProtocolControllerForUnit

Table B.128 - Required Properties for ProtocolControllerForUnit

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Dependent	ref		LogicalDevice.
Antecedent	ref		SCSIProtocolController.
DeviceNumber	string		Formatted as unseparated uppercase hexadecimal digits, with no leading “0x”.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

B.9.15.14 Optional Subprofiles

Table B.129 - Optional Profiles or Subprofiles

Name	Notes
None	

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

B.10 Router Profile

B.10.1 Description

A Router is a device that translates between different types of SCSI buses. The instance diagram shows a system with a parallel SCSI bus and Fibre Channel bus. Devices on the parallel bus are served to the Fibre Channel bus without changing the characteristics of the device.

B.10.2 Standard dependencies

The Router profile is based on the following standards:

Table B.130 - Router Standard Dependencies

Standard	Version	Organization
CIM Specification	2.2	DMTF
CIM Operations over HTTP	1.1	DMTF
CIM Schema	2.8 Preliminary	DMTF

B.10.3 Profile dependencies

The Router profile requires the Server Profile (7.3.7).

B.10.4 CIM Server requirements

B.10.4.1 Functional Profiles

Table B.131 - Required Functional Profiles

Profile Required	Functional Group	Dependency
YES	Basic Read	None
NO	Basic Write	Basic Read
NO	Instance Manipulation	Basic Write
NO	Schema Manipulation	Instance Manipulation
YES	Association Traversal	Basic Read
NO	Query Execution	Basic Read
NO	Qualifier Declaration	Schema Manipulation
YES	Indication	None

B.10.4.2 Extrinsic methods

The CIM Server shall support extrinsic methods for the Server profile.

B.10.4.3 Discovery

The CIM Server shall support SLP discovery as defined in the CIM Operations over HTTP specification.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

B.10.5 Instance diagrams

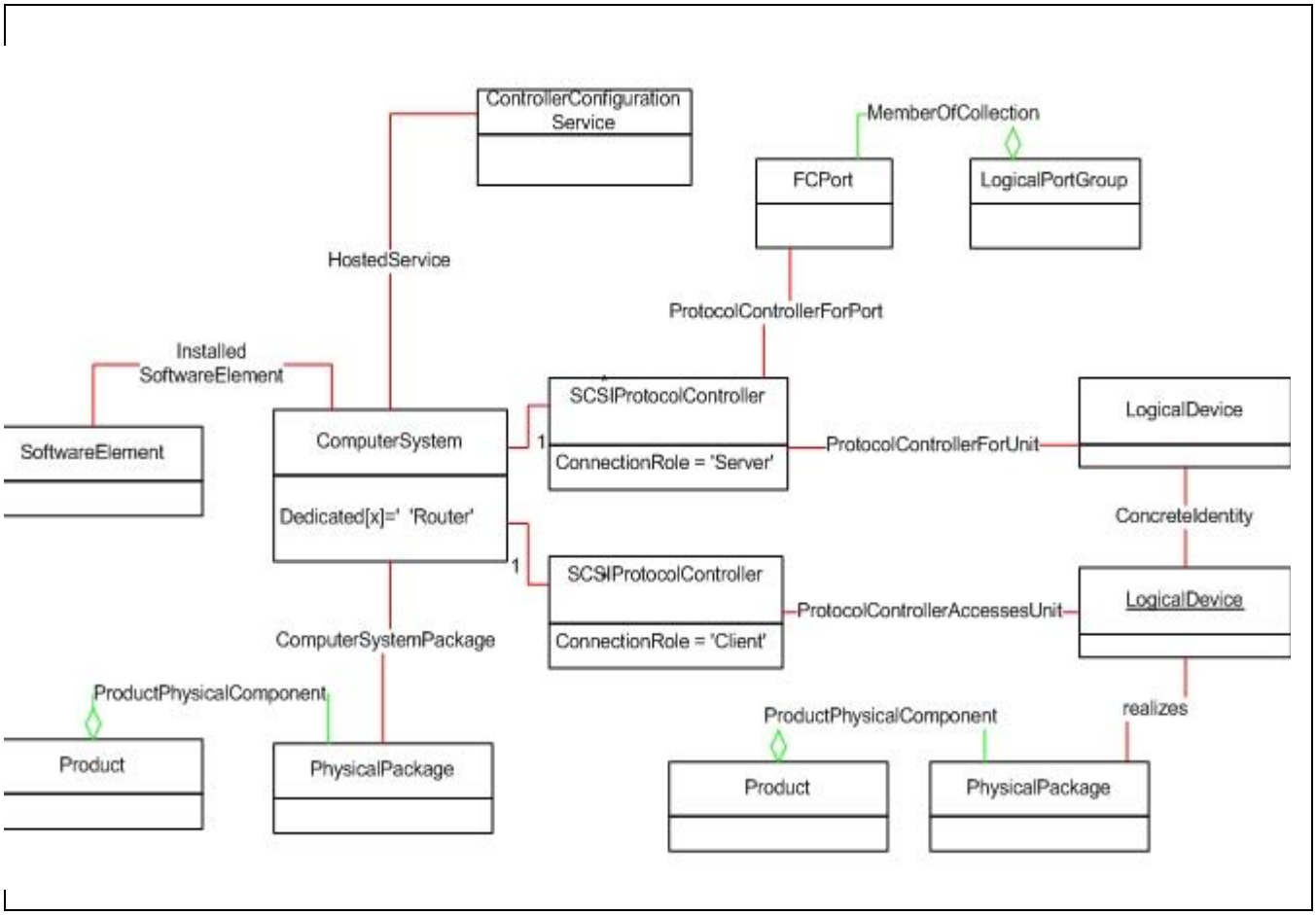


Figure B.19 - Router Instance Diagram

B.10.6 Durable names and correlatable IDs of the Profile

B.10.6.1 Durable names exported

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

B.10.6.2 Correlatable IDs used

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

B.10.7 Methods

None defined for this subprofile in isolation, though there may be particular requirements detailed in the profile that relies on this subprofile.

B.10.8 Client considerations

B.10.8.1 Basic design

The router model consists of 6 major groups of classes (Core, Physical, Software, SCSI buses, source / exported devices).

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

The CompuerSystem class is the core of the model. It is identified as a router by the dedicated attribute being set to “Router”. The PysicalPackage class and PortOnDevice class represent the physical aspects of the router and served devices. These classes contain attributes that can be used to identify the hardware. This information includes serial number, model number and vendor name.

The SoftwareElement class represents the product’s firmware or vendor-specific utilities that are running on the router. This class should be subclassed for each utility.

The SCSIProtocolController class and optionally the FCPort class represent the SCSI buses that are part of the router. The SCSIProtocolController class near the bottom of the instance diagram is the parallel SCSI side of the router. Note that it doesn’t have an association to a FCPort class. It has ProtocolControllerAccessesUnit associations to the devices on the bus. The SCSI addresses of the devices are stored in the association.

The SCSIProtocolController class near to top of the instance diagram has a ConcreteIdentity association to a FCPort class. This indicates the FCPort is a Fibre channel SCSI port. This FC bus connects to the SAN. This bus uses ProtocolControllerForUnit and ProtocolControllerForUnit associations to hold the address mapping and masking. The SCSIProtocolController manages these associations.

A LogicalDevice class represents the device on the back end bus. This class has a Realizes association to a PhysicalPackage class to identify the hardware. The class uses a ConcreteIdentity association to a second instance of LogicalDevice. This class represents the device as seen by the front end port.

B.10.9 Recipes

No recipes have been defined for this profile.

B.10.10 Instrumentation requirements

No implementation requirements have been defined for this profile.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

B.10.11 Required CIM Elements

Table B.132 - Required CIM Elements

Profile Classes & Associations	Notes
ComputerSystem (B.10.12.1)	
ComputerSystemPackage (B.10.12.2)	
FCPort (B.10.12.3)	
LogicalDevice (B.10.12.4)	
ConcreteIdentity (B.10.12.5)	
LogicalPortGroup (B.10.12.6)	
MemberOfCollection (B.10.12.7)	
SCSIProtocolController (B.10.12.8)	
ProtocolControllerAccessesUnit (B.10.12.9)	
ProtocolControllerForUnit (B.10.12.10)	
Packages	
Physical Package “package” (7.3.2.2)	
Associated Indications	
Creation/Deletion of a ComputerSystem	SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_ComputerSystem SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_ComputerSystem.
Creation/Deletion of an FCPort	SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_FCPort SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_FCPort.
Change in status of ComputerSystem	SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_ComputerSystem AND SourceInstance.OperationalStatus <> PreviousInstance.OperationalStatus.
Change in status of FCPort	SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_FCPort AND SourceInstance.OperationalStatus <> PreviousInstance.OperationalStatus.

B.10.12 Required Properties for CIM Elements

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

B.10.12.1 ComputerSystem

Table B.133 - Required Properties for ComputerSystem

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Caption	string	maxlen(64)	Short (one line) description.
Description	string		Longer description.
ElementName	string		User Friendly name.
OperationalStatus	uint16		
CreationClassName	string	maxlen(256), key	Name of Class.
Name	string	maxlen(256), key	
NameFormat	string	(override “nameformat”)	The ComputerSystem object and its derivatives are Top-level Objects of CIM. They provide the scope for numerous components. Having unique System keys is required. A heuristic is defined to create the ComputerSystem Name to attempt to always generate the same Name, independent of discovery protocol. This prevents inventory and management problems where the same asset or entity is discovered multiple times, but cannot be resolved to a single object. Use of the heuristic is optional, but recommended. The NameFormat property identifies how the ComputerSystem Name is generated, using a heuristic. The heuristic is outlined, in detail, in the CIM V2 System Model specification. It assumes that the documented rules are traversed in order, to determine and assign a Name. The NameFormat Values list defines the precedence order for assigning the ComputerSystem Name. Several rules do map to the same Value. Note that the ComputerSystem Name calculated using the heuristic is the System's key value. Other names can be assigned and used for the ComputerSystem, that better suit a business, using Aliases.

--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table B.133 - Required Properties for ComputerSystem (Continued)

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
OtherIdentifyingInfo[ ]	string		An array of free-form strings providing explanations and details behind the entries in the OtherIdentifyingInfo array. Note, each entry of this array is related to the entry in OtherIdentifyingInfo that is located at the same index.
IdentifyingDescription[ ]	string		An array of free-form strings providing explanations and details behind the entries in the OtherIdentifyingInfo array. Note, each entry of this array is related to the entry in OtherIdentifyingInfo that is located at the same index.
Dedicated[ ]	int16	“blockserver”	Enumeration indicating whether the ComputerSystem is a special-purpose System (i.e., dedicated to a particular use), versus being 'general purpose'. For example, one could specify that the System is dedicated to \"Print\" (value=11) or acts as a \"Hub\" (value=8). ||A clarification is needed with respect to the value 17 (\"Mobile User Device\"). An example of a dedicated user device is a mobile phone or bar code scanner in a store that communicates via radio frequency. These systems are quite limited in functionality and programmability and are not considered 'general purpose' computing platforms. Alternately, an example of a mobile system that is 'general purpose' (i.e., is not dedicated) is a hand-held computer. Although limited in its programmability, new software can be downloaded and its functionality expanded by the user.
OtherDedicatedDescription	string		A string describing how or why the system is dedicated when the Dedicated array includes the value 2, \"Other\".

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table B.133 - Required Properties for ComputerSystem (Continued)

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
ResetCapability	uint16		If enabled (value = 4), the ComputerSystem can be reset via hardware (e.g., the power and reset buttons). If disabled (value = 3), hardware reset is not allowed. In addition to Enabled and Disabled, other Values for the property are also defined - \"Not Implemented\" (5), \"Other\" (1) and \"Unknown\" (2).

B.10.12.2 ComputerSystemPackage

Table B.134 - Required Properties for ComputerSystemPackage

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Antecedent	ref		The reference to the PhysicalPackage(s) that realize a UnitaryComputerSystem.
Dependent	ref		The reference to the UnitaryComputerSystem.
PlatformGUID	string		A Globally Unique Identifier for the System's Package.

--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

B.10.12.3 FCPort

Table B.135 - Required Properties for FCPort

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
SystemName	string	key	
SystemCreationClassName	string	key	
CreationClassName	string	key	
ElementName	string		
OperationalStatus	uint16		
DeviceID	string	key, maxlen (64)	
Speed	uint64	units ("bits per second")	Speed of zero represents a link not established 1 Gbit is 1 062 500 000 bit/s 2 Gbit is 2 125 000 000 bit/s 4 Gbit is 4 250 000 000 bit/s) 10 Gbit single channel variants are 10 518 750 000 bit/s 10 Gbit four channel variants are 12 750 000 000 bit/s This is the raw bit rate.
MaxSpeed	uint64		The max speed of the Port in bits per second. FC-FS Port Speed Capabilities.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table B.135 - Required Properties for FCPort (Continued)

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
PortType	uint16	override	FC-GS Port.Type The specific mode currently enabled for the Port. The values: \"N\" = Node Port, \"NL\" = Node Port supporting FC arbitrated loop, \"E\" = Expansion Port connecting fabric elements (for example, FC switches), \"F\" = Fabric (element) Port, \"FL\" = Fabric (element) Port supporting FC arbitrated loop and \"B\" = Bridge Port. PortTypes are defined in ISO/IEC 14165-414. When set to 1 (\"Other\"), the related property OtherPortType contains a string description of the port's type. PortType is defined to force consistent naming of the 'type' property in subclasses and to guarantee unique enum values for all instances of NetworkPort. A range of values, DMTF_Reserved, has been defined that allows subclasses to override and define their specific port types. Vendor Reserved = 16 000..65 535 can be used if the PortType is not one already defined in the above enumerations and a vendor subclass is defined specifying the appropriate “value” and “valuemap”.
OtherNetworkPortType	string		Describes the type of module, when PortType is set to 1 (\"Other\").
PortNumber	uint16		NetworkPorts are often numbered relative to either a logical modules or a network element.
LinkTechnology	uint16		An enumeration of the types of links. When set to 1 (\"Other\"), the related property OtherLinkTechnology contains a string description of the link's type.
OtherLinkTechnology	uint16		A string value describing LinkTechnology when it is set to 1, \"Other\".

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table B.135 - Required Properties for FCPort (Continued)

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
PermanentAddress	string	maxlen (64)	PermanentAddress defines the network address hard-coded into a port. This hard-coded address may be changed via firmware upgrade or software configuration. If so, this field should be updated when the change is made. PermanentAddress should be left blank if no hard-coded address exists for the NetworkAdapter. Port WWN (InfiniBand: Port GUID).
NetworkAddresses[ ]	string	maxlen (64), arraytype ("indexed")	An array of strings indicating the network addresses for the port. FCID (InfiniBand: LIDs) FC-FS Address Identifier.
Speed	uint64	override	Speed of zero represents a link not established. 1 Gbit is 1 062 500 000 bit/s 2 Gbit is 2 125 000 000 bit/s 4 Gbit is 4 250 000 000 bit/s) 10 Gbit single channel variants are 10 518 750 000 bit/s 10 Gbit four channel variants are 12 750 000 000 bit/s This is the raw bit rate.
FullDuplex	boolean		Boolean indicating that the port is operating in full duplex mode.
AutoSense	boolean		A boolean indicating whether the NetworkPort is capable of automatically determining the speed or other communications characteristics of the attached network media.
SupportedMaximumTransmissionUnit	uint64		The maximum transmission unit (MTU) that can be supported."), Units ("Bytes").
ActiveMaximumTransmissionUnit	uint64		The active or negotiated maximum transmission unit (MTU) that can be supported.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table B.135 - Required Properties for FCPort (Continued)

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
PortType	uint16		FC-GS Port.Type The specific mode currently enabled for the Port. The values: \"N\" = Node Port, \"NL\" = Node Port supporting FC arbitrated loop, \"E\" = Expansion Port connecting fabric elements (for example, FC switches), \"F\" = Fabric (element) Port, \"FL\" = Fabric (element) Port supporting FC arbitrated loop, and \"B\" = Bridge Port. PortTypes are defined in ISO/IEC 14165-414. When set to 1 (\"Other\"), the related property OtherPortType contains a string description of the port's type. PortType is defined to force consistent naming of the 'type' property in subclasses and to guarantee unique enum values for all instances of NetworkPort. A range of values, DMTF_Reserved, has been defined that allows subclasses to override and define their specific port types. Vendor Reserved = 16 000..65 535 can be used if the PortType is not one already defined in the above enumerations and a vendor subclass is defined specifying the appropriate “value” and “valuemap”.
SupportedCOS	uint16[ ]		FC-GS Class Of Service An array of integers indicating the Fibre Channel Classes of Service that are supported. The active COS are indicated in ActiveCOS.
ActiveCOS	uint16[ ]		FC-GS Class Of Service An array of integers indicating the Classes of Service that are active. The Active COS is indicated in ActiveCOS.
SupportedFC4Types	uint16[ ]		FC-GS FC4-TYPEs An array of integers indicating the Fibre Channel FC-4 protocols supported. The protocols that are active and running are indicated in the ActiveFC4Types property.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table B.135 - Required Properties for FCPort (Continued)

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
ActiveFC4Types	uint16[ ]		FC-GS FC4-TYPE An array of integers indicating the Fibre Channel FC-4 protocols currently running. A list of all protocols supported is indicated in the SupportedFC4Types property.
UsageRestriction	uint16		Front-end only.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

B.10.12.4 LogicalDevice

Table B.136 - Required Properties for LogicalDevice

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Antecedent	ref	override	The physical component that implements the Device.
Dependent	ref	override	The LogicalDevice.

B.10.12.5 ConcreteIdentity

Table B.137 - Required Properties for ConcreteIdentity

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Antecedent	ref	override	
Dependent	ref	override	

B.10.12.6 LogicalPortGroup

Table B.138 - Required Properties for LogicalPortGroup

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
InstanceName	string		Node Symbolic Name.
SystemCreationClassName	string	propagated, key	
SystemName	string	propagated, key	
InstanceID	string	key	Node WWN||FC-GS InterconnectElement.Name, REQUIRED.

B.10.12.7 MemberOfCollection

Table B.139 - Required Properties of MemberOfCollection

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Collection	ref	key	The Collection that aggregates members.
ManagedElement	ref	key	The aggregated member of the Collection.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

B.10.12.8 SCSIProtocolController

Table B.140 - Required Properties for SCSIProtocolController

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Caption	string	maxlen(64)	Short (one line) description.
Description	string		Longer description.
ElementName	string		User Friendly name.
InstallDate	datetime		
OperationalStatus	uint16		
SystemCreationClassName	string	maxlen(256), key	The scoping System's CreationClassName.
SystemName	string	maxlen(256), key	The scoping System's Name.
CreationClassName	string	maxlen(256), key	The name of the concrete subclass.
DeviceID	string	maxlen(64), key	Unique identifying information.
PowerManagementSupported	boolean		
PowerManagementCapabilities	uint16[ ]		
Availability	uint16		
StatusInfo	uint16		
LastErrorCode	uint32		
ErrorDescription	string		
ErrorCleared	boolean		
OtherIdentifyingInfo	string[ ]		
PowerOnHours	uint64		
TotalPowerOnHours	uint64		
IdentifyingDescriptions	string[ ]		
AdditionalAvailability	uint16[ ]		
MaxQuiesceTime	uint64		
PortNumber	uint64		System level port or bus identification number.
TimeOfLastReset	datetime		Time of last reset of the Controller.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table B.140 - Required Properties for SCSIProtocolController (Continued)

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
ProtocolSupported	uint16		The protocol used by the Controller to access 'controlled' Devices.
MaxNumberControlled	uint32		Maximum number of directly addressable entities supported by this Controller. A value of 0 should be used if the number is unknown or unlimited.
ProtocolDescription	string		A free form string providing more information related to the ProtocolSupported by the Controller.
ProtectionManagement	uint16		An integer enumeration indicating whether or not the SCSIProtocolController provides redundancy or protection against device failures.
MaxDataWidth	uint32		Maximum data width (in bits) supported by the SCSIProtocolController.
MaxTransferRate	uint64		Maximum transfer rate (in bits per second) supported by the SCSIProtocolController.
ControllerTimeouts	uint32		Number of SCSIProtocolController timeouts that have occurred since the TimeOfLastReset.
SignalCapabilities[ ]	uint16		Signal capabilities that can be supported by the SCSIProtocolController. For example, the Controller may support \"Single Ended\" and \"Differential\". In this case, the values 3 and 4 would be written to the SignalCapabilities array.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

B.10.12.9 ProtocolControllerAccessesUnit

Table B.141 - Required Properties for ProtocolControllerAccessesUnit

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
NegotiatedSpeed	uint64		
NegotiatedDataWidth	uint32		
Dependent	ref	override	LogicalDevice Reference.
AccessState	uint16		
TimeOfDeviceReset	datetime		
NumberOfHardResets	uint32		
NumberOfSoftResets	uint32		
Antecedent	ref	override	SCSIProtocolController Reference.
SCSITimeouts	uint32		
SCSIRetries	uint32		
InitiatorId	uint32		
TargetId	uint32		
TargetLUN	uint64		
SCSIReservation	uint16		
SCSISignal	uint16		
MaxQueueDepth	uint32		
QueueDepthLimit	uint32		

B.10.12.10 ProtocolControllerForUnit

Table B.142 - Required Properties for ProtocolControllerForUnit

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
NegotiatedSpeed	uint64		
NegotiatedDataWidth	uint32		
Dependent	ref	override	LogicalDevice Reference.
AccessState	uint16		
TimeOfDeviceReset	datetime		
NumberOfHardResets	uint32		
NumberOfSoftResets	uint32		

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table B.142 - Required Properties for ProtocolControllerForUnit (Continued)

Property/ Method	Type	Qualifier/ Parameter	Description/Notes
Antecedent	ref	override	SCSIProtocolController Reference.
DeviceNumber	string		Formatted as unseparated uppercase hexadecimal digits, with no leading “0x”.

B.10.13 Optional Subprofiles

Table B.143 - Optional Profiles or Subprofiles

Name	Notes
Software Subprofile (7.3.3.7)	
Backend Ports subprofile (7.3.3.13)	
LUN Mapping/Masking	

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Annex C
(informative)

Mapping CIM Objects to SNMP MIB Structures

C.1 Purpose of this Annex

In order to encourage adoption of the WBEM initiative, its associated data model (CIM), protocol (xmlCIM), and profiles (described in previous clauses of this standard), the Storage Media Library (SML) workgroup defined a means of mapping CIM objects to SNMP MIB objects, or "fields." When this annex was written, SNMP (Simple Network Management Protocol) was the dominant non-proprietary network management protocol used by the storage devices described above. This "CIM-to-MIB" mapping methodology has been successfully used by members of SNIA-SML to demonstrate, at minimal cost in development time, WBEM-based interoperability in "plugfests" and industry demonstrations such as Storage Networking World.

Details of the SML workgroups' "CIM-to-MIB" mapping methodology are included in this standard in order to

- demonstrate that a standard path of backward compatibility is obtainable between WBEM and SNMP-based management paradigms,
- document one successful method of CIM-to-MIB mapping,
- recommend this method as *the* standard CIM-to-MIB mapping method in order to avoid a proliferation of deviant "de facto standards", and
- allow SNIA member companies outside the SML workgroup to benefit from earlier experience and work.

C.2 CIM-to-MIB Mapping Overview

CIM is an object-based modelling schema that supports all common object-oriented principles, including abstract class objects, instance objects, inheritance, single- and multiple-association, aggregation, properties, methods and qualifiers. In contrast, SNMP's ASN.1-based modelling schema is strictly hierarchical, involving such structures as nested parent and child nodes and scalar and tabular fields. While unique CIM objects are typically referenced by parent class name (or "Creation Class Name") and key properties, SNMP "objects" – actually singleton "fields" – are typically referenced by an Object Identifier (OID) that points to their position in the SNMP Management Information Base (MIB) hierarchy or "tree." (In the case of tabular fields, additional indexes are appended to a base OID to identify unique instances of information.)

The task of any CIM-to-MIB mapping methodology is primarily to create a one-to-one mapping between object-oriented information and tree-based hierarchical information. Naming constraints within the CIM and MIB domains shall also be adhered to in a way that prevents ambiguities in uniquely identifying and referencing information, particularly in the SNMP/MIB domain.

Therefore, the mapping methodology described here provides the following:

- a description of mapping CIM data – classes, instances, properties, associations – into an SNMP format involving nodes, fields, and tables;
- a naming convention in the SNMP/MIB domain that allows for unambiguous identification of the original CIM data;
- a data type mapping that allows common CIM data to be represented by existing ASN.1 data types.

C.3 CIM-to-MIB mapping methodology

C.3.1 General

Each subclause below specifies a method for mapping a particular CIM structure into an equivalent SNMP MIB structure. In each case, an SNMP/MIB domain naming convention is also called out. Finally, a select set

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

of data type mappings between CIM and ASN.1 is made. Examples in each subsection refer to the draft SML example MIB at the end of this annex.

C.3.2 Mapping CIM child classes to SNMP nodes

SNMP nodes or groups can be used to CIM child classes. In this methodology, the names for such nodes are a concatenation of the CIM class name and the word “Group.” For example, the CIM Chassis class would be represented in a MIB under a node named chassisGroup.

C.3.3 Mapping CIM properties to SNMP scalar fields

Scalar SNMP fields can be used to model CIM properties. In this methodology, the names for such fields are a concatenation of the SNMP node name, a dash (“-”) and the CIM property name. This mapping method applies to both inherited and non-inherited properties. For example, the Name property inherited by a StorageLibrary (sub)class from ManagedSystemElement would be represented in a MIB under the storageLibraryGroup and would therefore be called storageLibrary-Name.

C.3.4 Mapping multiple instances of CIM classes to SNMP tables

SNMP tables can be used to model multiple instances of CIM classes. For example, there could be several instances of MediaAccessDevice in a real implementation of the storage profile. In this methodology, these instances are grouped together in an SNMP table. Such tables then consist of the following parts: a Table node, an Entry node, an Index field, an optional ObjectType field and tabular fields representing properties from CIM classes. A “number of” field is also associated with all tables to allow applications to quickly determine the number of entries in a table and therefore determine the number of CIM instances modelled. See the SNIA-SML MIB in C.4 for examples.

C.3.5 Grouping of multiple subclasses into a single SNMP table

In some cases, a CIM parent class has several children, each of which has few or no unique properties. MediaAccessDevice is a good example of this case. In order to keep a MIB reasonably sized, the multiple subclasses of MediaAccessDevice could be grouped (in a MIB) under a single mediaAccessTypeGroup in a single mediaAccessDeviceTable. The non-CIM ObjectType field in the MIB is then used to specify which subclass is being represented by a particular row in this table. See the SNIA-SML MIB in C.4 for examples.

C.3.6 Mapping CIM data types into SNMP types and textual conventions

Several data types defined in CIM MOFs have been mapped to SNMP data types and textual conventions. The following mappings between CIM data types and SNMP data types have been made.

Table C.1 - CIM/SNMP Data Type Mapping

CIM data type	SNMP data type
string	DisplayString.
boolean	INTEGER enumeration.
datetime	OCTET STRING (using CimDataType below).
enumerated values	INTEGER enumeration.
uint64	OCTET STRING (using UINT64 below).
uint32	INTEGER (using UINT32 below).
uint64	INTEGER (using UINT16 below).
uint8	INTEGER (unranged).

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table C.1 - CIM/SNMP Data Type Mapping (Continued)

CIM data type	SNMP data type
real32	INTEGER (using UShortReal below).

The following ASN.1 "textual conventions" provide additional detail on CIM data types mapped to SNMP data types:

```
UShortReal ::= INTEGER (0..'ffff'h)
-- This textual convention can be used to represent short
-- unsigned 'real' numbers. Using this variable type,
-- a 3 digit number with 2 decimal places (xxx.xx)
-- can be represented. For example, 321.45 would be
-- represented as 32145"

CimDateTime ::= OCTET STRING (SIZE (24))
-- This textual convention can be used to represent a date
-- and time using the CIM DateTime convention. The bytes are
-- as follows:

-- octets contents range
-- ====== ======== =====
-- 1-4 year 0000-9999
-- 5-6 month 01-12
-- 7-8 day 01-31
-- 9-10 hour 00-23
-- 11-12 minute 00-59
-- 13-14 second 00-59
-- 15-20 microseconds 000000-999999
-- 21 sign '+' or '-'
-- 22-24 UTC offset in minutes 000-839

-- For example, Monday, May 25, 1998, at 1:30:15 PM EST would be
-- represented as 19980525133015000000-300

-- Note that values shall be zero-padded so that the entire
-- string is always the same 25-character length. Fields that
-- are not significant can be replaced with asterisk characters"

UINT64 ::= OCTET STRING (SIZE (8))
-- This textual convention can be used to represent 64-bit
-- numbers using the OCTET STRING type. SNMPv2 supports a
-- Counter64 type, but there is no C-language mapping for a
-- 64-bit variable that's much better than an array of 8 bytes

UINT32 ::= INTEGER (0..'7fffffff'h)
UINT16 ::= INTEGER (0..'ffff'h)
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

For CIM data types not listed here, no specific mappings have been made to SNMP for the Minimal MIB. See the SNIA-SML MIB in C.4 for examples.

C.3.7 Mapping CIM associations to SNMP index-keyed fields

Associations between CIM classes can be explicitly modelled in a MIB using table indexes as reference points. For example, several instances of MediaAccessDevices can be supported in a MIB, as can several instances of SoftwareElement. Both of these classes are represented by tables in a MIB, and both tables have their own indexes. A CIM association between SoftwareElement and MediaAccessDevice, called DeviceSoftware, is modelled in a MIB by a field named "SoftwareElement-DeviceSoftware-LogicalDeviceAssociationId". This MIB field holds the SNMP table index of the MediaAccessDevice with which an instance of SoftwareElement is associated. Note that this association is modelled as being unidirectional: *from* SoftwareElement *to* MediaAccessDevice, not vice versa. Users of this methodology have the option of modelling unidirectional or bidirectional associations in this way. See the SNIA-SML MIB in C.4 for examples.

C.3.8 Implied CIM associations and aggregations not in the MIB

In most cases, CIM associations and aggregations are *not* explicitly modelled as SNMP MIB structures. Instead, these “implied” associations and aggregations are expected to be “filled in” by a CIM provider supporting that profile (and MIB). For example, in the Tape Library profile, multiple classes are connected with the CIM StorageLibrary class. In CIM, StorageLibrary, which is a child class of System, is connected to several LogicalDevices (like MediaAccessDevice) via an association called SystemDevice. The SystemDevice association would not have to be explicitly modelled in a MIB, though it is included in a CIM profile. A provider supporting this profile should return valid information about instances of SystemDevice even though there are no MIB fields describing the association. In short, most CIM classes modelled in a MIB are assumed to be connected to one other, by implicit associations or aggregations, because they all “exist” on the same SNMP agent. See the SNIA-SML MIB in C.4 for examples.

C.3.9 Selective mapping of inherited CIM properties

As has been mentioned above, CIM child classes that are modelled in a MIB do not, in most cases, inherit all properties or associations from their CIM parent classes. When mapping a CIM profile to an SNMP MIB, no specifications or restrictions are made as to which properties are inherited by particular child classes (in the MIB representation) and which are not. Users of this methodology should include in their MIB only those CIM properties that are deemed critical to providers and clients.

C.4 Example mapping

The following example illustrates the SNIA-SML MIB generated using this CIM-to-MIB mapping methodology.

```
-- SML MIB Rev 1.11
SML-MIB
     DEFINITIONS ::= BEGIN

     IMPORTS
        OBJECT-TYPE
           FROM RFC-1212
        enterprises
           FROM RFC1155-SMI
        DisplayString
           FROM RFC1213-MIB
        ;
```

--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
-- Textual Conventions

UShortReal ::= INTEGER (0..'ffff'h)
-- This textual convention can be used to represent short
-- unsigned 'real' numbers. Using this variable type,
-- a 3 digit number with 2 decimal places (xxx.xx)
-- can be represented. For example, 321.45 would be
-- represented as 32145"

CimDateTime ::= OCTET STRING (SIZE (24))
-- This textual convention can be used to represent a date
-- and time using the CIM DateTime convention. The bytes are
-- as follows:

--   octetscontentsrange
--   ===================
--   1-4 year  0000-9999
--   5-6 month 01-12
--   7-8 day   01-31
--   9-10 hour 00-23
--   11-12 minute 00-59
--   13-14 second 00-59
--   15-20 microseconds000000-999999
--   21 sign   '+' or '-'
--   22-24 UTC offset in minutes000-839

-- For example, Monday, May 25, 1998, at 1:30:15 PM EST would be
-- represented as 19980525133015000000-300

-- Note that values shall be zero-padded so that the entire
-- string is always the same 25-character length. Fields that
-- are not significant can be replaced with asterisk characters"

UINT64 ::= OCTET STRING (SIZE (8))
-- This textual convention can be used to represent 64-bit
-- numbers using the OCTET STRING type. SNMPv2 supports a
-- Counter64 type, but there is no C-language mapping for a
-- 64-bit variable that's much better than an array of 8 bytes

UINT32 ::= INTEGER (0..'7fffffff'h)
UINT16 ::= INTEGER (0..'ffff'h)
-- MIB Fields

smlRoot OBJECT IDENTIFIER ::= { experimental 202 }

smlMibVersion OBJECT-TYPE
     SYNTAX DisplayString (SIZE (0..4))
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
            "Description here"
        ::= { smlRoot 1 }

    smlCimVersion OBJECT-TYPE
        SYNTAX DisplayString (SIZE (0..4))
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
            "Description here"
        ::= { smlRoot 2 }

    productGroup
        OBJECT IDENTIFIER
        ::= { smlRoot 3 }

    product-Name OBJECT-TYPE
        SYNTAX DisplayString (SIZE (0..255))
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
            "Description here"
        ::= { productGroup 1 }

    product-IdentifyingNumber OBJECT-TYPE
        SYNTAX DisplayString (SIZE (0..255))
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
            "Description here"
        ::= { productGroup 2 }

    product-Vendor OBJECT-TYPE
        SYNTAX DisplayString (SIZE (0..255))
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
            "Description here"
        ::= { productGroup 3 }

    product-Version OBJECT-TYPE
        SYNTAX DisplayString (SIZE (0..255))
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
```

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
            "Description here"
        ::= { productGroup 4 }

    chassisGroup
        OBJECT IDENTIFIER

        ::= { smlRoot 4 }

    chassis-Manufacturer OBJECT-TYPE
        SYNTAX DisplayString (SIZE (0..255))
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
            "Description here"
        ::= { chassisGroup 1 }

    chassis-Model OBJECT-TYPE
        SYNTAX DisplayString (SIZE (0..64))
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
            "Description here"
        ::= { chassisGroup 2 }

    chassis-SerialNumber OBJECT-TYPE
        SYNTAX DisplayString (SIZE (0..64))
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
            "Description here"
        ::= { chassisGroup 3 }

    chassis-LockPresent OBJECT-TYPE
        SYNTAX INTEGER {
                    unknown (0),
                    true (1),
                    false (2) }
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
            "Description here"
        ::= { chassisGroup 4 }

    chassis-SecurityBreach OBJECT-TYPE
        SYNTAX INTEGER {
                    unknown (0),
                    other (1),
```

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
                noBreach (2),
                breachAttempted (3) }
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
            "Description here"
        ::= { chassisGroup 5 }

chassis-IsLocked OBJECT-TYPE
        SYNTAX INTEGER {
                unknown (0),
                true (1),
                false (2) }
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
            "Description here"
        ::= { chassisGroup 6 }

storageLibraryGroup
        OBJECT IDENTIFIER
        ::= { smlRoot 5 }

storageLibrary-Name OBJECT-TYPE
        SYNTAX DisplayString (SIZE (0..255))
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
            "Description here"
        ::= { storageLibraryGroup 1 }

storageLibrary-Description OBJECT-TYPE
        SYNTAX DisplayString (SIZE (0..255))
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
            "Description here"
        ::= { storageLibraryGroup 2 }

storageLibrary-Caption OBJECT-TYPE
        SYNTAX DisplayString (SIZE (0..64))
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
            "Description here"
        ::= { storageLibraryGroup 3 }
```

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
storageLibrary-Status OBJECT-TYPE
     SYNTAX DisplayString (SIZE (0..10))
     ACCESS read-only
     STATUS mandatory
     DESCRIPTION
        "Description here"
     ::= { storageLibraryGroup 4 }

storageLibrary-InstallDate OBJECT-TYPE
     SYNTAX CimDateTime
     ACCESS read-only
     STATUS mandatory
     DESCRIPTION
        "Description here"
     ::= { storageLibraryGroup 5 }

mediaAccessDeviceGroup
     OBJECT IDENTIFIER
     ::= { smlRoot 6 }

numberOfMediaAccessDevices OBJECT-TYPE
     SYNTAX INTEGER
     ACCESS read-only
     STATUS mandatory
     DESCRIPTION
        "Description here"
     ::= { mediaAccessDeviceGroup 1 }

mediaAccessDeviceTable OBJECT-TYPE
     SYNTAX SEQUENCE OF TableInfo-1
     ACCESS not-accessible
     STATUS mandatory
     DESCRIPTION
        "Description here"
     ::= { mediaAccessDeviceGroup 2 }

mediaAccessDeviceEntry OBJECT-TYPE
     SYNTAX TableInfo-1
     ACCESS not-accessible
     STATUS mandatory
     DESCRIPTION
        "Description here"
     INDEX { mediaAccessDeviceIndex }
     ::= { mediaAccessDeviceTable 1 }

TableInfo-1 ::= SEQUENCE
{
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
     mediaAccessDeviceIndex INTEGER,
     mediaAccessDeviceObjectType INTEGER,
     mediaAccessDevice-Name DisplayString,
     mediaAccessDevice-Status DisplayString,
     mediaAccessDevice-Availability INTEGER,
     mediaAccessDevice-NeedsCleaning INTEGER
}

mediaAccessDeviceIndex OBJECT-TYPE
     SYNTAX INTEGER
     ACCESS read-only
     STATUS mandatory
     DESCRIPTION
        "Description here"
     ::= { mediaAccessDeviceEntry 1 }

mediaAccessDeviceObjectType OBJECT-TYPE
     SYNTAX INTEGER {
               unknown (0),
               wormDrive (1),
               magnetoOpticalDrive (2),
               tapeDrive (3),
               dvdDrive (4),
               cdromDrive (5) }
     ACCESS read-only
     STATUS mandatory
     DESCRIPTION
        "Description here"
     ::= { mediaAccessDeviceEntry 2 }

mediaAccessDevice-Name OBJECT-TYPE
     SYNTAX DisplayString (SIZE (0..255))
     ACCESS read-only
     STATUS mandatory
     DESCRIPTION
        "Description here"
     ::= { mediaAccessDeviceEntry 3 }

mediaAccessDevice-Status OBJECT-TYPE
     SYNTAX DisplayString (SIZE (0..10))
     ACCESS read-only
     STATUS mandatory
     DESCRIPTION
        "Description here"
     ::= { mediaAccessDeviceEntry 4 }

mediaAccessDevice-Availability OBJECT-TYPE
```

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
        SYNTAX INTEGER {
                  unknown (0),
                  other (1),
                  runningFullPower (2),
                  warning (3),
                  inTest (4),
                  notApplicable (5),
                  powerOff (6),
                  offLine (7),
                  offDuty (8),
                  degraded (9),
                  notInstalled (10),
                  installError (11),
                  powerSaveUnknown (12),
                  powerSaveLowPowerMode (13),
                  powerSaveStandby (14),
                  powerCycle (15),
                  powerSaveWarning (16),
                  paused (17),
                  notReady (18) }
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
           "Description here"
        ::= { mediaAccessDeviceEntry 5 }

    mediaAccessDevice-NeedsCleaning OBJECT-TYPE
        SYNTAX INTEGER {
                  unknown (0),
                  true (1),
                  false (2) }
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
           "Description here"
        ::= { mediaAccessDeviceEntry 6 }

    physicalMediaGroup
        OBJECT IDENTIFIER
        ::= { smlRoot 7 }

    numberOfPhysicalMedias OBJECT-TYPE
        SYNTAX INTEGER
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
           "Description here"
```

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
     ::= { physicalMediaGroup 1 }

physicalMediaTable OBJECT-TYPE
     SYNTAX SEQUENCE OF TableInfo-2
     ACCESS not-accessible
     STATUS mandatory
     DESCRIPTION
        "Description here"
     ::= { physicalMediaGroup 2 }

physicalMediaEntry OBJECT-TYPE
     SYNTAX TableInfo-2
     ACCESS not-accessible
     STATUS mandatory
     DESCRIPTION
        "Description here"
     INDEX { physicalMediaIndex }
     ::= { physicalMediaTable 1 }

TableInfo-2 ::= SEQUENCE
{
     physicalMediaIndex INTEGER,
     physicalMediaObjectType INTEGER,
     physicalMedia-Removable INTEGER,
     physicalMedia-Replaceable INTEGER,
     physicalMedia-HotSwappable INTEGER,
     physicalMedia-Capacity UINT64,
     physicalMedia-MediaType INTEGER,
     physicalMedia-MediaDescription DisplayString,
     physicalMedia-CleanerMedia INTEGER,
     physicalMedia-DualSided INTEGER,
     physicalMedia-PhysicalLabel DisplayString
}

physicalMediaIndex OBJECT-TYPE
     SYNTAX INTEGER
     ACCESS read-only
     STATUS mandatory
     DESCRIPTION
        "Description here"
     ::= { physicalMediaEntry 1 }

physicalMediaObjectType OBJECT-TYPE
     SYNTAX INTEGER {
               tape (0),
               other (1) }
     ACCESS read-only
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
        STATUS mandatory
        DESCRIPTION
           "Description here"
        ::= { physicalMediaEntry 2 }

    physicalMedia-Removable OBJECT-TYPE
        SYNTAX INTEGER {
                  unknown (0),
                  true (1),
                  false (2) }
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
           "Description here"
        ::= { physicalMediaEntry 3 }

    physicalMedia-Replaceable OBJECT-TYPE
        SYNTAX INTEGER {
                  unknown (0),
                  true (1),
                  false (2) }
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
           "Description here"
        ::= { physicalMediaEntry 4 }

    physicalMedia-HotSwappable OBJECT-TYPE
        SYNTAX INTEGER {
                  unknown (0),
                  true (1),
                  false (2) }
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
           "Description here"
        ::= { physicalMediaEntry 5 }

    physicalMedia-Capacity OBJECT-TYPE
        SYNTAX UINT64
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
           "Description here"
        ::= { physicalMediaEntry 6 }

    physicalMedia-MediaType OBJECT-TYPE
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
SYNTAX INTEGER {
          unknown (0),
          other (1),
          tape (2),
          qic (3),
          ait (4),
          dtf (5),
          dat (6),
          eightmmTape (7),
          nineteenmmTape (8),
          dlt (9),
          halfInchMO (10),
          catridgeDisk (11),
          jazDisk (12),
          zipDisk (13),
          syQuestDisk (14),
          winchesterDisk (15),
          cdRom (16),
          cdRomXA (17),
          cdI (18),
          cdRecordable (19),
          dvd (20),
          dvdRWPlus (21),
          dvdRAM (22),
          dvdROM (23),
          dvdVideo (24),
          divx (25),
          cdRW (26),
          cdDA (27),
          cdPlus (28),
          dvdRecordable (29),
          dvdRW (30),
          dvdAudio (31),
          dvd5 (32),
          dvd9 (33),
          dvd10 (34),
          dvd18 (35),
          moRewriteable (36),
          moWriteOnce (37),
          moLIMDOW (38),
          phaseChangeWO (39),
          phaseChangeRewriteable (40),
          phaseChangeDualRewriteable (41),
          ablativeWriteOnce (42),
          nearField (43),
          miniQic (44),
          travan (45),
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
                eightmmMetal (46),
                eightmmAdvanced (47),
                nctp (48),
                ltoUltrium (49),
                ltoAccelis (50),
                tape9Track (51),
                tape18Track (52),
                tape36Track (53),
                magstar3590 (54),
                magstarMP (55),
                d2Tape (56),
                dstSmall (57),
                dstMedium (58),
                dstLarge (59) }
     ACCESS read-only
     STATUS mandatory
     DESCRIPTION
        "Description here"
     ::= { physicalMediaEntry 7 }

physicalMedia-MediaDescription OBJECT-TYPE
     SYNTAX DisplayString (SIZE (0..255))
     ACCESS read-only
     STATUS mandatory
     DESCRIPTION
        "Description here"
     ::= { physicalMediaEntry 8 }

physicalMedia-CleanerMedia OBJECT-TYPE
     SYNTAX INTEGER {
                unknown (0),
                true (1),
                false (2) }
     ACCESS read-only
     STATUS mandatory
     DESCRIPTION
        "Description here"
     ::= { physicalMediaEntry 9 }

physicalMedia-DualSided OBJECT-TYPE
     SYNTAX INTEGER {
                unknown (0),
                true (1),
                false (2) }
     ACCESS read-only
     STATUS mandatory
     DESCRIPTION
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
        "Description here"
    ::= { physicalMediaEntry 10 }

physicalMedia-PhysicalLabel OBJECT-TYPE
    SYNTAX DisplayString (SIZE (0..255))
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "Description here"
    ::= { physicalMediaEntry 11 }

physicalPackageGroup
    OBJECT IDENTIFIER
    ::= { smlRoot 8 }

numberOfPhysicalPackages OBJECT-TYPE
    SYNTAX INTEGER
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "Description here"
    ::= { physicalPackageGroup 1 }

physicalPackageTable OBJECT-TYPE
    SYNTAX SEQUENCE OF TableInfo-3
    ACCESS not-accessible
    STATUS mandatory
    DESCRIPTION
        "Description here"
    ::= { physicalPackageGroup 2 }

physicalPackageEntry OBJECT-TYPE
    SYNTAX TableInfo-3
    ACCESS not-accessible
    STATUS mandatory
    DESCRIPTION
        "Description here"
    INDEX { physicalPackageIndex }
    ::= { physicalPackageTable 1 }

TableInfo-3 ::= SEQUENCE
{
    physicalPackageIndex INTEGER,
    physicalPackage-Manufacturer DisplayString,
    physicalPackage-Model DisplayString,
    physicalPackage-SerialNumber DisplayString,
    physicalPackage-Realizes-MediaAccessDeviceIndex INTEGER
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
}

physicalPackageIndex OBJECT-TYPE
     SYNTAX INTEGER
     ACCESS read-only
     STATUS mandatory
     DESCRIPTION
        "Description here"
     ::= { physicalPackageEntry 1 }

physicalPackage-Manufacturer OBJECT-TYPE
     SYNTAX DisplayString (SIZE (0..255))
     ACCESS read-only
     STATUS mandatory
     DESCRIPTION
        "Description here"
     ::= { physicalPackageEntry 2 }

physicalPackage-Model OBJECT-TYPE
     SYNTAX DisplayString (SIZE (0..64))
     ACCESS read-only
     STATUS mandatory
     DESCRIPTION
        "Description here"
     ::= { physicalPackageEntry 3 }

physicalPackage-SerialNumber OBJECT-TYPE
     SYNTAX DisplayString (SIZE (0..64))
     ACCESS read-only
     STATUS mandatory
     DESCRIPTION
        "Description here"
     ::= { physicalPackageEntry 4 }

physicalPackage-Realizes-MediaAccessDeviceIndex OBJECT-TYPE
     SYNTAX INTEGER
     ACCESS read-only
     STATUS mandatory
     DESCRIPTION
        "Description here"
     ::= { physicalPackageEntry 5 }

softwareElementGroup
     OBJECT IDENTIFIER
     ::= { smlRoot 9 }

numberOfSoftwareElements OBJECT-TYPE
```

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
     SYNTAX INTEGER
     ACCESS read-only
     STATUS mandatory
     DESCRIPTION
        "Description here"
     ::= { softwareElementGroup 1 }

softwareElementTable OBJECT-TYPE
     SYNTAX SEQUENCE OF TableInfo-4
     ACCESS not-accessible
     STATUS mandatory
     DESCRIPTION
        "Description here"
     ::= { softwareElementGroup 2 }

softwareElementEntry OBJECT-TYPE
     SYNTAX TableInfo-4
     ACCESS not-accessible
     STATUS mandatory
     DESCRIPTION
        "Description here"
     INDEX { softwareElementIndex }
     ::= { softwareElementTable 1 }

TableInfo-4 ::= SEQUENCE
{
     softwareElementIndex INTEGER,
     softwareElement-Name DisplayString,
     softwareElement-Version DisplayString,
     softwareElement-SoftwareElementID DisplayString,
     softwareElement-Manufacturer DisplayString,
     softwareElement-BuildNumber DisplayString,
     softwareElement-SerialNumber DisplayString,
     softwareElement-CodeSet DisplayString,
     softwareElement-IdentificationCode DisplayString,
     softwareElement-LanguageEdition DisplayString,
     softwareElement-Associations OBJECT IDENTIFIER,
     softwareElement-DeviceSoftware-LogicalDeviceAssociation-ObjectType
                              INTEGER,
     softwareElement-DeviceSoftware-LogicalDeviceAssociationId INTEGER
}

softwareElementIndex OBJECT-TYPE
     SYNTAX INTEGER
     ACCESS read-only
     STATUS mandatory
     DESCRIPTION
        "Description here"
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
        ::= { softwareElementEntry 1 }

    softwareElement-Name OBJECT-TYPE
        SYNTAX DisplayString (SIZE (0..255))
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
            "Description here"
        ::= { softwareElementEntry 2 }

    softwareElement-Version OBJECT-TYPE
        SYNTAX DisplayString (SIZE (0..255))
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
            "Description here"
        ::= { softwareElementEntry 3 }

    softwareElement-SoftwareElementID OBJECT-TYPE
        SYNTAX DisplayString (SIZE (0..255))
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
            "Description here"
        ::= { softwareElementEntry 4 }

    softwareElement-Manufacturer OBJECT-TYPE
        SYNTAX DisplayString (SIZE (0..64))
        ACCESS read-only
        DESCRIPTIONSTATUSmandatory
            "Description here"
        ::= { softwareElementEntry 5 }

    softwareElement-BuildNumber OBJECT-TYPE
        SYNTAX DisplayString (SIZE (0..64))
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
            "Description here"
        ::= { softwareElementEntry 6 }

    softwareElement-SerialNumber OBJECT-TYPE
        SYNTAX DisplayString (SIZE (0..64))
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
            "Description here"
```

--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
     ::= { softwareElementEntry 7 }

softwareElement-CodeSet OBJECT-TYPE
     SYNTAX DisplayString (SIZE (0..64))
     ACCESS read-only
     STATUS mandatory
     DESCRIPTION
        "Description here"
     ::= { softwareElementEntry 8 }

softwareElement-IdentificationCode OBJECT-TYPE
     SYNTAX DisplayString (SIZE (0..64))
     ACCESS read-only
     STATUS mandatory
     DESCRIPTION
        "Description here"
     ::= { softwareElementEntry 9 }

softwareElement-LanguageEdition OBJECT-TYPE
     SYNTAX DisplayString (SIZE (0..32))
     ACCESS read-only
     STATUS mandatory
     DESCRIPTION
        "Description here"
     ::= { softwareElementEntry 10 }

softwareElement-Associations OBJECT-TYPE
     SYNTAX OBJECT IDENTIFIER
     ACCESS not-accessible
     STATUS mandatory
     DESCRIPTION
        "Description here"
     ::= { softwareElementEntry 11 }

softwareElement-DeviceSoftware-LogicalDeviceAssociation-ObjectType
                            OBJECT-TYPE
     SYNTAX INTEGER {
               mediaAccessDevice (0),
               storageLibrary (1),
               other (2) }
     ACCESS read-only
     STATUS mandatory
     DESCRIPTION
        "Description here"
     ::= { softwareElementEntry 12 }

softwareElement-DeviceSoftware-LogicalDeviceAssociationId OBJECT-TYPE
     SYNTAX INTEGER
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
            "Description here"
        ::= { softwareElementEntry 13 }

    endOfSmlMib OBJECT-TYPE
        SYNTAX OBJECT IDENTIFIER
        ACCESS not-accessible
        STATUS mandatory
        DESCRIPTION
            "Description here"
        ::= { smlRoot 10 }

    END
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Annex D

(normative)

Compliance with this standard

D.1 Compliance statement

The declaration of SMI-S compliance of a given CIM Instance within a CIM Server also declares that any CIM Instance associated, directly or indirectly, to the first CIM Instance will also be SMIS compliant if SMIS itself declares compliance rules for either CIM Instance or instances of their superclasses.

D.2 Compliance declaration

- The declaration of SMI-S compliance is made through the use of the server profile and the declaration of supported profiles.
- Direct association between CIM Instances is made through instance of a CIM Association.
- Indirect association between CIM Instance is made through more than one CIM Association.
- SMI-S Compliance is assessed against CIM Instances that are directly or indirectly associated to the CIM Instance declared as part of the declaration of supported registered profiles. These CIM Instances comprise the compliance test set.
- All CIM Instances/CIM Classes included in the compliance test set for whom compliance rules are defined in SMI-S or for superclasses thereof shall themselves be compliant to the rules defined in SMI-S.
- Compliance tests on a superclass of a given CIM Instance are limited to the attributes and behaviors defined for the superclass.

D.3 Server Profile and compliance

Compliance is declared by the implementation of the Server Profile. All profiles require the Server Profile. The server profile defines the means by which a SMI-S Client can determine the profiles and subprofile support and the ComputerSystems associated (see 7.3.7, “Server Profile” for more details.)

D.4 Example

A CIM Agent for Vendor X declares compliance to the Array Profile and the Pool Manipulation Capabilities and Setting Subprofile through the Server Profile. Once the association (via the ElementConformsToProfile association) is made from the Array Profile declaration to the ComputerSystem that realizes the Array Profile, then compliance tests can be carried out. Vendor X decided to extend the StorageVolume class with additional properties. StorageVolume is associated to the ComputerSystem via SystemDevice association. ComputerSystem, StorageVolume and SystemDevice are defined in SMI-S as required CIM elements (see 7.3.4.1.12.17, “SystemDevice”).

In implementing FCPort, Vendor X decided to not provide ElementName but did provide the rest of the required properties. Vendor X decided to not use to WWN and instead used a vendor-specific value for the PermanentAddress (see 7.2.4, “Correlatable and Durable Names”). Additionally, Vendor X added FRUStatus to their subclass of FCPort. Vendor X also decided to model the back-end fibre channel, but not use an SMI-S model to do so. These backend FCPorts are associated to the ComputerSystem via the ConsumedSystemDevice association, a subclass of SystemDevice without properties overridden. These backend fibre channel ports where modelled using a Vendor X specific class, BackendFCPorts, that is not derived from FCPort. These BackendFCPorts were associated to the ComputerSystem with the ConsumedSystemDevice.PartComponent role.

The compliance test includes FCPort because the compliance declaration identified a particular ComputerSystem, the entry point into compliant CIM instantiation of the Array Profile. the compliance test includes FCPorts as part of the test set because the SystemDevice association, also defined as part of the

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

profile, includes the FCPort realized in that implementation. The compliance test also includes BackendFCPorts because the ConsumedSystemDevice association to the ComputerSystem for these instances is a SystemDevice association.

The compliance test locates the StorageConfigurationService, StoragePools including a Primordial StoragePool and StorageCapabilities associated to the ComputerSystem. Vendor X's implementation supports the creation of a StoragePool. Using one of the sizes returned by an invocation of the Primordial StoragePool's getSupportedSizes() method, the test attempts to create a StoragePool using the Primordial StoragePool reference and a StorageSetting generated from one of the StorageCapabilities.

The compliance test for Vendor X's Array Profile implementation fails because

- FCPort.PermanentName property has a noncompliance value. Specifically, the FCPort.Permanent-Address is required to be WWN, 16 unseperated uppercase hex digits,
- ElementName property was not provided (i.e., was null),
- the SystemDevice associations contained references to BackendFCPort in the PartComponent property. CIM defined that the PartComponent is a LogicalDevice. Since BackendFCPort is not a LogicalDevice, then the test failed,
- the "Size not supported" return code was returned from CreateOrModifyStoragePool even though one of the supported sizes was used verbatim.

The compliance test for Vendor X's Array Profile implementation did not fail because

- StorageVolume was extended,
- SystemDevice was extended.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Annex E
(informative)

Optional Profiles and Subprofiles

E.1 Introduction

Some sections of the CIM Schema are undergoing substantial review and redesign. In cases where it is known that the model will change substantially within the likely lifetime of this standard, components that are optional or peripheral have been relocated to this annex to avoid setting an erroneous expectation of design stability within the developer community. As the associated schema sections stabilize, these profiles or subprofiles will be relocated into the appropriate clauses of the body of the standard.

E.2 Provider Subprofile

E.2.1 Description

The CIM Provider model defines the capabilities of a provider and how it registers/unregisters from a CIM Object Manager. This model is optional for the CIM Server profile, but if supported shall adhere to the “required elements” table.

E.2.2 Standard dependencies

The Provider subprofile is defined using the CIM Schema 2.7 final. As such it can be used in profiles at 2.7 and later. It does not require that Profiles be on a later schema. It will operate within profiles that are at the CIM schema 2.7 final or later. The subprofile will operate correctly with CIM Specification 2.2 (or later) and CIM Operations over HTTP 1.1 (or later).

E.2.3 Subprofile dependencies

The Provider subprofile introduces no Profile dependencies.

E.2.4 CIM Server requirements

For the SMI-S uses of the Provider subprofile, support for Basic Read and Association Traversal functional profiles shall be supported (by the base Profile CIM server).

The Provider subprofile does not require support for extrinsic methods.

The Provider subprofile is not advertised.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

E.2.5 Instance diagrams

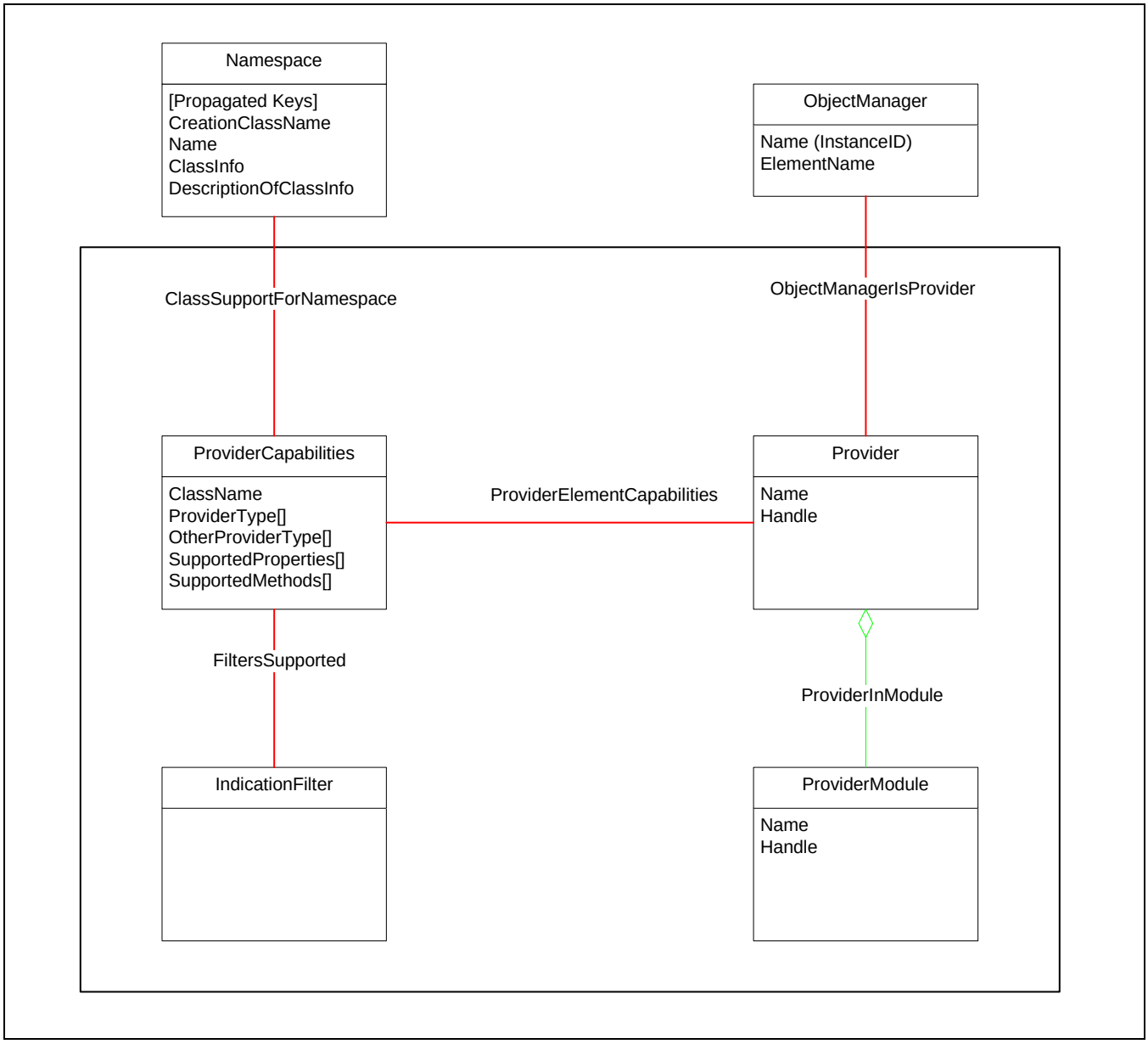


Figure E.1 - Provider Subprofile Model

E.2.6 Durable names and correlatable IDs

The Provider subprofile does not introduce any objects that have durable names or correlatable IDs, and it does not use any durable names or correlatable IDs.

E.2.7 Methods

The only methods supported for the Provider subprofile are intrinsic read methods (including association traversal). The model is populated by the CIM Server based on its own installation and startup procedures.

There are no extrinsic methods that are supported for the Provider Subprofile.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

E.2.8 Client considerations

None.

E.2.9 Recipes

None.

E.2.10 Instrumentation requirements

None.

E.2.11 Required CIM Elements

Table E.1 - Subprofile Required Classes, Associations, Methods and Indications

Subprofile Class & Associations	Notes
Provider	
ProviderCapabilities	
ProviderElementCapabilities	
ClassSupportForNamespace	
ProviderModule	
ProviderInModule	
IndicationFilter	
FiltersSupported	
ObjectManagerIsProvider	
Profile indications	**Notes**

E.2.12 Required properties for CIM Elements

E.2.12.1 Provider

A CIM Provider instruments one or more aspects of the CIM Schema. A CIM_Provider operates at the request of the CIM_ObjectManager to perform operations on CIM objects. The properties CreationClassName, SystemCreationClassName and SystemName are sometimes set to empty strings. In this case, the CIM Object Manager shall interpret the properties with the local system information.

Provider is subclassed from WBEMService.

Table E.2 - Required Properties for Provider

Class Properties	Type	Qualifier/ Parameter	Notes
Description	string		

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table E.2 - Required Properties for Provider (Continued)

Class Properties	Type	Qualifier/ Parameter	Notes
ElementName	string		
OperationalStatus[ ]	uint16		
StatusDescriptions[ ]	string		This shall not be NULL if “Other” is identified in OperationalStatus.
SystemCreationClassName	string	key, maxlen(256)	
SystemName	string	key, maxlen(256)	
CreationClassName	string	key, maxlen(256)	
Started	boolean		
Name	string	key, override	
Handle	string	req	

E.2.12.2 ProviderCapabilities

This class defines the capabilities of the associated provider.

ProviderCapabilities is subclassed from Capabilities.

Table E.3 - Required Properties for ProviderCapabilities

Class Properties	Type	Qualifier/ Parameter	Notes
Description	string		
InstanceID	string	key	
ElementName	string	req	
ClassName	string	req	
ProviderTypes[ ]	uint16		
OtherProviderTypes[ ]	string		
SupportedProperties[ ]	string		
SupportedMethods[ ]	string		

E.2.12.3 ProviderElementCapabilities

ProviderElementCapabilities is an association describing the Capabilities that are supported by a Provider.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

ProviderElementCapabilities is subclassed from ElementCapabilities.

Table E.4 - Required Properties ProviderElementCapabilities

Class Properties	Type	Qualifier/ Parameter	Notes
ManagedElement	ref		The CIM Provider.
Capabilities	ref		The CIM Provider's Capabilities.

E.2.12.4 ClassSupportForNamespace

ClassSupportForNamespace is an association describing the target Namespace for the instances of the class listed in the referenced ProviderCapabilities.ClassName property.

ClassSupportForNamespace is subclassed from Dependency.

Table E.5 - Required Properties for ClassSupportForNamespace

Class Properties	Type	Qualifier/ Parameter	Notes
Antecedent	ref		The Namespace in which the class instances are defined.
Dependent	ref		The ProviderCapabilities instance supporting the class instances.

E.2.12.5 ProviderModule

A ProviderModule consists of one or more Provider Services. It can be enabled/disabled, which affects the component Services.

ProviderModule is subclassed from EnabledLogicalElement.

Table E.6 - Required Properties for ProviderModule

Class Properties	Type	Qualifier/ Parameter	Notes
Description	string		
ElementName	string		
OperationalStatus[ ]	uint16		
StatusDescriptions[ ]	string		This shall not be NULL if “Other” is identified in OperationalStatus.
Name	string	key, override	
Handle	string	req	

E.2.12.6 ProviderInModule

An association describing the Providers that are contained in a ProviderModule.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

ProviderInModule is subclassed from Component.

Table E.7 - Required Properties for ProviderInModule

Class Properties	Type	Qualifier/ Parameter	Notes
GroupComponent	ref		The CIM ProviderModule.
PartComponent	ref		The CIM Providers.

E.2.12.7 IndicationFilter

IndicationFilter defines the criteria for generating an Indication and what data should be returned in the Indication. It is derived from CIM_ManagedElement to allow modelling the dependency of the filter on a specific service.

E.2.12.8 IndicationFilter is subclassed from ManagedElement.

Table E.8 - Required Properties for IndicationFilter

Class Properties	Type	Qualifier/ Parameter	Notes
Description	string		
ElementName	string		
SystemCreationClassName	string	key, maxlen(256)	
SystemName	string	key, maxlen(256)	
CreationClassName	string	key, maxlen(256)	
Name	string	key	
Query	string	req	
QueryLanguage	string	req	

E.2.12.9 FiltersSupported

FiltersSupported is an association describing the CIM IndicationFilters that are supported by a Provider.

FiltersSupported is subclassed from Dependency.

Table E.9 - Required Properties for FiltersSupported

Class Properties	Type	Qualifier/ Parameter	Notes
Antecedent	ref		The CIM IndicationFilter supported for the CIM classes listed in ClassNames array property of the referenced ProviderCapabilities instance.
Dependent	ref		The CIM Provider Capabilities.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

E.2.12.10 ObjectManagerIsProvider

This association indicates that the referenced ObjectManager acts as a Provider for the CIM classes listed in the associated ProviderCapabilities.

ObjectManagerIsProvider is subclassed from ConcreteIdentity.

Table E.10 - Required Properties for ObjectManagerIsProviderRequired

Class Properties	Type	Qualifier/ Parameter	Notes
SystemElement	ref		SystemElement represents one aspect of the Logical Element.
SameElement	ref		SameElement represents an alternate aspect of the System entity.

E.2.12.11 Optional Subprofiles and Profiles

There are no optional subprofiles or profiles for this subprofile.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Bibliography

The following documents provide useful background and context for the understanding this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 14165-413, *Information technology – Fibre Channel – Part 413: Generic Services-3 (FC-GS-3)* (under consideration)

ISO/IEC 14165-414, *Information technology – Fibre Channel – Part 414: Generic Services-4 (FC-GS-4)* (under consideration) [ANSI INCITS 387-2004]

ISO/IEC 14165-511, *Information technology – Fibre Channel – Part 511: Host Bus Adapter Application Programming Interface (FC-HBA)* (under consideration) [ANSI INCITS 386-2004]

ISO/IEC 14776-371, *Information technology – Small Computer System Interface – Part 371: Enclosure Services Command Set (SCSI-3 SES)* (under consideration) [ANSI INCITS 305-1998 (R2003)]

INCITS 424 - 2007, *Information technology - Fibre Channel Framing and Signaling - 2 (FC-FS-2)* (forwarded to ISO as NP 14165-252)

The following reference materials on SLP are recommended to assist in vendor implementations of SLP processes.

Kempf, J. and P. St. Pierre. _Service Location Protocol for Enterprise Networks_. New York: John Wiley and Sons, Inc., 1999.

Perkins, C. and E. Guttman. “DHCP Options for Service Location Protocol.” IETF RFC 2610, June 1999.

Guttman, E., C. Perkins, and J. Veizades, and M. Day. “Service Location Protocol, Version 2." IETF RFC 2608, June 1999.

Guttman, E., C. Perkins, and J. Kempf. “Service Templates and service: Schemes.” IETF RFC 2609, June 1999.

Kempf, J. and E. Guttman. “An API for Service Location.” IETF Informational RFC 2614, June 1999.

Guttman, E. “The serviceid: URI Scheme for Service Location.” draft-guttman-svrloc-serviceid-01.txt, IETF Informational Draft, Network Working Group, January 4, 1999

Physical Interconnects

ADP	Automation/Drive Interface – Physical Layer	[T10/1556-D]
FC-AL	Fibre Channel Arbitrated Loop	[ANSI X3.272:1996]
FC-AL-2	ISO/IEC 14165-122: *Information technology – Fibre Channel – Part 122: Arbitrated Loop -2* (FC-AL-2)	[ANSI NCITS.332:1999]
FC-100-DF-EL-S	ISO/IEC 14165-114: *Information technology – Fibre Channel – Part 114: 100 MB/s copper physical interface (FC-100-DF-EL-S)*	
FC-PI	ISO/IEC 14165-115: *Information technology – Fibre Channel – Part 115: Physical interfaces (FC-PI)*	[NCITS1235-D]

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

FC-PI-2	*Fibre Channel Physical Interfaces-2*	[T11/1506-D]
FC-PH	*Fibre Channel Physical and Signalling Interface*	[ANSI X3.230:1994]
	Fibre Channel Physical Amendment 1	[ANSI X3.230/AM1:1996]
FC-PH-3	*Fibre Channel 3rd Generation Physical Interface*	[ANSI X3.303 -1998]
FC-FS	*ISO/IEC 14165-251: Information technology – Fibre Channel – Part 251: Fibre Channel Framing and Signalling Interface (FC-FS)*(under consideration)	[T11/1331-D]
	High Performance Serial Bus	[ANSI IEEE 1394:1995]
	High Performance Serial Bus (supplement to ANSI IEEE 1394:1995)	[ANSI IEEE 1394a:2000]
SPI-2	*ISO/IEC 14776-112:2002, Information technology - Small Computer System Interface (SCSI) – Part 112: SCSI Parallel Interface-2 (SPI-2)*	[ANSI X3.302:1999]
SPI-3	*ISO/IEC 14776-113:2002, Information technology – Small Computer System Interface (SCSI) – Part 113: Parallel Interface-3 (SPI-3)*	[ANSI NCITS.336:2000]
SPI-4	*ISO/IEC 14776-114, Information technology – Small Computer System Interface (SCSI) – Part 114: Parallel Interface-4 (SPI-4)* (under consideration)	[ANSI INCITS.362-200x]
SPI-5	*ISO/IEC 14776-115:2004, Information technology - Small Computer System Interface (SCSI) – Part 115: Parallel Interface -5 (SPI-5)*	[T10/1525 -D]
SSA-PH	*Serial Storage Architecture Physical Layer 1*	[ANSI X3.293:1996]
SSA-PH-2	*Serial Storage Architecture Physical Layer 2*	[ANSI NCITS.307:1998]
SAS	*ISO/IEC 14776-150:2004, Information technology – Small Computer System Interface (SCSI) – Part 150: Serial Attached SCSI (SAS)*	[T10/1562-D]

SCSI transport protocols

ADT	*Automation/Drive Interface – Transport Protocol*	[T10/1557-D]
SSA-TL-1	*Serial Storage Architecture Transport Layer 1*	[ANSI X3.295:1996]
SSA-TL-2	*Serial Storage Architecture Transport Layer 2*	[ANSI NCITS.308:1998]
FCP	*ISO/IEC 14776-221: replaced by 14776-222, SCSI -3 Fibre Channel Protocol (FCP)*	[ANSI X3.269:1996]
FCP-2	*ISO/IEC 14776-222: Information technology – Small Computer System Interface (SCSI) – Part 222: Fibre Channel Protocol for SCSI, second version (FCP-2) being replaced by 14776-223, SCSI -3*	ANSI NCITS 350-2003 [T10/1144-D]

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

FCP-3	ISO/IEC 14776-223, *Information technology – Small Computer System Interface (SCSI) – Part 223: Fibre Channel Protocol-3 (FCP-3)* (under consideration)	[T10/1560-D]
SBP-2	ISO/IEC 14776-232:2001, *Information technology – Small Computer System Interface (SCSI) – Part 232:Serial Bus Protocol 2 (SBP-2)*	[ANSI NCITS.325:1999]
SBP-3	ISO/IEC 14776–233:(under development), *Information technology – Small Computer System Interface (SCSI) – Part 233: Serial Bus Protocol -3 (SBP-3)* (under consideration)	[T10/1467-D]
SSA-S2P	Serial Storage Architecture SCSI -2 Protocol	[ANSI X3.294:1996]
SSA-S3P	Serial Storage Architecture SCSI -3 Protocol	[ANSI NCITS.309:1998]
SRP	SCSI RDMA Protocol	[T10/1415-D]
SST	SCSI on Scheduled Transfer	[T10/1380-D]
SVP	SCSI VI Protocol	[T10/1415-D]
Shared command sets		
SPC	SCSI -3 Primary Commands	[ANSI X3.301:1997]
SPC-2	ISO/IEC 14776-452, *Information technology – Small Computer System Interface (SCSI) – Part 452: SCSI Primary Commands-2 (SPC-2)*	[ANSI NCITS.351-2001] [T10/1236-D]
SPC-3	ISO/IEC 14776-453, *Information technology – Small Computer System Interface (SCSI) – Part 453: SCSI Primary Commands-3 (SPC-3)* (under consideration)	ANSI INCITS 408-2005 [T10/1416-D]
Device-type specific command sets		
SBC	ISO/IEC 14776-321:2002, *Information technology – Small Computer System Interface (SCSI-3) – Part 321: Block Commands (SBC)*	[ANSI NCITS.306:1998]
SBC-2	ISO/IEC 14776-322, *Information technology – Small Computer System Interface (SCSI) – Part 322: Block Commands-2 (SBC-2)* (under consideration)	[T10/1417-D]
SSC	ISO/IEC 14776-331:2002, *Information technology - Small Computer System Interface (SCSI) – Part 331: SCSI -3 Stream Commands (SSC)*	[ANSI NCITS.335:2000]
SSC-2	ISO/IEC 14776-332, *Information technology – Small Computer System Interface (SCSI) – Part 332: Stream Commands -2 (SSC-2)* (under consideration)	[T10/1434-D]

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

SMC	ISO/IEC 14776-351, *Information technology – Small Computer System Interface (SCSI-3) – Part 351: Medium Changer Commands (SMC)* (under consideration)	[ANSI NCITS.314:1998]
SMC-2	ISO/IEC 14776 -352, *Information technology – Small Computer System Interface (SCSI) – Part 252: Medium Changer Commands -2 (SMC-2)* (under consideration)	[T10/1383-D]
MMC	SCSI -3 Multimedia Command Set	[ANSI X3.304:1997]
MMC-2	ISO/IEC 14776-362, *Information technology – Small Computer System Interface (SCSI) – Part 362: Multimedia Command Set-2 (MMC-2)* (under consideration)	[ANSI NCITS.333:2000]
MMC-3	ISO/IEC 14776-363, *Information technology – Small Computer System Interface (SCSI) – Part 363: Multimedia Command Set-3 (*MMC-3) (under consideration)	[T10/1363-D]
MMC-4	ISO/IEC 14776-364, *Information technology – Small Computer System Interface (SCSI) – Part 364: Multimedia Command Set -3 (*MMC-4) (under consideration)	[T10/1545-D]
SCC	ISO/IEC 14776-341:2000, *Information technology – Small Computer System Interface (SCSI-3) – Part 341: Controller Commands (SCC)*	[ANSI X3.276:1997]
SCC-2	ISO/IEC 14776-342:2000, *Information technology – Small Computer System Interface (SCSI) – Part 342: Controller Commands-2 (SCC-2)*	[ANSI NCITS.318:1998]
RBC	ISO/IEC 14776-326:2002, *Information technology – Small Computer System Interface (SCSI) – Part 326: Reduced Block Commands (RBC)*	[ANSI NCITS.330:2000]
RMC	SCSI Reduced MultiMedia Commands	[T10/1364-D]
SES	ISO/IEC 14776-371, *Information technology – Small Computer System Interface (SCSI-3) – Part 371: Enclosure Services Commands (SES)* (under consideration)	[ANSI NCITS.305:1998]
SES-2	ISO/IEC 14776-372, *Information technology – Small Computer System Interface (SCSI) – Part 372: SCSI -3 Enclosure Services Commands (SES-2)* (under consideration)	[T10/1559-D]
OMC	ISO/IEC 14776-381:2000, *Information technology - Small Computer System Interface (SCSI) – Part 381: Optical Memory Card device commands (OMC)*	
OSD	Object-based Storage Devices Commands	[T10/1355-D]

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

MSC	SCSI Management Server Commands]	[T10/1528-D
ADC	Automation/Drive Interface – Commands	[T10/1558 -D]
Architecture model		
SAM	ISO/IEC 14776-411:1999, *Information technology – Small Computer System Interface (SCSI-3) – Part 411: Architecture Model (SAM)*	[ANSI X3.270:1996]
SAM-2	ISO/IEC 14776-412:(under development), *Information technology – Small Computer System Interface (SCSI) – Part 412: SCSI Architecture Model-2 (SAM-2)* (under consideration)	[T10/1157-D]
SAM-3	ISO/IEC 14776-413:(under development), *Information technology – Small Computer System Interface (SCSI) – Part 413: SCSI Architecture Model-3 (SAM-2)* (under consideration)	ANSI INCITS 402-2005 [T10/1561-D]
Common Access method		
CAM	ISO/IEC 9316-2:2000 06, *Information technology – Small computer system interface-2 (SCSI -2) – Part 2: Common Access Method (CAM) – Transport and SCSI interface module (CAM)*	[ANSI X3.232:1996]

The following web sites may also provide useful background and context for the understanding this document. The dynamic nature of web documents and web site organization may have invalidated some of the cited URLs, but all were operational at this time of publication.

CIM	http://www.dmtf.org/education/tutorials/ http://www.dmtf.org/education/standards/cim http://www.dmtf.org/education/standards/published_documents
XML	http://www.w3.org/XML/
Language Bindings	http://wbemsource.org
FC Identifiers	http://standards.ieee.org/regauth/oui
WQL	http://msdn2.microsoft.com/en-us/library/aa394606.aspx

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

ISBN 2-8318-9200-7



ICS 35.200

Typeset and printed by the IEC Central Office
GENEVA, SWITZERLAND

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale