



ISO/IEC 24775-4

Edition 1.0 2014-10

INTERNATIONAL STANDARD



Information technology – Storage management –
Part 4: Block devices

IECNORM.COM : Click to view the full PDF of ISO/IEC 24775-4:2014



THIS PUBLICATION IS COPYRIGHT PROTECTED

Copyright © 2014 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 Varembe
CH-1211 Geneva 20
Switzerland

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

IEC Catalogue - webstore.iec.ch/catalogue

The stand-alone application for consulting the entire bibliographical information on IEC International Standards, Technical Specifications, Technical Reports and other documents. Available for PC, Mac OS, Android Tablets and iPad.

IEC publications search - www.iec.ch/searchpub

The advanced search enables to find IEC publications by a variety of criteria (reference number, text, technical committee,...). It also gives information on projects, replaced and withdrawn publications.

IEC Just Published - webstore.iec.ch/justpublished

Stay up to date on all new IEC publications. Just Published details all new publications released. Available online and also once a month by email.

Electropedia - www.electropedia.org

The world's leading online dictionary of electronic and electrical terms containing more than 30 000 terms and definitions in English and French, with equivalent terms in 14 additional languages. Also known as the International Electrotechnical Vocabulary (IEV) online.

IEC Glossary - std.iec.ch/glossary

More than 55 000 electrotechnical terminology entries in English and French extracted from the Terms and Definitions clause of IEC publications issued since 2002. Some entries have been collected from earlier publications of IEC TC 37, 77, 86 and CISPR.

IEC Customer Service Centre - webstore.iec.ch/csc

If you wish to give us your feedback on this publication or need further assistance, please contact the Customer Service Centre: csc@iec.ch.

IECNORM.COM : Click to view the full text of ISO/IEC 24775-4:2014



ISO/IEC 24775-4

Edition 1.0 2014-10

INTERNATIONAL STANDARD



Information technology – Storage management –
Part 4: Block devices

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

PRICE CODE

H

ICS 35.200

ISBN 978-2-8322-1896-9

Warning! Make sure that you obtained this publication from an authorized distributor.

Contents

FOREWORD.....	41
Introduction	43
1 Scope	45
2 Normative References.....	45
3 Terms and definitions	45
4 Typographical Conventions	46
4.1 Maturity Model.....	46
4.2 Experimental Maturity Level.....	46
4.3 Implemented Maturity Level.....	46
4.4 Stable Maturity Level.....	47
4.5 Finalized Maturity Level	47
4.6 Deprecated Material	47
5 Array Profile.....	49
5.1 Description	49
5.2 Health and Fault Management.....	51
5.3 Cascading Considerations	51
5.4 Supported Subprofiles and Packages.....	51
5.5 Methods of the Profile	52
5.6 Client Considerations and Recipes.....	52
5.7 Registered Name and Version	52
5.8 CIM Elements.....	53
5.8.1 General.....	53
5.8.2 CIM_ComputerSystem (Top Level System)	54
5.8.3 CIM_FilterCollection (Array Predefined FilterCollection)	54
5.8.4 CIM_HostedCollection (Array to predefined FilterCollection)	55
5.8.5 CIM_IndicationFilter (Array System Creation)	55
5.8.6 CIM_IndicationFilter (Array System Deletion)	56
5.8.7 CIM_MemberOfCollection (Predefined Filter Collection to Array Filters)	57
5.8.8 CIM_ProtocolControllerForUnit (Arbitrary LU for All LUNs View)	57
5.8.9 CIM_ProtocolControllerForUnit (Storage volumes for All LUNs View)	58
5.8.10 CIM_SCSIArbitraryLogicalUnit (Arbitrary LU).....	58
5.8.11 CIM_SCSIProtocolController (All LUNs View).....	59
5.8.12 CIM_SystemDevice (System to SCSIArbitraryLogicalUnit).....	59
5.8.13 CIM_SystemDevice (System to SCSIProtocolController)	59
6 Block Services Package	61
6.1 Description	61
6.1.1 General.....	61
6.1.2 Storage Capacity States.....	61
6.1.3 StoragePools	62
6.1.4 Blocks, Metadata, and Capacity Reported	62
6.1.5 StoragePool Management Instance Diagram.....	63
6.1.6 StoragePool, StorageVolume and LogicalDisk Manipulation	63
6.1.7 Declaring Storage Configuration Options	66
6.1.8 StorageVolume Creation Instance Diagram	70
6.1.9 Backward Compatibility	70
6.1.10 Capacity Management.....	71
6.1.11 Mapping of RAID levels to Data Redundancy and Package Redundancy	73
6.1.12 Storage Setting Associations to Storage Capabilities	74
6.1.13 The Usage Property	74
6.1.14 Read-Only Model Requirements	75
6.1.15 StorageExtent Conservation.....	76
6.1.16 Formulas For Calculating Capacity	80

6.1.17	Storage Element Manipulation	81
6.1.18	Block Services Predefined Indications	82
6.2	Health and Fault Management Considerations	84
6.3	Cascading Considerations	84
6.4	Supported Profile, Subprofiles and Packages	84
6.5	Methods of this Profile	84
6.5.1	Extrinsic Methods on StorageCapabilities	84
6.5.2	Intrinsic Methods on StorageSetting	87
6.5.3	Extrinsic Methods on StorageConfiguration	87
6.5.4	Extrinsic Methods on StoragePool	93
6.6	Client Considerations and Recipes	99
6.6.1	Representative Instance Diagram	99
6.6.2	Goals and Settings	99
6.6.3	Representative StoragePool Creation Example	100
6.6.4	Representative example of StorageVolume or LogicalDisk Creation	102
6.6.5	Summarize the StoragePools in a block storage system and verify the capacity reported	105
6.6.6	Conditional: Create StoragePool and Storage Element on Block Server (e.g., Array or Volume Manager)	106
6.6.7	Conditional: Expand Storage Element on Block Server	114
6.6.8	Conditional: Create Storage Element from Elements on Block Server	119
6.6.9	Optional: Intentionally General a CIM Error	125
6.7	Registered Name and Version	126
6.8	CIM Elements	126
6.8.1	General	126
6.8.2	CIM_AllocatedFromStoragePool (Pool from Pool)	132
6.8.3	CIM_AllocatedFromStoragePool (Volume or LogicalDisk from Pool)	132
6.8.4	CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StorageVolume or LogicalDisk)	133
6.8.5	CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StoragePool)	133
6.8.6	CIM_ElementCapabilities (StorageCapabilities to StorageConfigurationService)	133
6.8.7	CIM_ElementCapabilities (StorageCapabilities to StoragePool)	134
6.8.8	CIM_ElementCapabilities (StorageConfigurationCapabilities to StorageConfigurationService)	134
6.8.9	CIM_ElementCapabilities (StorageConfigurationCapabilities to concrete StoragePool)	134
6.8.10	CIM_ElementCapabilities (StorageConfigurationCapabilities to primordial StoragePool)	135
6.8.11	CIM_ElementCapabilities (Used to declare the naming capabilities of the StoragePool)	135
6.8.12	CIM_ElementCapabilities (Used to declare the naming capabilities of the StorageVolume or LogicalDisk)	136
6.8.13	CIM_ElementSettingData	136
6.8.14	CIM_EnabledLogicalElementCapabilities (For StorageConfigurationService)	136
6.8.15	CIM_EnabledLogicalElementCapabilities (For StoragePool)	137
6.8.16	CIM_FilterCollection (Block Services Predefined FilterCollection)	138
6.8.17	CIM_HostedCollection (System to predefined IndicationFilters)	138
6.8.18	CIM_HostedService	138
6.8.19	CIM_HostedStoragePool	139
6.8.20	CIM_IndicationFilter (Logical Disk Creation)	139
6.8.21	CIM_IndicationFilter (Logical Disk Deletion)	140
6.8.22	CIM_IndicationFilter (Logical Disk OperationalStatus)	141
6.8.23	CIM_IndicationFilter (Storage Pool Creation)	142
6.8.24	CIM_IndicationFilter (Storage Pool Deletion)	142
6.8.25	CIM_IndicationFilter (Storage Pool TotalManagedSpace)	143

6.8.26	CIM_IndicationFilter (Storage Volume Creation).....	144
6.8.27	CIM_IndicationFilter (Storage Volume Deletion)	145
6.8.28	CIM_IndicationFilter (Storage Volume OperationalStatus).....	146
6.8.29	CIM_IndicationFilter (WQL Logical Disk OperationalStatus).....	147
6.8.30	CIM_IndicationFilter (WQL Storage Volume OperationalStatus)	148
6.8.31	CIM_LogicalDisk.....	149
6.8.32	CIM_MemberOfCollection (Block Services Filter Collection to FilterCollection)..	150
6.8.33	CIM_MemberOfCollection (Predefined Filter Collection to Block Services Filters)	150
6.8.34	CIM_OwningJobElement.....	151
6.8.35	CIM_StorageCapabilities.....	151
6.8.36	CIM_StorageConfigurationCapabilities (Concrete)	153
6.8.37	CIM_StorageConfigurationCapabilities (Global).....	154
6.8.38	CIM_StorageConfigurationCapabilities (Primordial)	155
6.8.39	CIM_StorageConfigurationService	156
6.8.40	CIM_StoragePool (Concrete)	156
6.8.41	CIM_StoragePool (Empty).....	157
6.8.42	CIM_StoragePool (Primordial).....	158
6.8.43	CIM_StorageSetting	158
6.8.44	CIM_StorageSettingWithHints.....	160
6.8.45	CIM_StorageSettingsAssociatedToCapabilities.....	161
6.8.46	CIM_StorageSettingsGeneratedFromCapabilities	162
6.8.47	CIM_StorageVolume	162
6.8.48	CIM_SystemDevice (System to StorageVolume or LogicalDisk)	163
6.8.49	SNIA_StorageVolume	164
7	Block Storage Views Profile	166
7.1	Description	166
7.1.1	Synopsis	166
7.1.2	Overview.....	166
7.1.3	Class Diagram for SNIA View Classes.....	168
7.1.4	Implementation	169
7.2	Health and Fault Management Consideration.....	182
7.3	Cascading Considerations	182
7.4	Supported Profiles, Subprofiles, and Packages.....	182
7.5	Methods of the Profile	182
7.5.1	Extrinsic Methods of the Profile	182
7.5.2	Intrinsic Methods of the Profile	182
7.6	Client Considerations and Recipes	183
7.6.1	Use Cases	183
7.6.2	Recipes.....	186
7.7	CIM Elements.....	186
7.7.1	General.....	186
7.7.2	CIM_ElementCapabilities (View Capabilities)	191
7.7.3	SNIA_AllocatedFromStoragePoolView (StoragePoolView to StoragePool).....	191
7.7.4	SNIA_AllocatedFromStoragePoolView (Volume to StoragePoolView)	192
7.7.5	SNIA_AllocatedFromStoragePoolView (VolumeView to StoragePool)	192
7.7.6	SNIA_AllocatedFromStoragePoolViewView (PoolView to PoolView)	193
7.7.7	SNIA_AllocatedFromStoragePoolViewView (VolumeView to PoolView)	194
7.7.8	SNIA_BaseInstance (DiskDrive).....	194
7.7.9	SNIA_BaseInstance (StorageSetting)	194
7.7.10	SNIA_BaseInstance (Volume).....	195
7.7.11	SNIA_BasedOnView (ExtentOnDriveExtent)	195
7.7.12	SNIA_BasedOnView (VolumeOnExtent).....	196
7.7.13	SNIA_ConcreteComponentView	196
7.7.14	SNIA_ContainerView.....	197

7.7.15	SNIA_DiskDriveView	197
7.7.16	SNIA_DriveComponentViewView	199
7.7.17	SNIA_ElementStatisticalDataView (DiskDriveView)	200
7.7.18	SNIA_ElementStatisticalDataView (VolumeView)	200
7.7.19	SNIA_ExposedView	200
7.7.20	SNIA_ExtentComponentView	201
7.7.21	SNIA_HostedStoragePoolView	201
7.7.22	SNIA_MappingProtocolControllerView	202
7.7.23	SNIA_MaskingMappingView	203
7.7.24	SNIA_ProtocolControllerForUnitView	204
7.7.25	SNIA_ReplicaPairView	205
7.7.26	SNIA_StoragePoolView	208
7.7.27	SNIA_SystemDeviceView (DiskDriveViews)	210
7.7.28	SNIA_SystemDeviceView (MappingProtocolControllerViews)	210
7.7.29	SNIA_SystemDeviceView (ReplicaPairViews)	211
7.7.30	SNIA_SystemDeviceView (VolumeViews)	211
7.7.31	SNIA_ViewCapabilities	211
7.7.32	SNIA_VolumeView	212
8	Block Server Performance Subprofile	216
8.1	Description	216
8.1.1	Synopsis	216
8.1.2	Overview	216
8.2	Implementation	218
8.2.1	Performance Additions Overview	218
8.2.2	Performance Additions to base Array Profile	221
8.2.3	Performance Additions to base Storage Virtualizer Profile	222
8.2.4	Performance Additions to base Volume Management Profile	225
8.2.5	Summary of BlockStorageStatisticsData support by Profile	227
8.2.6	Server Profile Support for the Block Server Performance Subprofile	227
8.2.7	Default Manifest Collection	227
8.2.8	Performance Additions applied to Multiple Computer Systems	228
8.2.9	Performance Additions to Backend Ports	229
8.2.10	Performance Additions to Extent Composition	231
8.2.11	Performance Additions to Disk Drives	232
8.2.12	Performance Additions to SCSIArbitraryLogicalUnits (Controller LUNs)	233
8.2.13	Performance Additions for Remote Mirrors	234
8.2.14	Client Defined Manifest Collections	234
8.2.15	Capabilities Support for Block Server Performance Subprofile	237
8.3	Health and Fault Management Considerations	238
8.4	Cascading Considerations	238
8.5	Supported Subprofiles and Packages	238
8.6	Methods of the Profile	238
8.6.1	Extrinsic Methods of the Profile	238
8.6.2	Intrinsic Methods of the Profile	243
8.7	Client Considerations and Recipes	245
8.7.1	Bulk Performance Statistics Gathering	245
8.7.2	Building an Object Map of Metered Elements	254
8.7.3	Retrieving Statistics for a Specific Volume	263
8.7.4	Summary of Statistics Support by Element	267
8.7.5	Formulas and Calculations	268
8.7.6	Block Server Performance Supported Capabilities Patterns	269
8.7.7	Correlation of Block Storage Statistics and Fabric Statistics	270
8.8	CIM Elements	270
8.8.1	General	270

8.8.2	CIM_AssociatedBlockStatisticsManifestCollection (Client defined collection)	273
8.8.3	CIM_AssociatedBlockStatisticsManifestCollection (Provider defined collection)	273
8.8.4	CIM_BlockStatisticsCapabilities	274
8.8.5	CIM_BlockStatisticsManifest (Client Defined)	275
8.8.6	CIM_BlockStatisticsManifest (Provider Support)	276
8.8.7	CIM_BlockStatisticsManifestCollection (Client Defined)	277
8.8.8	CIM_BlockStatisticsManifestCollection (Provider Defined)	278
8.8.9	CIM_BlockStatisticsService	278
8.8.10	CIM_BlockStorageStatisticalData	280
8.8.11	CIM_ElementCapabilities	283
8.8.12	CIM_ElementStatisticalData (Back end Port Stats)	284
8.8.13	CIM_ElementStatisticalData (Component System Stats)	284
8.8.14	CIM_ElementStatisticalData (Disk Stats)	285
8.8.15	CIM_ElementStatisticalData (Extent Stats)	285
8.8.16	CIM_ElementStatisticalData (Front end Port Stats)	286
8.8.17	CIM_ElementStatisticalData (Logical Disk Stats)	286
8.8.18	CIM_ElementStatisticalData (Remote Copy Stats)	287
8.8.19	CIM_ElementStatisticalData (Top Level System Stats)	287
8.8.20	CIM_ElementStatisticalData (Volume Stats)	288
8.8.21	CIM_HostedCollection (Client Defined)	288
8.8.22	CIM_HostedCollection (Default)	289
8.8.23	CIM_HostedCollection (Provider Supplied)	289
8.8.24	CIM_HostedService	290
8.8.25	CIM_MemberOfCollection (Member of client defined collection)	290
8.8.26	CIM_MemberOfCollection (Member of pre-defined collection)	290
8.8.27	CIM_MemberOfCollection (Member of statistics collection)	291
8.8.28	CIM_StatisticsCollection	291
8.8.29	SNIA_BlockStatisticsCapabilities	292
8.8.30	SNIA_BlockStatisticsManifest (Client Defined)	292
8.8.31	SNIA_BlockStatisticsManifest (Provider Support)	293
9	CKD Block Services Profile	295
9.1	Description	295
9.1.1	Synopsis	295
9.1.2	Overview	295
9.1.3	Implementation	295
9.2	Health and Fault Management Consideration	298
9.3	Cascading Considerations	298
9.4	Supported Profiles, Subprofiles, and Packages	298
9.5	Methods of the Profile	298
9.6	Client Considerations and Recipes	298
9.7	Registered Name and Version	298
9.8	CIM Elements	299
9.8.1	General	299
9.8.2	CIM_AllocatedFromStoragePool	305
9.8.3	CIM_AllocatedFromStoragePool (Pool from Pool)	305
9.8.4	CIM_AllocatedFromStoragePool (Volume or LogicalDisk from Pool)	305
9.8.5	CIM_ElementCapabilities	306
9.8.6	CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StorageVolume or LogicalDisk)	306
9.8.7	CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StoragePool)	306
9.8.8	CIM_ElementCapabilities (StorageCapabilities to Storage ConfigurationService)	307
9.8.9	CIM_ElementCapabilities (StorageCapabilities to StoragePool)	307
9.8.10	CIM_ElementCapabilities (StorageConfigurationCapabilities to	

	StorageConfigurationService).....	307
9.8.11	CIM_ElementCapabilities (StorageConfigurationCapabilities to concrete StoragePool).....	308
9.8.12	CIM_ElementCapabilities (StorageConfigurationCapabilities to primordial StoragePool).....	308
9.8.13	CIM_ElementCapabilities (Used to declare the naming capabilities of the StoragePool).....	308
9.8.14	CIM_ElementCapabilities (Used to declare the naming capabilities of the StorageVolume or LogicalDisk).....	309
9.8.15	CIM_ElementSettingData.....	309
9.8.16	CIM_EnabledLogicalElementCapabilities (For StorageConfigurationService)....	309
9.8.17	CIM_EnabledLogicalElementCapabilities (For StoragePool).....	310
9.8.18	CIM_FilterCollection (Block Services Predefined FilterCollection).....	311
9.8.19	CIM_HostedCollection (System to predefined IndicationFilters).....	311
9.8.20	CIM_HostedService.....	311
9.8.21	CIM_HostedStoragePool.....	311
9.8.22	CIM_IndicationFilter (Logical Disk Creation).....	312
9.8.23	CIM_IndicationFilter (Logical Disk Deletion).....	312
9.8.24	CIM_IndicationFilter (Logical Disk OperationalStatus).....	313
9.8.25	CIM_IndicationFilter (Storage Pool Creation).....	314
9.8.26	CIM_IndicationFilter (Storage Pool Deletion).....	315
9.8.27	CIM_IndicationFilter (Storage Pool TotalManagedSpace).....	316
9.8.28	CIM_IndicationFilter (Storage Volume Creation).....	317
9.8.29	CIM_IndicationFilter (Storage Volume Deletion).....	318
9.8.30	CIM_IndicationFilter (Storage Volume OperationalStatus).....	319
9.8.31	CIM_IndicationFilter (WQL Logical Disk OperationalStatus).....	320
9.8.32	CIM_IndicationFilter (WQL Storage Volume OperationalStatus).....	321
9.8.33	CIM_LogicalDisk.....	322
9.8.34	CIM_MemberOfCollection (Block Services Filter Collection to FilterCollection)..	323
9.8.35	CIM_MemberOfCollection (Predefined Filter Collection to Block Services Filters).....	324
9.8.36	CIM_OwningJobElement.....	324
9.8.37	CIM_StorageConfigurationCapabilities.....	324
9.8.38	CIM_StorageConfigurationCapabilities (Concrete).....	324
9.8.39	CIM_StorageConfigurationCapabilities (Global).....	325
9.8.40	CIM_StorageConfigurationCapabilities (Primordial).....	326
9.8.41	CIM_StorageConfigurationService.....	327
9.8.42	CIM_StoragePool.....	327
9.8.43	CIM_StoragePool (Concrete).....	327
9.8.44	CIM_StoragePool (Empty).....	328
9.8.45	CIM_StoragePool (Primordial).....	329
9.8.46	CIM_StorageSettingWithHints.....	329
9.8.47	CIM_StorageSettingsAssociatedToCapabilities.....	330
9.8.48	CIM_StorageSettingsGeneratedFromCapabilities.....	330
9.8.49	CIM_SystemDevice (System to StorageVolume or LogicalDisk).....	330
9.8.50	SNIA_StorageCapabilities.....	330
9.8.51	SNIA_StorageSetting.....	331
9.8.52	SNIA_StorageVolume.....	333
9.8.53	SNIA_StorageVolume.....	334
10	Copy Services Subprofile.....	336
10.1	Description.....	336
10.1.1	Synopsis.....	336
10.1.2	Overview.....	336
10.1.3	Copy Services Discovery.....	338
10.1.4	Copy Services Capabilities.....	339

10.1.5	Replication modeling	342
10.1.6	Associations	344
10.1.7	Durable Names and Correlatable IDs of the Profile	349
10.1.8	Accessibility to Created Elements	349
10.1.9	Completion of Long Operations.....	350
10.1.10	State Management For Associated Replicas	352
10.1.11	Reporting Time of Synchronization	355
10.1.12	State Transition Rules	355
10.1.13	State Transitions.....	357
10.1.14	Accessibility to Associations and Elements.....	365
10.1.15	Host Access Restrictions.....	366
10.1.16	Settings, Specialized Elements and Pools for Replicas	366
10.1.17	Backward Compatibility	367
10.1.18	Mutually Exclusive Capabilities	367
10.1.19	Deleting the Target Elements	368
10.1.20	Using StorageSettings for Replicas.....	368
10.1.21	Finding and Creating Target Elements.....	368
10.1.22	Using StoragePools for Replicas.....	368
10.1.23	Thinly Provisioned Elements	370
10.1.24	Indication Events	370
10.2	Health and Fault Management Considerations.....	372
10.2.1	Health Indications.....	372
10.2.2	Replication Error Messages	374
10.3	Cascading Considerations	374
10.4	Supported Subprofiles and Packages.....	374
10.5	Methods of the Profile	375
10.5.1	Intrinsic Methods of the Profile	375
10.5.2	Extrinsic Methods of the Profile	375
10.6	Client Considerations and Recipes.....	393
10.6.1	Discovery of Copy support and Capabilities.....	393
10.6.2	Creating and Managing Replicas	394
10.6.3	Using StorageSetting for Replicas.....	395
10.6.4	Finding and Creating Target Elements.....	395
10.6.5	Creating and Managing Pools for Delta Replicas.....	396
10.6.6	Creating and Managing Mirrors.....	397
10.6.7	Creating a Clone and Redirected Restore Operations	399
10.6.8	Creating and Managing Snapshots	399
10.6.9	Managing Background Copy	404
10.6.10	Recipes.....	405
10.6.11	Replica Modification	405
10.6.12	Replica Creation Or Attachment.....	408
10.7	CIM Elements.....	413
10.7.1	General.....	413
10.7.2	CIM_ElementCapabilities (Associates ReplicationServiceCapabilities and ReplicationService)	415
10.7.3	CIM_ElementCapabilities (Associates StorageReplicationCapabilities and StorageConfigurationService) ..	415
10.7.4	CIM_ElementCapabilities (StorageConfigurationCapabilities to StorageConfigurationService).....	416
10.7.5	CIM_ElementCapabilities (StorageConfigurationCapabilities to StoragePool) ...	416
10.7.6	CIM_HostedService (Replication Service)	416
10.7.7	CIM_HostedService (Storage Configuration Service)	417
10.7.8	CIM_ReplicaPoolForStorage.....	417
10.7.9	CIM_ReplicationService	417
10.7.10	CIM_ReplicationServiceCapabilities.....	418

10.7.11	CIM_ReplicationSettingData	419
10.7.12	CIM_SettingsDefineState	420
10.7.13	CIM_StorageCapabilities	421
10.7.14	CIM_StorageConfigurationCapabilities	421
10.7.15	CIM_StorageConfigurationService	422
10.7.16	CIM_StoragePool	423
10.7.17	CIM_StorageReplicationCapabilities	423
10.7.18	CIM_StorageSetting	425
10.7.19	CIM_StorageSynchronized	426
10.7.20	CIM_StorageSynchronized (Between StorageExtent elements)	427
10.7.21	CIM_SynchronizationAspect	429
11	Disk Drive Subprofile	430
12	Disk Drive Lite Subprofile	431
12.1	Description	431
12.1.1	General	431
12.1.2	Base model	431
12.1.3	Associations to external classes	431
12.1.4	Active Management	432
12.1.5	Diagram of CIM Elements	432
12.1.6	Durable Names and Correlatable IDs of the Profile	432
12.1.7	Conditional Associations to other profiles	432
12.1.8	Optional Associations to other profiles	433
12.2	Health and Fault Management Considerations	433
12.3	Cascading Considerations	434
12.4	Supported Profiles, Subprofiles and Packages	434
12.5	Methods of this Profile – Extrinsic Methods on Disk Drives	434
12.6	Registered Name and Version	434
12.7	CIM Elements	434
12.7.1	General	434
12.7.2	CIM_ATAPort (Disk Drive Target ATA Port)	437
12.7.3	CIM_ATAProtocolEndpoint (Disk Drive target ATA Protocol Endpoint)	437
12.7.4	CIM_AssociatedComponentExtent (Pool Component to Primordial Pool)	437
12.7.5	CIM_BasedOn (Bottom Level BasedOn)	438
12.7.6	CIM_ConcreteComponent (Disk Extent to Primordial Pool)	438
12.7.7	CIM_Container	439
12.7.8	CIM_DeviceSAPImplementation (ATA)	439
12.7.9	CIM_DeviceSAPImplementation (SCSI)	439
12.7.10	CIM_DiskDrive	440
12.7.11	CIM_ElementSoftwareIdentity	440
12.7.12	CIM_FCPort (Disk Drive Target FC Port)	441
12.7.13	CIM_FilterCollection (Disk Drive Lite Predefined FilterCollection)	441
12.7.14	CIM_HostedCollection (System to predefined IndicationFilters)	442
12.7.15	CIM_IndicationFilter (Disk Drive Creation)	442
12.7.16	CIM_IndicationFilter (Disk Drive Deletion)	443
12.7.17	CIM_MediaPresent	443
12.7.18	CIM_MemberOfCollection (Disk Drive Lite Filter Collection to FilterCollection) ..	444
12.7.19	CIM_MemberOfCollection (Predefined Filter Collection to Disk Drive Lite Filters)	444
12.7.20	CIM_PhysicalPackage	444
12.7.21	CIM_ProtocolControllerAccessesUnit	445
12.7.22	CIM_Realizes	445
12.7.23	CIM_SAPAvailableForElement	446
12.7.24	CIM_SASPort (Disk Drive Target SAS Port)	446
12.7.25	CIM_SCSIInitiatorTargetLogicalUnitPath	446

12.7.26	CIM_SCSIProtocolEndpoint (Disk Drive target SCSI Protocol Endpoint)	447
12.7.27	CIM_SPIPort (Disk Drive Target Parallel SCSI Port)	447
12.7.28	CIM_SoftwareIdentity	448
12.7.29	CIM_StorageExtent (Primordial Disk Drive Extent)	448
12.7.30	CIM_SystemDevice (Disk Drive System)	449
12.7.31	CIM_SystemDevice (Port System)	449
12.7.32	CIM_SystemDevice (Storage Extent System)	450
12.7.33	SNIA_DiskDrive	450
13	Disk Sparing Subprofile	451
13.1	Description	451
13.1.1	General	451
13.1.2	Durable Names and Correlatable IDs of the Profile	452
13.1.3	Sparing Model	452
13.1.4	Modeling Fail Over, Past and Present	454
13.1.5	Sparing Configuration and Control	456
13.2	Health and Fault Management Considerations	456
13.3	Cascading Conjunctions	457
13.4	Supported Subprofiles and Packages	457
13.5	Methods of the Profile	457
13.5.1	AssignSpares	457
13.5.2	UnassignSpares	458
13.5.3	GetAvailableSpareExtents	458
13.5.4	FailOver	458
13.5.5	RebuildStorageExtent	458
13.5.6	CheckParityConsistency	459
13.5.7	RepairParity	460
13.5.8	CheckStorageElement	460
13.6	Client Considerations and Recipes	460
13.6.1	General	460
13.6.2	Determine if spare model is constructed correctly	461
13.7	Registered Name and Version	462
13.8	CIM Elements	462
13.8.1	General	462
13.8.2	CIM_AssociatedComponentExtent (Spare to Storage Pool)	463
13.8.3	CIM_ConcreteDependency (Extent to LogicalDisk)	463
13.8.4	CIM_ConcreteDependency (Extent to Pool)	464
13.8.5	CIM_ConcreteDependency (Extent to StorageVolume)	464
13.8.6	CIM_ElementCapabilities	464
13.8.7	CIM_HostedCollection (ComputerSystem to FailoverStorageExtentsCollection)	464
13.8.8	CIM_HostedCollection (ComputerSystem to RedundancySet)	465
13.8.9	CIM_HostedService (ComputerSystem to SpareConfigurationService)	465
13.8.10	CIM_IsSpare	465
13.8.11	CIM_LogicalDisk	466
13.8.12	CIM_MemberOfCollection	466
13.8.13	CIM_Spared	467
13.8.14	CIM_StorageExtent (Spare)	467
13.8.15	CIM_StoragePool	468
13.8.16	CIM_StorageRedundancySet	468
13.8.17	CIM_StorageVolume	468
13.8.18	SNIA_FailoverStorageExtentsCollection	469
13.8.19	SNIA_SpareConfigurationCapabilities	469
13.8.20	SNIA_SpareConfigurationService	470
14	Erase Profile	471
14.1	Description	471

14.1.1	General.....	471
14.1.2	Existing Erasure standards.....	471
14.2	Health and Fault Management Considerations.....	473
14.3	Cascading Considerations.....	473
14.4	Supported Profiles, Subprofiles, and Packages.....	473
14.5	Methods of the Profile.....	473
14.6	Client Considerations and Recipes.....	474
14.6.1	General.....	474
14.6.2	Recipe 1: Volume Erasure.....	474
14.6.3	Recipe 2: Volume Deletion.....	476
14.7	Registered Name and Version.....	477
14.8	CIM Elements.....	478
14.8.1	General.....	478
14.8.2	CIM_AllocatedFromStoragePool.....	478
14.8.3	CIM_LogicalDisk.....	478
14.8.4	CIM_StoragePool.....	479
14.8.5	CIM_StorageVolume.....	479
14.8.6	SNIA_ErasureCapabilities.....	479
14.8.7	SNIA_ErasureService.....	480
14.8.8	SNIA_ErasureSetting.....	480
15	Extent Composition Subprofile.....	481
15.1	Description.....	481
15.1.1	General.....	481
15.1.2	Decomposition.....	481
15.1.3	Composition.....	481
15.1.4	Model Element Summary.....	481
15.1.5	Relation to other Packages and Subprofiles.....	482
15.1.6	Remaining Extents.....	483
15.1.7	Scenarios.....	484
15.2	Health and Fault Management Considerations.....	497
15.3	Cascading Considerations.....	497
15.4	Supported Subprofiles and Packages.....	497
15.5	Methods of the Profile.....	497
15.6	Client Considerations and Recipes.....	498
15.6.1	Traverse the virtualization hierarchy of a StorageVolume or LogicalDisk.....	498
15.6.2	Find the Primordial Extents used by a Storage Volume or Logical Disk.....	502
15.7	Registered Name and Version.....	503
15.8	CIM Elements.....	504
15.8.1	General.....	504
15.8.2	CIM_AssociatedComponentExtent (Pool Component to Concrete Pool).....	505
15.8.3	CIM_AssociatedRemainingExtent (Pool to its remaining extents).....	505
15.8.4	CIM_BasedOn (Mid level BasedOn).....	506
15.8.5	CIM_BasedOn (Top level BasedOn).....	506
15.8.6	CIM_CompositeExtent (Composite Intermediate).....	506
15.8.7	CIM_CompositeExtent (Composite Pool Component).....	507
15.8.8	CIM_CompositeExtentBasedOn.....	508
15.8.9	CIM_ConcreteComponent (Pool Component to Concrete Pool).....	508
15.8.10	CIM_ConcreteComponent (Remaining Extent to Pool).....	509
15.8.11	CIM_FilterCollection (Extent Composition Predefined FilterCollection).....	509
15.8.12	CIM_HostedCollection (System to predefined IndicationFilters).....	510
15.8.13	CIM_MemberOfCollection (Extent Composition Filter Collection to FilterCollection).....	510
15.8.14	CIM_MemberOfCollection (Predefined Filter Collection to Extent Composition Filters).....	510
15.8.15	CIM_StorageExtent (Intermediate).....	511

15.8.16	CIM_StorageExtent (Pool Component)	511
15.8.17	CIM_StorageExtent (Remaining).....	512
15.8.18	CIM_SystemDevice (Composite Extent System)	513
15.8.19	CIM_SystemDevice (Storage Extent System).....	513
16	LUN Creation Subprofile.....	514
17	Extent Mapping Subprofile	515
18	LUN Mapping and Masking Subprofile	516
18.1	General	516
18.2	Compatibility with SMI-S 1.0 clients	516
19	Masking and Mapping Subprofile	517
19.1	Description	517
19.1.1	General.....	517
19.1.2	Views and Paths.....	517
19.1.3	Model Elements.....	518
19.1.4	SCSIProtocolController Views.....	519
19.1.5	Initiator ID Collections.....	520
19.1.6	Default View / Default Logical Unit Access.....	521
19.1.7	Arbitrary Logical Units	521
19.1.8	Read-only versus Read-Write access	521
19.1.9	Read-Only Volumes	521
19.1.10	Finding Volumes that are not Mapped.....	521
19.1.11	Limits on Map counts per Logical Unit.....	521
19.1.12	Deactivated Logical Units	522
19.1.13	SCSIProtocolController Properties	522
19.1.14	Initiator Setting Data.....	522
19.1.15	Durable Names and Correlatable IDs of the Profile	524
19.1.16	Instrumentation Requirements	524
19.1.17	Element Naming.....	525
19.2	Health and Fault Management Considerations.....	525
19.3	Cascading Considerations	525
19.4	Supported Subprofiles, and Packages.....	526
19.5	Methods of the Profile	526
19.5.1	ExposePaths	526
19.5.2	HidePaths.....	528
19.5.3	ExposeDefaultLUs.....	530
19.5.4	HideDefaultLUs	532
19.5.5	CreateStorageHardwareID	534
19.5.6	DeleteStorageHardwareID	534
19.5.7	CreateHardwareIDCollection.....	534
19.5.8	AddHardwareIDsToCollection	535
19.5.9	DeleteProtocolController	535
19.6	Client Considerations and Recipes	536
19.6.1	Expose and Hide LUNs	536
19.6.2	Set Host Mode for a Port.....	543
19.6.3	Set Host Mode for a ProtocolController.....	544
19.7	Registered Name and Version	545
19.8	CIM Elements.....	545
19.8.1	General.....	545
19.8.2	CIM_AuthorizedPrivilege	547
19.8.3	CIM_AuthorizedSubject.....	548
19.8.4	CIM_AuthorizedTarget	548
19.8.5	CIM_ConcreteDependency (Associates ControllerConfigurationService and ProtocolController)	548
19.8.6	CIM_ConcreteDependency	

	(Associates PrivilegeManagementService and AuthorizedPrivilege)	549
19.8.7	CIM_ConcreteDependency (Associates StorageHardwareIDManagementService and StorageHardwareID)	549
19.8.8	CIM_ConcreteDependency (Associates StorageHardwareIDManagementService and SystemSpecificCollection)	550
19.8.9	CIM_ControllerConfigurationService	550
19.8.10	CIM_ElementCapabilities (EnabledLogicalElementCapabilities to ControllerConfigurationService)	550
19.8.11	CIM_ElementCapabilities (EnabledLogicalElementCapabilities to ProtocolController)	551
19.8.12	CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StorageHardwareID)	551
19.8.13	CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StorageHardwareIDManagementService)	551
19.8.14	CIM_ElementCapabilities (EnabledLogicalElementCapabilities to SystemSpecificCollection)	552
19.8.15	CIM_ElementCapabilities (System to ProtocolControllerMaskingCapabilities) ...	552
19.8.16	CIM_ElementSettingData (Associates ComputerSystem and StorageClientSettingData)	552
19.8.17	CIM_ElementSettingData (Associates Port and StorageClientSettingData)	553
19.8.18	CIM_ElementSettingData (Associates ProtocolController and StorageClientSettingData)	553
19.8.19	CIM_ElementSettingData (Associates StorageHardwareID and StorageClientSettingData)	553
19.8.20	CIM_EnabledLogicalElementCapabilities	554
19.8.21	CIM_HostedCollection	554
19.8.22	CIM_HostedService (Associates ComputerSystem and ControllerConfigurationService)	555
19.8.23	CIM_HostedService (Associates ComputerSystem and PrivilegeManagementService)	555
19.8.24	CIM_HostedService (Associates ComputerSystem and StorageHardwareIDManagementService)	555
19.8.25	CIM_MemberOfCollection	556
19.8.26	CIM_PrivilegeManagementService	556
19.8.27	CIM_ProtocolController	556
19.8.28	CIM_ProtocolControllerForUnit	557
19.8.29	CIM_ProtocolControllerMaskingCapabilities	557
19.8.30	CIM_SAPAvailableForElement	558
19.8.31	CIM_StorageClientSettingData	559
19.8.32	CIM_StorageHardwareID	559
19.8.33	CIM_StorageHardwareIDManagementService	559
19.8.34	CIM_SystemSpecificCollection	560
19.8.35	SNIA_ProtocolControllerMaskingCapabilities	560
19.8.36	SNIA_StorageHardwareID	561
19.8.37	SNIA_StorageHardwareIDManagementService	561
20	Pool Manipulation Capabilities, and Settings Subprofile	562
21	Storage Server Asymmetry Profile	563
21.1	Description	563
21.1.1	Overview	563
21.1.2	Relationship to Multiple Computer System Subprofile	563
21.1.3	Relationship to Masking and Mapping Subprofile	563
21.1.4	Relationship to T10	564
21.1.5	Behavior, Characteristics, and Capabilities	564
21.1.6	Model	565
21.2	Health and Fault Management Consideration	571

21.3	Cascading Considerations	571
21.4	Supported Profiles, Subprofiles, and Packages.....	571
21.5	Methods of the Profile	571
21.5.1	Assign Storage Resource Affinity	571
21.6	Client Considerations and Recipes	572
21.6.1	Determine which ports provide full bandwidth access to a storage element	572
21.7	Registered Name and Version	574
21.8	CIM Elements.....	574
21.8.1	General.....	574
21.8.2	CIM_AsymmetricAccessibility.....	576
21.8.3	CIM_ElementCapabilities (To Top-level ComputerSystem)	577
21.8.4	CIM_HostedCollection (Top-Level System to Load Group)	577
21.8.5	CIM_HostedCollection (Top-Level System to Port Group).....	578
21.8.6	CIM_MemberOfCollection (SATA Target Port Group)	578
21.8.7	CIM_MemberOfCollection (SB Target Port Group)	578
21.8.8	CIM_MemberOfCollection (SCSI Target Port Group)	579
21.8.9	CIM_MemberOfCollection (Storage Resource Load Group aggregating Storage Pools)	579
21.8.10	CIM_MemberOfCollection (Storage Resource Load Group aggregating Storage Volumes)	579
21.8.11	CIM_MemberOfCollection (iSCSI Target Port Group).....	580
21.8.12	CIM_StorageConfigurationService	580
21.8.13	CIM_StorageProcessorAffinity (StorageResourceLoadGroup)	581
21.8.14	CIM_StorageProcessorAffinity (Target Port Group)	581
21.8.15	CIM_StorageResourceLoadGroup (Load Groups).....	582
21.8.16	CIM_StorageServerAsymmetryCapabilities	582
21.8.17	CIM_TargetPortGroup (Port Groups)	583
22	Block Services Resource Ownership Subprofile	585
22.1	Description	585
22.1.1	General.....	585
22.1.2	Design considerations	588
22.1.3	Privilege Propagation	590
22.2	Client Considerations and Recipes	590
23	Storage Virtualizer Profile	592
23.1	Description	592
23.1.1	General.....	592
23.1.2	Instance Diagram	592
23.1.3	Storage Virtualization System	593
23.1.4	Disk Drive Lite	594
23.1.5	Controller Software.....	594
23.1.6	Device Management Access.....	594
23.1.7	Physical Modeling.....	594
23.1.8	Services.....	594
23.1.9	Ports	594
23.1.10	Model Element Summary	594
23.2	Health and Fault Management.....	595
23.3	Storage Virtualizer Support for Cascading.....	595
23.4	Supported Subprofiles and Packages.....	596
23.5	Methods of the Profile	598
23.6	Client Considerations and Recipes	598
23.7	Registered Name and Version	598
23.8	CIM Elements.....	598
23.8.1	General.....	598
23.8.2	CIM_AssociatedComponentExtent (Pool Component to Primordial Pool)	602

23.8.3	CIM_ComputerSystem (Shadow).....	602
23.8.4	CIM_ComputerSystem (Top Level System).....	603
23.8.5	CIM_ConcreteComponent (Imported Extents to Primordial Pool).....	604
23.8.6	CIM_Dependency (Systems).....	604
23.8.7	CIM_FilterCollection (Storage Virtualizer Predefined FilterCollection)	604
23.8.8	CIM_HostedCollection (Allocated Resources)	605
23.8.9	CIM_HostedCollection (Remote Resources).....	605
23.8.10	CIM_HostedCollection (Storage Virtualizer to predefined FilterCollection)	606
23.8.11	CIM_IndicationFilter (Storage Virtualizer LogicalPort OperationalStatus).....	606
23.8.12	CIM_IndicationFilter (Storage Virtualizer Storage Volume OperationalStatus) ...	607
23.8.13	CIM_IndicationFilter (Storage Virtualizer System Creation)	608
23.8.14	CIM_IndicationFilter (Storage Virtualizer System Deletion)	609
23.8.15	CIM_IndicationFilter (Storage Virtualizer System OperationalStatus).....	610
23.8.16	CIM_IndicationFilter (WQL Storage Virtualizer FCPort OperationalStatus)	611
23.8.17	CIM_IndicationFilter (WQL Storage Virtualizer Storage Volume OperationalStatus).....	612
23.8.18	CIM_IndicationFilter (WQL Storage Virtualizer System OperationalStatus).....	613
23.8.19	CIM_LogicalIdentity (Shadow Storage Volume).....	614
23.8.20	CIM_MemberOfCollection (Allocated Resources).....	614
23.8.21	CIM_MemberOfCollection (Predefined Filter Collection to Storage Virtualizer Filters)	615
23.8.22	CIM_MemberOfCollection (Remote Resources).....	615
23.8.23	CIM_ProtocolControllerForUnit (Arbitrary LU for All LUNs View).....	615
23.8.24	CIM_ProtocolControllerForUnit (Storage volumes for All LUNs View)	616
23.8.25	CIM_RemoteServiceAccessPoint (Shadow)	616
23.8.26	CIM_SAPAvailableForElement.....	617
23.8.27	CIM_SCSIArbitraryLogicalUnit (Arbitrary LU).....	617
23.8.28	CIM_SCSIProtocolController (All LUNs View).....	618
23.8.29	CIM_StorageExtent (Imported Extents).....	618
23.8.30	CIM_StorageVolume (Shadow).....	618
23.8.31	CIM_SystemDevice (Shadow StorageVolumes).....	620
23.8.32	CIM_SystemDevice (System to SCSIArbitraryLogicalUnit).....	620
23.8.33	CIM_SystemDevice (System to SCSIProtocolController)	621
23.8.34	CIM_SystemDevice (System to StorageExtent).....	621
23.8.35	SNIA_AllocatedResources	621
23.8.36	SNIA_RemoteResources	622
24	Volume Composition Profile	623
24.1	Description	623
24.1.1	Overview.....	623
24.1.2	Relationship to Block Services Package	623
24.1.3	Relationship to Extent Composition.....	623
24.1.4	Model.....	624
24.1.5	Quality of Service (QoS) Considerations.....	625
24.1.6	Composite Stripe Length and Depth	625
24.1.7	Examples.....	626
24.2	Striped and Concatenated Composite Volumes	634
24.3	Health and Fault Management Consideration.....	635
24.4	Cascading Considerations	635
24.5	Supported Profiles, Subprofiles, and Packages.....	636
24.6	Methods of the Profile	636
24.6.1	General.....	636
24.6.2	CreateOrModifyCompositeElement.....	636
24.6.3	RemoveElementsFromElement.....	639
24.6.4	ReturnElementToElements	640
24.6.5	GetAvailableElements	641

24.6.6	GetCompositeElements.....	642
24.6.7	GetSupportedStripeLengths.....	643
24.6.8	GetSupportedStripeLengthRange.....	643
24.6.9	GetSupportedStripeDepths.....	644
24.6.10	GetSupportedStripeDepthRange.....	644
24.7	Client Considerations and Recipes.....	645
24.7.1	Indications.....	645
24.7.2	Recipe 1: Create Composite Volume.....	646
24.7.3	Recipe 2: Delete Composite Volume.....	648
24.8	Registered Name and Version.....	650
24.9	CIM Elements.....	650
24.9.1	General.....	650
24.9.2	CIM_CompositeExtent.....	651
24.9.3	CIM_CompositeExtentBasedOn (Volume Composition).....	651
24.9.4	CIM_ElementCapabilities.....	652
24.9.5	CIM_ElementSettingData.....	652
24.9.6	CIM_HostedService (Associates ComputerSystem and the ElementCompositionService).....	652
24.9.7	CIM_StorageElementCompositionCapabilities.....	652
24.9.8	CIM_StorageElementCompositionService.....	653
24.9.9	CIM_StorageSetting.....	654
24.9.10	CIM_StorageVolume.....	655
25	Volume Management Profile.....	657
25.1	Description.....	657
25.1.1	General.....	657
25.1.2	Instance Diagram.....	657
25.1.3	Input Class of the Volume Manager.....	657
25.1.4	Export Class of the Volume Manager.....	658
25.1.5	Initializing OS Disks for Volume Manager Use.....	658
25.1.6	Creating Pools and Logical Volumes.....	658
25.1.7	Storage Settings for Volumes.....	658
25.1.8	Durable Names and Other Correlatable ids of the Profile.....	658
25.2	Health and Fault Management Considerations.....	658
25.3	Cascading Considerations.....	658
25.4	Supported Subprofiles and Packages.....	659
25.5	Methods of the Profile.....	659
25.6	Client Considerations and Recipes.....	659
25.6.1	General.....	659
25.6.2	Storage Configuration.....	659
25.7	Registered Name and Version.....	659
25.8	CIM Elements.....	660
25.8.1	General.....	660
25.8.2	CIM_AllocatedFromStoragePool (LogicalDisk from Pool).....	660
25.8.3	CIM_AllocatedFromStoragePool (Pool from Pool).....	661
25.8.4	CIM_ComputerSystem.....	661
25.8.5	CIM_ElementCapabilities.....	662
25.8.6	CIM_ElementSettingData.....	662
25.8.7	CIM_HostedStoragePool.....	662
25.8.8	CIM_LogicalDisk.....	663
25.8.9	CIM_StorageCapabilities.....	663
25.8.10	CIM_StoragePool (Concrete).....	664
25.8.11	CIM_StoragePool (Primordial).....	664
25.8.12	CIM_StorageSetting.....	665
25.8.13	CIM_SystemDevice.....	665

26	Storage Element Protection SubProfile	667
26.1	Description	667
26.1.1	Overview.....	667
26.1.2	Use Cases	667
26.1.3	Functionality	667
26.1.4	Class Model.....	667
26.1.5	Access permission.....	669
26.1.6	Retention period	671
26.1.7	Protection State Transition	672
26.1.8	Sample Usage Scenario.....	672
26.2	Health and Fault Management Consideration.....	677
26.3	Cascading Considerations	677
26.4	Supported Profiles, Subprofiles, and Packages.....	677
26.5	Methods of the Profile – Protect.....	677
26.6	Client Considerations and Recipes	679
26.6.1	Start Volume Protection.....	679
26.6.2	Extend the Retention Period.....	681
26.7	Registered Name and Version	682
26.8	CIM Elements.....	683
26.8.1	General.....	683
26.8.2	CIM_ElementCapabilities	683
26.8.3	CIM_HostedService.....	683
26.8.4	SNIA_ElementProtectionSettingData	684
26.8.5	SNIA_StorageProtectionCapabilities.....	684
26.8.6	SNIA_StorageProtectionService	684
26.8.7	SNIA_StorageProtectionSetting	685
27	Replication Services Profile.....	686
27.1	Description	686
27.1.1	Synopsis.....	686
27.1.2	Supported Profiles, Subprofiles, and Packages	686
27.1.3	Overview.....	686
27.1.4	Key Features	687
27.1.5	Replication Services and Copy Services Profiles.....	687
27.1.6	Key Components	687
27.1.7	Replication Services Discovery	688
27.1.8	Replication Services Capabilities	688
27.1.9	SyncTypes.....	689
27.1.10	Modes.....	689
27.1.11	Locality of Target Elements	689
27.1.12	Remote Replication	693
27.1.13	Undiscovered Resources	693
27.1.14	Multi-hop Replication	694
27.1.15	Groups.....	695
27.1.16	Associations	697
27.1.17	Operations on List of Synchronizations	701
27.1.18	State Management For Associated Replicas	701
27.1.19	Unsynchronized and Skewed CopyStates	705
27.1.20	Accessibility to Associations and Elements.....	705
27.1.21	Host Access Restrictions.....	705
27.1.22	Deleting the Target Elements.....	705
27.1.23	Completion of Long Operations.....	705
27.1.24	Managing Background Copy	706
27.1.25	Managing CopyPriority	706
27.1.26	Using StorageSettings for Replicas.....	706

27.1.27	Finding and Creating Target Elements.....	707
27.1.28	Using StoragePools (e.g. ResourcePools) for Replicas.....	707
27.1.29	Provider Configurations for Remote Replication	709
27.1.30	Thinly Provisioned Elements	709
27.1.31	Indications	710
27.2	Health and Fault Management Consideration.....	711
27.3	Replication Services Support for Cascading.....	711
27.3.1	General.....	711
27.3.2	ServiceAccessPoint and SharedSecret Instances	712
27.3.3	Cascading Support.....	712
27.4	Mapping of Copy Services and Replication Services Properties and Methods	714
27.5	Methods of the Profile	714
27.5.1	Overview.....	714
27.5.2	Replication Services and Copy Services Properties and Methods Mapping.....	742
27.6	Client Considerations and Recipes – Creating and Managing Replicas.....	743
27.7	Registered Name and Version.....	743
27.8	CIM Elements.....	744
27.8.1	General.....	744
27.8.2	CIM_ConnectivityCollection.....	747
27.8.3	CIM_ElementCapabilities	748
27.8.4	CIM_GroupSynchronized	748
27.8.5	CIM_HostedAccessPoint (ForProtocolEndpoint).....	751
27.8.6	CIM_HostedAccessPoint (ForRemoteServiceAccessPoint).....	751
27.8.7	CIM_HostedCollection (Allocated Resources)	751
27.8.8	CIM_HostedCollection (Between ComputerSystem and ConnectivityCollection)	752
27.8.9	CIM_HostedCollection (Between ComputerSystem and ReplicationGroup).....	752
27.8.10	CIM_HostedCollection (Remote Resources).....	753
27.8.11	CIM_HostedService.....	753
27.8.12	CIM_MemberOfCollection (Allocated Resources).....	753
27.8.13	CIM_MemberOfCollection (ProtocolEndpoints to ConnectivityCollection)	754
27.8.14	CIM_MemberOfCollection (Remote Resources)	754
27.8.15	CIM_OrderedMemberOfCollection	754
27.8.16	CIM_ProtocolEndpoint.....	755
27.8.17	CIM_RemoteServiceAccessPoint.....	755
27.8.18	CIM_ReplicaPoolForStorage.....	756
27.8.19	CIM_ReplicationEntity	756
27.8.20	CIM_ReplicationGroup	757
27.8.21	CIM_ReplicationSettingData	758
27.8.22	CIM_SAPAvailableForElement.....	759
27.8.23	CIM_ServiceAffectsElement (Between ReplicationService and ConnectivityCollection)	760
27.8.24	CIM_ServiceAffectsElement (Between ReplicationService and ReplicationEntity).....	760
27.8.25	CIM_ServiceAffectsElement (Between ReplicationService and ReplicationGroup).....	760
27.8.26	CIM_SettingsDefineState (Between ReplicationGroup and SynchronizationAspect)	761
27.8.27	CIM_SettingsDefineState (Between storage object and SynchronizationAspect)	761
27.8.28	CIM_SharedSecret.....	761
27.8.29	CIM_StorageSynchronized.....	762
27.8.30	CIM_SynchronizationAspect	765
27.8.31	SNIA_AllocatedResources	765
27.8.32	SNIA_RemoteResources	766

27.8.33	SNIA_ReplicationService	767
27.8.34	SNIA_ReplicationServiceCapabilities	768
28	Thin Provisioning Profile	772
28.1	Description	772
28.1.1	Background	772
28.1.2	Model	772
28.2	Health and Fault Management Consideration	775
28.3	Cascading Considerations	775
28.4	Supported Profiles, Subprofiles, and Packages	775
28.5	Methods of the Profile	775
28.6	Client Considerations and Recipes	776
28.6.1	Create a Pool from a Parent Pool	776
28.6.2	Create a Pool from Extents	781
28.6.3	Creating a Thinly Provisioned Volume	781
28.6.4	Capacity Properties for fully provisioned RAID1 Volume	786
28.6.5	Capacity Properties for Thin Provisioning	788
28.7	Registered Name and Version	789
28.8	CIM Elements	789
28.8.1	General	789
28.8.2	CIM_HostedStoragePool	790
28.8.3	SNIA_LogicalDisk	790
28.8.4	SNIA_StorageConfigurationCapabilities (Concrete)	791
28.8.5	SNIA_StorageConfigurationCapabilities (Global)	792
28.8.6	SNIA_StorageConfigurationCapabilities (Primordial)	793
28.8.7	SNIA_StorageConfigurationService	794
28.8.8	SNIA_StoragePool (Concrete)	795
28.8.9	SNIA_StoragePool (Empty)	796
28.8.10	SNIA_StoragePool (Primordial)	797
28.8.11	SNIA_StorageSetting	798
28.8.12	SNIA_StorageVolume	799
29	Pools from Volumes Profile	802
29.1	Description	802
29.1.1	Overview	802
29.1.2	Terminology	802
29.1.3	Relationship to Block Services Package	802
29.1.4	Relationship to Extent Composition	802
29.1.5	Class Model	802
29.1.6	Model Elements	803
29.1.7	Example	803
29.2	Block Services Enhancements	806
29.2.1	General	806
29.2.2	StoragePool Manipulation Methods	806
29.2.3	Declaring Storage Configuration Options	806
29.2.4	The Usage Property	807
29.3	Health and Fault Management Considerations	807
29.4	Cascading Considerations	807
29.5	Supported Profiles, Subprofiles, and Packages	807
29.6	Methods of the Profile	807
29.6.1	General	807
29.6.2	CreateOrModifyStoragePool	807
29.6.3	DeleteStoragePool	808
29.6.4	Storage Element Modification	808
29.7	Client Considerations and Recipes	808
29.7.1	Client Considerations	808

29.7.2	Recipe 1: Create StoragePool.....	808
29.8	Registered Name and Version	811
29.9	CIM Elements.....	811
29.9.1	General.....	811
29.9.2	CIM_AllocatedFromStoragePool (Volume from Pool)	812
29.9.3	CIM_ElementCapabilities	812
29.9.4	CIM_StorageCapabilities.....	812
29.9.5	CIM_StorageVolume	813
29.9.6	CIM_SystemDevice	813
29.9.7	SNIA_StorageConfigurationCapabilities.....	813
29.9.8	SNIA_StoragePool	813
29.9.9	SNIA_StorageSetting	814
30	Group Masking and Mapping Profile	815
30.1	Description	815
30.1.1	Synopsis.....	815
30.1.2	Overview.....	815
30.1.3	Model Elements.....	816
30.1.4	Device Numbers.....	820
30.1.5	Group Masking and Mapping Capabilities.....	822
30.2	Health and Fault Management Consideration.....	823
30.3	Cascading Considerations	823
30.4	Methods of the Profile	823
30.4.1	General.....	823
30.4.2	CreateGroup.....	823
30.4.3	DeleteGroup	824
30.4.4	AddMembers	825
30.4.5	RemoveMembers	825
30.4.6	CreateMaskingView.....	825
30.4.7	DeleteMaskingView	826
30.4.8	ModifyMaskingView.....	826
30.5	Client Considerations and Recipes	827
30.5.1	Using Groups in Masking and Mapping.....	827
30.6	Registered Name and Version	828
30.7	CIM Elements.....	828
30.7.1	General.....	828
30.7.2	CIM_AssociatedDeviceMaskingGroup	830
30.7.3	CIM_AssociatedInitiatorMaskingGroup	831
30.7.4	CIM_AssociatedTargetMaskingGroup.....	831
30.7.5	CIM_AuthorizedPrivilege	831
30.7.6	CIM_AuthorizedSubject.....	832
30.7.7	CIM_AuthorizedTarget	832
30.7.8	CIM_ConcreteDependency (Associates ControllerConfigurationService and ProtocolController)	833
30.7.9	CIM_ConcreteDependency (Associates PrivilegeManagementService and AuthorizedPrivilege).....	833
30.7.10	CIM_ConcreteDependency (Associates StorageHardwareIDManagementService and StorageHardwareID).....	833
30.7.11	CIM_ConcreteDependency (Associates StorageHardwareIDManagementService and SystemSpecificCollection)	834
30.7.12	CIM_DeviceMaskingGroup.....	834
30.7.13	CIM_ElementCapabilities (EnabledLogicalElementCapabilities to ControllerConfigurationService).....	834
30.7.14	CIM_ElementCapabilities (EnabledLogicalElementCapabilities to ProtocolController).....	835
30.7.15	CIM_ElementCapabilities (EnabledLogicalElementCapabilities	

to StorageHardwareID)	835
30.7.16 CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StorageHardwareIDManagementService)	835
30.7.17 CIM_ElementCapabilities (EnabledLogicalElementCapabilities to SystemSpecificCollection)	836
30.7.18 CIM_ElementCapabilities (System to ProtocolControllerMaskingCapabilities) ...	836
30.7.19 CIM_ElementSettingData (Associates ComputerSystem and StorageClientSettingData)	836
30.7.20 CIM_ElementSettingData (Associates Port and StorageClientSettingData)	837
30.7.21 CIM_ElementSettingData (Associates ProtocolController and StorageClientSettingData)	837
30.7.22 CIM_ElementSettingData (Associates StorageHardwareID and StorageClientSettingData)	837
30.7.23 CIM_EnabledLogicalElementCapabilities	838
30.7.24 CIM_GroupMaskingMappingCapabilities	838
30.7.25 CIM_GroupMaskingMappingService	841
30.7.26 CIM_HostedCollection	842
30.7.27 CIM_HostedService (Associates ComputerSystem and ControllerConfigurationService)	843
30.7.28 CIM_HostedService (Associates ComputerSystem and PrivilegeManagementService)	843
30.7.29 CIM_HostedService (Associates ComputerSystem and StorageHardwareIDManagementService)	843
30.7.30 CIM_InitiatorMaskingGroup	844
30.7.31 CIM_MemberOfCollection	844
30.7.32 CIM_PrivilegeManagementService	845
30.7.33 CIM_ProtocolController	845
30.7.34 CIM_ProtocolControllerForUnit	845
30.7.35 CIM_SAPAvailableForElement	846
30.7.36 CIM_ServiceAffectsElement (Between GroupMaskingMappingService and MaskingGroup)	846
30.7.37 CIM_StorageClientSettingData	847
30.7.38 CIM_StorageHardwareID	847
30.7.39 CIM_StorageHardwareIDManagementService	847
30.7.40 CIM_SystemSpecificCollection	848
30.7.41 CIM_TargetMaskingGroup	848
30.7.42 SNIA_ProtocolControllerMaskingCapabilities	849
30.7.43 SNIA_StorageHardwareID	849
30.7.44 SNIA_StorageHardwareIDManagementService	850
Annex A (informative) SMI-S Information Model	851
Annex B (informative) Registry of StorageExtent Definitions	852
Bibliography	858

Figure 1 - Experimental Maturity Level Tag	46
Figure 2 - Implemented Maturity Level Tag	47
Figure 3 - Stable Maturity Level Tag	47
Figure 4 - Deprecated Tag	48
Figure 5 - Array Profile Instance Diagram	49
Figure 6 - Array Package Diagram	50
Figure 7 - Storage Capacity State	61
Figure 8 - StoragePool Manipulation Instance Diagram	63
Figure 9 - Capabilities Specific to a StoragePool	64
Figure 10 - StorageVolume Creation Instance Diagram	70
Figure 11 - Storage Configuration	72
Figure 12 - StorageExtent Conservation - Step 1	78
Figure 13 - StorageExtent Conservation - Step 2	79
Figure 14 - StorageExtent Conservation - Step 3	80
Figure 15 - Block Services Predefined FilterCollection	83
Figure 16 - Representative Block Service Instance Diagram	99
Figure 17 - StoragePool Creation - Initial State	100
Figure 18 - StoragePool Creation - Step 1	101
Figure 19 - StoragePool Creation - Step 2	101
Figure 20 - StoragePool Creation - Step 3	102
Figure 21 - StorageVolume Creation - Initial State	102
Figure 22 - StorageVolume Creation - Step 1	103
Figure 23 - StorageVolume Creation - Step 2	104
Figure 24 - StorageVolume Creation - Step 3	105
Figure 25 - Class Diagram for SNIA_ View Classes	168
Figure 26 - Block Storage View Class Capabilities	169
Figure 27 - SNIA_ VolumeView and related associations	170
Figure 28 - SNIA_ DiskDriveView and related associations	172
Figure 29 - SNIA_ ExposedView Association	174
Figure 30 - SNIA_ MaskingMapView Association	175
Figure 31 - The SNIA_ MappingProtocolControllerView	176
Figure 32 - The SNIA_ StoragePoolView	178
Figure 33 - The SNIA_ ReplicaPairView	181
Figure 34 - Block Server Performance Subprofile Summary Instance Diagram	219
Figure 35 - Base Array Profile Block Server Performance Instance Diagram	222
Figure 36 - Base Storage Virtualizer Profile Block Server Performance Instance Diagram	224
Figure 37 - Base Volume Management Profile Block Server Performance Instance Diagram	226
Figure 38 - Multiple Computer System Subprofile Block Server Performance Instance Diagram ..	229
Figure 39 - Fibre Channel Initiator Port Subprofile Block Server Performance Instance Diagram ..	230
Figure 40 - Extent Composition Subprofile Block Server Performance Instance Diagram	231
Figure 41 - Disk Drive Lite Subprofile Block Server Performance Instance Diagram	232
Figure 42 - SCSIArbitraryLogicalUnit Block Server Performance Instance Diagram	233
Figure 43 - Remote Mirrors Block Server Performance Instance Diagram	234
Figure 44 - Block Server Performance Manifest Collections	236
Figure 45 - Block Services Support for Count Key Data Storage	296
Figure 46 - Copy Services Discovery	338
Figure 47 - Local Replica	342
Figure 48 - Multi-Level Local Replication	343
Figure 49 - Multiple Snapshots Per Source Element	344

Figure 50 - SettingsDefineState Association	348
Figure 51 - SynchronizationAspect Instance	349
Figure 52 - State Transitions for Mirrors and Clones	357
Figure 53 - State Transitions for Snapshots and Migration.....	358
Figure 54 - CopyState Transitions	360
Figure 55 - Sample CopyState and ProgressStatus Transitions	365
Figure 56 - Fixed Space Consumption	369
Figure 57 - Variable Space Consumption	370
Figure 58 - Fixed Space Consumption	400
Figure 59 - Variable Space Consumption	401
Figure 60 - CIM Elements in the Disk Drive Model	432
Figure 61 - Sparing Instance Diagram	451
Figure 62 - Variations of RS per Storage Element.....	453
Figure 63 - During Failure	455
Figure 64 - Before Failure	455
Figure 65 - After Failure	456
Figure 66 - Model Elements	473
Figure 67 - Remaining Extents in Extent Composition	483
Figure 68 - Volume Composition from General QOS Pool	485
Figure 69 - Single QOS Pool Composition (RAID Groups).....	486
Figure 70 - Single QOS Pool Composition - Two Concretes.....	487
Figure 71 - Concatenation Composition	489
Figure 72 - RAID0 Composition	489
Figure 73 - RAID1 Composition	490
Figure 74 - RAID10 Composition	491
Figure 75 - RAID0+1 Composition	492
Figure 76 - RAID4, 5 Composition	493
Figure 77 - RAID 6, 5DP, 4DP	494
Figure 78 - RAID15 Composition	495
Figure 79 - RAID50 Composition	496
Figure 80 - RAID51 Composition	497
Figure 81 - Generic System with no Configuration Service	519
Figure 82 - Generic System with ControllerConfigurationService.....	519
Figure 83 - Relationship of Initiator IDs, Endpoints, and Logical Units	520
Figure 84 - StorageClientSettingData Model	523
Figure 85 - Entire Model	524
Figure 86 - Storage Asymmetry Class Hierarchy.....	565
Figure 87 - Asymmetry with MCS	567
Figure 88 - Ports Do Not Failover, Healthy	568
Figure 89 - Ports Do Not Failover, Failed Controller	569
Figure 90 - Ports Failover, Healthy	570
Figure 91 - Ports Failover, Failed Controller	571
Figure 92 - Resource Ownership for Block Services	586
Figure 93 - ServiceAffectsElement Associations for ResourceOwnership	589
Figure 94 - AuthorizedPrivilege Associations for ResourceOwnership.....	590
Figure 95 - Storage Virtualizer Package Diagram	592
Figure 96 - Storage Virtualizer System Instance	593
Figure 97 - Virtualizer, Cascading and Initiator Ports	596
Figure 98 - Volume Composition Class Mode	624

Figure 99 - Example 1 Step 1	627
Figure 100 - Example 1 Step 2	628
Figure 101 - First Alternative Example - Before Composition	629
Figure 102 - First Alternative Example - After Composition	630
Figure 103 - Second Alternative Example - Before Composition	631
Figure 104 - Second Alternative Example - After Composition	632
Figure 105 - Example 2 - Before Composition	633
Figure 106 - Example 2 - After Composition	634
Figure 107 - Striping and Concatenation	635
Figure 108 - Volume Management Instance Diagram	658
Figure 109 - Storage Element Protection Class Model	668
Figure 110 - Retention Time Line	671
Figure 111 - Protection State Transition Diagram	672
Figure 112 - Step 1 - Initial State	673
Figure 113 - Step 2 - Volume Set to Read-only	674
Figure 114 - Step 3 - Second Volume Set to Read-only	675
Figure 115 - Step 4 - Volume Set to Read/Write Disabled	676
Figure 116 - Step 5 Volume Access Changed	677
Figure 117 - Replication Services Discovery	688
Figure 118 - Local Replica	690
Figure 119 - Remote Replica	691
Figure 120 - Remote Replication over two Paths	692
Figure 121 - Expanded Remote Replica	693
Figure 122 - An instance of ReplicationEntity	694
Figure 123 - StorageSynchronized and ReplicationEntity	694
Figure 124 - Multi-hop Replication	695
Figure 125 - Group Instances	696
Figure 126 - Sequentially Consistent Example	697
Figure 127 - Associated Groups and Elements	698
Figure 128 - SettingsDefineState Association	699
Figure 129 - SynchronizationAspect Instance	700
Figure 130 - One-to-Many Association	701
Figure 131 - CopyState Transitions	703
Figure 132 - Sample CopyState and ProgressStatus Transitions	704
Figure 133 - Fixed Space Consumption	708
Figure 134 - Variable Space Consumption	709
Figure 135 - Instance Diagram for Access to shadow Resources	712
Figure 136 - Instance of ServiceAccessPoint	712
Figure 137 - Replication Services support for Cascading	713
Figure 138 - Cascading and Replication Groups	714
Figure 139 - Thin Provisioning	773
Figure 140 - RAID1 Capacity after Volume Creation	787
Figure 141 - RAID1 Capacity with Thin Volume and RAID-at-Pool Approach	788
Figure 142 - RAID1 Capacity with Thin Volume and RAID-at-Volume Approach	789
Figure 143 - Class Model	802
Figure 144 - Before Pool Creation	804
Figure 145 - After Pool Creation	805
Figure 146 - After Pool Creation without Extent Composition	806
Figure 147 - Group Masking and Mapping Model	817

Figure 148 - Masking Groups	818
Figure 149 - Nested Masking Groups	819
Figure 150 - Nested Masking Group Example	820
Figure 151 - Example ConsistentLogicalUnitNumber set to true	821
Figure 152 - Example ConsistentLogicalUnitNumber set to false	822

IECNORM.COM : Click to view the full PDF of ISO/IEC 24775-4:2014

Table 1 - Supported Profiles for Array	51
Table 2 - CIM Elements for Array	53
Table 3 - SMI Referenced Properties/Methods for CIM_ComputerSystem (Top Level System)	54
Table 4 - SMI Referenced Properties/Methods for CIM_FilterCollection (Array Predefined FilterCollection)	55
Table 5 - SMI Referenced Properties/Methods for CIM_HostedCollection (Array to predefined FilterCollection)	55
Table 6 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Array System Creation)	56
Table 7 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Array System Deletion)	57
Table 8 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Predefined Filter Collection to Array Filters)	57
Table 9 - SMI Referenced Properties/Methods for CIM_ProtocolControllerForUnit (Arbitrary LU for All LUNs View)	58
Table 10 - SMI Referenced Properties/Methods for CIM_ProtocolControllerForUnit (Storage volumes for All LUNs View)	58
Table 11 - SMI Referenced Properties/Methods for CIM_SCSIArbitraryLogicalUnit (Arbitrary LU)	59
Table 12 - SMI Referenced Properties/Methods for CIM_SCSIProtocolController (All LUNs View)	59
Table 13 - SMI Referenced Properties/Methods for CIM_SystemDevice (System to SCSIArbitraryLogicalUnit)	59
Table 14 - SMI Referenced Properties/Methods for CIM_SystemDevice (System to SCSIProtocolController)	60
Table 15 - Mapping: Supported Actions to Methods	67
Table 16 - Valid Values for StorageConfigurationCapabilities associated to a Pool	68
Table 17 - SupportedStoragePoolFeatures Array	69
Table 18 - SupportedStoragePoolFeatures Array	69
Table 19 - RAID Mapping	73
Table 20 - Meaning of Usage values	75
Table 21 - Classes Required In Read-Only Implementation	76
Table 22 - Standard Messages for Block Services Package	84
Table 23 - Supported Profiles for Block Services	84
Table 24 - CIM Elements for Block Services	126
Table 25 - SMI Referenced Properties/Methods for CIM_AllocatedFromStoragePool (Pool from Pool)	132
Table 26 - SMI Referenced Properties/Methods for CIM_AllocatedFromStoragePool (Volume or LogicalDisk from Pool)	133
Table 27 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StorageVolume or LogicalDisk)	133
Table 28 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StoragePool)	133
Table 29 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (StorageCapabilities to StorageConfigurationService)	134
Table 30 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (StorageCapabilities to StoragePool)	134
Table 31 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (StorageConfigurationCapabilities to StorageConfigurationService)	134
Table 32 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (StorageConfigurationCapabilities to concrete StoragePool)	135
Table 33 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (StorageConfigurationCapabilities to primordial StoragePool)	135
Table 34 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (Used to declare the naming capabilities of the StoragePool)	135
Table 35 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (Used to declare the naming capabilities of the StorageVolume or LogicalDisk)	136
Table 36 - SMI Referenced Properties/Methods for CIM_ElementSettingData	136
Table 37 - SMI Referenced Properties/Methods for CIM_EnabledLogicalElementCapabilities (For StorageConfigurationService)	137

Table 38 - SMI Referenced Properties/Methods for CIM_EnabledLogicalElementCapabilities (For StoragePool)	137
Table 39 - SMI Referenced Properties/Methods for CIM_FilterCollection (Block Services Predefined FilterCollection)	138
Table 40 - SMI Referenced Properties/Methods for CIM_HostedCollection (System to predefined IndicationFilters)	138
Table 41 - SMI Referenced Properties/Methods for CIM_HostedService	139
Table 42 - SMI Referenced Properties/Methods for CIM_HostedStoragePool	139
Table 43 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Logical Disk Creation)	139
Table 44 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Logical Disk Deletion)	140
Table 45 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Logical Disk OperationalStatus)	141
Table 46 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Storage Pool Creation)	142
Table 47 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Storage Pool Deletion)	143
Table 48 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Storage Pool TotalManagedSpace)	144
Table 49 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Storage Volume Creation)	145
Table 50 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Storage Volume Deletion)	146
Table 51 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Storage Volume OperationalStatus)	147
Table 52 - SMI Referenced Properties/Methods for CIM_IndicationFilter (WQL Logical Disk OperationalStatus)	148
Table 53 - SMI Referenced Properties/Methods for CIM_IndicationFilter (WQL Storage Volume OperationalStatus)	149
Table 54 - SMI Referenced Properties/Methods for CIM_LogicalDisk	149
Table 55 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Block Services Filter Collection to FilterCollection)	150
Table 56 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Predefined Filter Collection to Block Services Filters)	151
Table 57 - SMI Referenced Properties/Methods for CIM_OwningJobElement	151
Table 58 - SMI Referenced Properties/Methods for CIM_StorageCapabilities	151
Table 59 - SMI Referenced Properties/Methods for CIM_StorageConfigurationCapabilities (Concrete)	153
Table 60 - SMI Referenced Properties/Methods for CIM_StorageConfigurationCapabilities (Global)	154
Table 61 - SMI Referenced Properties/Methods for CIM_StorageConfigurationCapabilities (Primordial)	155
Table 62 - SMI Referenced Properties/Methods for CIM_StorageConfigurationService	156
Table 63 - SMI Referenced Properties/Methods for CIM_StoragePool (Concrete)	157
Table 64 - SMI Referenced Properties/Methods for CIM_StoragePool (Empty)	157
Table 65 - SMI Referenced Properties/Methods for CIM_StoragePool (Primordial)	158
Table 66 - SMI Referenced Properties/Methods for CIM_StorageSetting	159
Table 67 - SMI Referenced Properties/Methods for CIM_StorageSettingWithHints	160
Table 68 - SMI Referenced Properties/Methods for CIM_StorageSettingsAssociatedToCapabilities	161
Table 69 - SMI Referenced Properties/Methods for CIM_StorageSettingsGeneratedFromCapabilities	162
Table 70 - SMI Referenced Properties/Methods for CIM_StorageVolume	162
Table 71 - SMI Referenced Properties/Methods for CIM_SystemDevice (System to StorageVolume or LogicalDisk)	164
Table 72 - SMI Referenced Properties/Methods for SNIA_StorageVolume	165
Table 73 - Related Profiles for Block Storage Views	166
Table 74 - Discovery of the Volumes on an Array	183
Table 75 - Discovery of the Disk Drives in a Primordial Pool	183
Table 76 - Discover Volumes exposed on a (Target) Port	184
Table 77 - Discover (target port) redundancy for a Volume	184
Table 78 - Discover Volumes exposed to a Host Port	184

Table 79 - Discover Mapping information for an array	185
Table 80 - Discover the Pool topology for an array	185
Table 81 - Discover the Replica Pairs for an array	186
Table 82 - CIM Elements for Block Storage Views	186
Table 83 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (View Capabilities)	191
Table 84 - SMI Referenced Properties/Methods for SNIA_AllocatedFromStoragePoolView (StoragePoolView to StoragePool)	192
Table 85 - SMI Referenced Properties/Methods for SNIA_AllocatedFromStoragePoolView (Volume to StoragePoolView)	192
Table 86 - SMI Referenced Properties/Methods for SNIA_AllocatedFromStoragePoolView (VolumeView to StoragePool)	193
Table 87 - SMI Referenced Properties/Methods for SNIA_AllocatedFromStoragePoolViewView (PoolView to PoolView)	193
Table 88 - SMI Referenced Properties/Methods for SNIA_AllocatedFromStoragePoolViewView (VolumeView to PoolView)	194
Table 89 - SMI Referenced Properties/Methods for SNIA_BaseInstance (DiskDrive)	194
Table 90 - SMI Referenced Properties/Methods for SNIA_BaseInstance (StorageSetting)	195
Table 91 - SMI Referenced Properties/Methods for SNIA_BaseInstance (Volume)	195
Table 92 - SMI Referenced Properties/Methods for SNIA_BasedOnView (ExtentOnDriveExtent)	196
Table 93 - SMI Referenced Properties/Methods for SNIA_BasedOnView (VolumeOnExtent)	196
Table 94 - SMI Referenced Properties/Methods for SNIA_ConcreteComponentView	197
Table 95 - SMI Referenced Properties/Methods for SNIA_ContainerView	197
Table 96 - SMI Referenced Properties/Methods for SNIA_DiskDriveView	198
Table 97 - SMI Referenced Properties/Methods for SNIA_DriveComponentViewView	199
Table 98 - SMI Referenced Properties/Methods for SNIA_ElementStatisticalDataView (DiskDriveView)	200
Table 99 - SMI Referenced Properties/Methods for SNIA_ElementStatisticalDataView (VolumeView)	200
Table 100 - SMI Referenced Properties/Methods for SNIA_ExposedView	201
Table 101 - SMI Referenced Properties/Methods for SNIA_ExtentComponentView	201
Table 102 - SMI Referenced Properties/Methods for SNIA_HostedStoragePoolView	202
Table 103 - SMI Referenced Properties/Methods for SNIA_MappingProtocolControllerView	202
Table 104 - SMI Referenced Properties/Methods for SNIA_MaskingMapView	204
Table 105 - SMI Referenced Properties/Methods for SNIA_ProtocolControllerForUnitView	205
Table 106 - SMI Referenced Properties/Methods for SNIA_ReplicaPairView	205
Table 107 - SMI Referenced Properties/Methods for SNIA_StoragePoolView	208
Table 108 - SMI Referenced Properties/Methods for SNIA_SystemDeviceView (DiskDriveViews)	210
Table 109 - SMI Referenced Properties/Methods for SNIA_SystemDeviceView (MappingProtocolControllerViews)	211
Table 110 - SMI Referenced Properties/Methods for SNIA_SystemDeviceView (ReplicaPairViews)	211
Table 111 - SMI Referenced Properties/Methods for SNIA_SystemDeviceView (VolumeViews)	211
Table 112 - SMI Referenced Properties/Methods for SNIA_ViewCapabilities	212
Table 113 - SMI Referenced Properties/Methods for SNIA_VolumeView	213
Table 114 - Related Profiles for Block Server Performance	216
Table 115 - Summary of Element Types by Profile	227
Table 116 - Creation, Deletion and Modification Methods in Block Server Performance Subprofile	238
Table 117 - Summary of Statistics Support by Element	267
Table 118 - Formulas and Calculations	269
Table 119 - Block Server Performance Subprofile Supported Capabilities Patterns	269
Table 120 - CIM Elements for Block Server Performance	270
Table 121 - SMI Referenced Properties/Methods for CIM_AssociatedBlockStatisticsManifestCollection (Client defined collection)	273
Table 122 - SMI Referenced Properties/Methods for CIM_AssociatedBlockStatisticsManifestCollection (Provider defined collection)	274

Table 123 - SMI Referenced Properties/Methods for CIM_BlockStatisticsCapabilities	274
Table 124 - SMI Referenced Properties/Methods for CIM_BlockStatisticsManifest (Client Defined).....	275
Table 125 - SMI Referenced Properties/Methods for CIM_BlockStatisticsManifest (Provider Support)	276
Table 126 - SMI Referenced Properties/Methods for CIM_BlockStatisticsManifestCollection (Client Defined).....	278
Table 127 - SMI Referenced Properties/Methods for CIM_BlockStatisticsManifestCollection (Provider Defined)	278
Table 128 - SMI Referenced Properties/Methods for CIM_BlockStatisticsService	279
Table 129 - SMI Referenced Properties/Methods for CIM_BlockStorageStatisticalData.....	281
Table 130 - SMI Referenced Properties/Methods for CIM_ElementCapabilities	284
Table 131 - SMI Referenced Properties/Methods for CIM_ElementStatisticalData (Back end Port Stats)	284
Table 132 - SMI Referenced Properties/Methods for CIM_ElementStatisticalData (Component System Stats)	285
Table 133 - SMI Referenced Properties/Methods for CIM_ElementStatisticalData (Disk Stats)	285
Table 134 - SMI Referenced Properties/Methods for CIM_ElementStatisticalData (Extent Stats)	286
Table 135 - SMI Referenced Properties/Methods for CIM_ElementStatisticalData (Front end Port Stats)	286
Table 136 - SMI Referenced Properties/Methods for CIM_ElementStatisticalData (Logical Disk Stats).....	287
Table 137 - SMI Referenced Properties/Methods for CIM_ElementStatisticalData (Remote Copy Stats)	287
Table 138 - SMI Referenced Properties/Methods for CIM_ElementStatisticalData (Top Level System Stats)	288
Table 139 - SMI Referenced Properties/Methods for CIM_ElementStatisticalData (Volume Stats)	288
Table 140 - SMI Referenced Properties/Methods for CIM_HostedCollection (Client Defined).....	289
Table 141 - SMI Referenced Properties/Methods for CIM_HostedCollection (Default)	289
Table 142 - SMI Referenced Properties/Methods for CIM_HostedCollection (Provider Supplied)	289
Table 143 - SMI Referenced Properties/Methods for CIM_HostedService.....	290
Table 144 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Member of client defined collection)	290
Table 145 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Member of pre-defined collection).....	291
Table 146 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Member of statistics collection)	291
Table 147 - SMI Referenced Properties/Methods for CIM_StatisticsCollection.....	291
Table 148 - SMI Referenced Properties/Methods for SNIA_BlockStatisticsCapabilities	292
Table 149 - SMI Referenced Properties/Methods for SNIA_BlockStatisticsManifest (Client Defined).....	292
Table 150 - SMI Referenced Properties/Methods for SNIA_BlockStatisticsManifest (Provider Support)	294
Table 151 - Supported Profiles for CKD Block Services	298
Table 152 - CIM Elements for CKD Block Services	299
Table 153 - SMI Referenced Properties/Methods for CIM_AllocatedFromStoragePool (Pool from Pool).....	305
Table 154 - SMI Referenced Properties/Methods for CIM_AllocatedFromStoragePool (Volume or LogicalDisk from Pool)	306
Table 155 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StorageVolume or LogicalDisk)	306
Table 156 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StoragePool)	307
Table 157 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (StorageCapabilities to StorageConfigurationService)	307
Table 158 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (StorageCapabilities to StoragePool)	307

Table 159 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (StorageConfigurationCapabilities to StorageConfigurationService).....	308
Table 160 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (StorageConfigurationCapabilities to concrete StoragePool)	308
Table 161 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (StorageConfigurationCapabilities to primordial StoragePool)	308
Table 162 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (Used to declare the naming capabilities of the StoragePool).....	309
Table 163 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (Used to declare the naming capabilities of the StorageVolume or LogicalDisk)	309
Table 164 - SMI Referenced Properties/Methods for CIM_EnabledLogicalElementCapabilities (For StorageConfigurationService).....	310
Table 165 - SMI Referenced Properties/Methods for CIM_EnabledLogicalElementCapabilities (For StoragePool)	310
Table 166 - SMI Referenced Properties/Methods for CIM_FilterCollection (Block Services Predefined FilterCollection)	311
Table 167 - SMI Referenced Properties/Methods for CIM_HostedCollection (System to predefined IndicationFilters)	311
Table 168 - SMI Referenced Properties/Methods for CIM_HostedStoragePool.....	311
Table 169 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Logical Disk Creation)	312
Table 170 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Logical Disk Deletion).....	313
Table 171 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Logical Disk OperationalStatus)	314
Table 172 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Storage Pool Creation)	315
Table 173 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Storage Pool Deletion)	316
Table 174 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Storage Pool TotalManagedSpace)	317
Table 175 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Storage Volume Creation)	318
Table 176 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Storage Volume Deletion)	319
Table 177 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Storage Volume OperationalStatus).....	320
Table 178 - SMI Referenced Properties/Methods for CIM_IndicationFilter (WQL Logical Disk OperationalStatus).....	321
Table 179 - SMI Referenced Properties/Methods for CIM_IndicationFilter (WQL Storage Volume OperationalStatus)	322
Table 180 - SMI Referenced Properties/Methods for CIM_LogicalDisk.....	322
Table 181 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Block Services Filter Collection to FilterCollection)	323
Table 182 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Predefined Filter Collection to Block Services Filters).....	324
Table 183 - SMI Referenced Properties/Methods for CIM_OwningJobElement.....	324
Table 184 - SMI Referenced Properties/Methods for CIM_StorageConfigurationCapabilities (Concrete)	325
Table 185 - SMI Referenced Properties/Methods for CIM_StorageConfigurationCapabilities (Global).....	326
Table 186 - SMI Referenced Properties/Methods for CIM_StorageConfigurationCapabilities (Primordial)	326
Table 187 - SMI Referenced Properties/Methods for CIM_StoragePool (Concrete)	328
Table 188 - SMI Referenced Properties/Methods for CIM_StoragePool (Empty).....	328
Table 189 - SMI Referenced Properties/Methods for CIM_StoragePool (Primordial).....	329
Table 190 - SMI Referenced Properties/Methods for CIM_SystemDevice (System to StorageVolume or LogicalDisk)	330
Table 191 - SMI Referenced Properties/Methods for SNIA_StorageCapabilities	330
Table 192 - SMI Referenced Properties/Methods for SNIA_StorageSetting	332
Table 193 - SMI Referenced Properties/Methods for SNIA_StorageVolume	333
Table 194 - SMI Referenced Properties/Methods for SNIA_StorageVolume	335

Table 195 - Related Profiles for Copy Services	336
Table 196 - Comparing SyncTypes.....	340
Table 197 - Alignment of SupportedSynchronizationType and SupportedReplicationType	341
Table 198 - Alignment of SyncType/Mode and CopyType.....	345
Table 199 - Alignment of CopyState and SyncState.....	345
Table 200 - Synchronization Operation Support Requirements.....	352
Table 201 - SyncState Values.....	353
Table 202 - CopyStates Values	355
Table 203 - SyncMaintained and WhenSynced Properties.....	356
Table 204 - Indications.....	371
Table 205 - Copy Services Alert Indications	373
Table 206 - Copy Services Error Responses.....	374
Table 207 - Extrinsic Methods of StorageConfigurationService.....	376
Table 208 - ModifySynchronization.....	376
Table 209 - CreateReplica Method	377
Table 210 - TargetPool Parameter for Delta Replicas	378
Table 211 - Extrinsic Methods of ReplicationService.....	380
Table 212 - GetAvailableTargetElements Method	384
Table 213 - Extrinsic Methods of ReplicationServiceCapabilities	386
Table 214 - SyncTypes	387
Table 215 - Modes	387
Table 216 - Local or Remote.....	387
Table 217 - ReplicationTypes	388
Table 218 - Features.....	388
Table 219 - Operations	389
Table 220 - Comparison of Similar Operations	391
Table 221 - SettingsDefineState Operations.....	391
Table 222 - Thin Provisioning Features	392
Table 223 - Components.....	392
Table 224 - Replica Specialization by CopyType.....	393
Table 225 - Replica Specialization by SyncType/Mode	394
Table 226 - Patterns Supported for StorageReplicationCapabilities.....	401
Table 227 - Space Consumption Properties	403
Table 228 - Space Consumption Properties, Fixed Pattern.....	403
Table 229 - CIM Elements for Copy Services	413
Table 230 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (Associates ReplicationServiceCapabilities and ReplicationService).....	415
Table 231 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (Associates StorageReplicationCapabilities and StorageConfigurationService).....	416
Table 232 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (StorageConfigurationCapabilities to StorageConfigurationService).....	416
Table 233 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (StorageConfigurationCapabilities to StoragePool)	416
Table 234 - SMI Referenced Properties/Methods for CIM_HostedService (Replication Service)	417
Table 235 - SMI Referenced Properties/Methods for CIM_HostedService (Storage Configuration Service)	417
Table 236 - SMI Referenced Properties/Methods for CIM_ReplicaPoolForStorage.....	417
Table 237 - SMI Referenced Properties/Methods for CIM_ReplicationService	418
Table 238 - SMI Referenced Properties/Methods for CIM_ReplicationServiceCapabilities.....	418
Table 239 - SMI Referenced Properties/Methods for CIM_ReplicationSettingData	420
Table 240 - SMI Referenced Properties/Methods for CIM_SettingsDefineState	421
Table 241 - SMI Referenced Properties/Methods for CIM_StorageCapabilities	421
Table 242 - SMI Referenced Properties/Methods for CIM_StorageConfigurationCapabilities	422
Table 243 - SMI Referenced Properties/Methods for CIM_StorageConfigurationService	422

Table 244 - SMI Referenced Properties/Methods for CIM_StoragePool	423
Table 245 - SMI Referenced Properties/Methods for CIM_StorageReplicationCapabilities	423
Table 246 - SMI Referenced Properties/Methods for CIM_StorageSetting	425
Table 247 - SMI Referenced Properties/Methods for CIM_StorageSynchronized.....	426
Table 248 - SMI Referenced Properties/Methods for CIM_StorageSynchronized (Between StorageExtent elements).....	428
Table 249 - SMI Referenced Properties/Methods for CIM_SynchronizationAspect	429
Table 250 - OperationalStatus For DiskDrive	433
Table 251 - Enabled State	434
Table 252 - CIM Elements for Disk Drive Lite	434
Table 253 - SMI Referenced Properties/Methods for CIM_ATAPort (Disk Drive Target ATA Port).....	437
Table 254 - SMI Referenced Properties/Methods for CIM_ATAProtocolEndpoint (Disk Drive target ATA Protocol Endpoint)	437
Table 255 - SMI Referenced Properties/Methods for CIM_AssociatedComponentExtent (Pool Component to Primordial Pool)	438
Table 256 - SMI Referenced Properties/Methods for CIM_BasedOn (Bottom Level BasedOn).....	438
Table 257 - SMI Referenced Properties/Methods for CIM_ConcreteComponent (Disk Extent to Primordial Pool).....	438
Table 258 - SMI Referenced Properties/Methods for CIM_Container	439
Table 259 - SMI Referenced Properties/Methods for CIM_DeviceSAPImplementation (ATA).....	439
Table 260 - SMI Referenced Properties/Methods for CIM_DeviceSAPImplementation (SCSI)	439
Table 261 - SMI Referenced Properties/Methods for CIM_DiskDrive.....	440
Table 262 - SMI Referenced Properties/Methods for CIM_ElementSoftwareIdentity	441
Table 263 - SMI Referenced Properties/Methods for CIM_FCPort (Disk Drive Target FC Port)	441
Table 264 - SMI Referenced Properties/Methods for CIM_FilterCollection (Disk Drive Lite Predefined FilterCollection).....	442
Table 265 - SMI Referenced Properties/Methods for CIM_HostedCollection (System to predefined IndicationFilters).....	442
Table 266 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Disk Drive Creation)	442
Table 267 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Disk Drive Deletion).....	443
Table 268 - SMI Referenced Properties/Methods for CIM_MediaPresent.....	444
Table 269 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Disk Drive Lite Filter Collection to FilterCollection)	444
Table 270 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Predefined Filter Collection to Disk Drive Lite Filters)	444
Table 271 - SMI Referenced Properties/Methods for CIM_PhysicalPackage.....	445
Table 272 - SMI Referenced Properties/Methods for CIM_ProtocolControllerAccessesUnit	445
Table 273 - SMI Referenced Properties/Methods for CIM_Realizes	445
Table 274 - SMI Referenced Properties/Methods for CIM_SAPAvailableForElement.....	446
Table 275 - SMI Referenced Properties/Methods for CIM_SASPort (Disk Drive Target SAS Port)	446
Table 276 - SMI Referenced Properties/Methods for CIM_SCSIInitiatorTargetLogicalUnitPath	447
Table 277 - SMI Referenced Properties/Methods for CIM_SCSIProtocolEndpoint (Disk Drive target SCSI Protocol Endpoint).....	447
Table 278 - SMI Referenced Properties/Methods for CIM_SPIPort (Disk Drive Target Parallel SCSI Port)	448
Table 279 - SMI Referenced Properties/Methods for CIM_SoftwareIdentity	448
Table 280 - SMI Referenced Properties/Methods for CIM_StorageExtent (Primordial Disk Drive Extent)	449
Table 281 - SMI Referenced Properties/Methods for CIM_SystemDevice (Disk Drive System)	449
Table 282 - SMI Referenced Properties/Methods for CIM_SystemDevice (Port System)	450
Table 283 - SMI Referenced Properties/Methods for CIM_SystemDevice (Storage Extent System)	450
Table 284 - SMI Referenced Properties/Methods for SNIA_DiskDrive.....	450
Table 285 - Supported Methods to Method Mapping.....	454
Table 286 - Supported Profiles for Disk Sparing.....	457

Table 287 - CIM Elements for Disk Sparing	462
Table 288 - SMI Referenced Properties/Methods for CIM_AssociatedComponentExtent (Spare to Storage Pool)	463
Table 289 - SMI Referenced Properties/Methods for CIM_ConcreteDependency (Extent to LogicalDisk)	463
Table 290 - SMI Referenced Properties/Methods for CIM_ConcreteDependency (Extent to Pool)	464
Table 291 - SMI Referenced Properties/Methods for CIM_ConcreteDependency (Extent to StorageVolume)	464
Table 292 - SMI Referenced Properties/Methods for CIM_ElementCapabilities	464
Table 293 - SMI Referenced Properties/Methods for CIM_HostedCollection (ComputerSystem to FailoverStorageExtentsCollection)	465
Table 294 - SMI Referenced Properties/Methods for CIM_HostedCollection (ComputerSystem to RedundancySet)	465
Table 295 - SMI Referenced Properties/Methods for CIM_HostedService (ComputerSystem to SpareConfigurationService)	465
Table 296 - SMI Referenced Properties/Methods for CIM_IsSpare	466
Table 297 - SMI Referenced Properties/Methods for CIM_LogicalDisk	466
Table 298 - SMI Referenced Properties/Methods for CIM_MemberOfCollection	467
Table 299 - SMI Referenced Properties/Methods for CIM_Spared	467
Table 300 - SMI Referenced Properties/Methods for CIM_StorageExtent (Spare)	467
Table 301 - SMI Referenced Properties/Methods for CIM_StoragePool	468
Table 302 - SMI Referenced Properties/Methods for CIM_StorageRedundancySet	468
Table 303 - SMI Referenced Properties/Methods for CIM_StorageVolume	469
Table 304 - SMI Referenced Properties/Methods for SNIA_FailoverStorageExtentsCollection	469
Table 305 - SMI Referenced Properties/Methods for SNIA_SpareConfigurationCapabilities	469
Table 306 - SMI Referenced Properties/Methods for SNIA_SpareConfigurationService	470
Table 307 - Erase Method	473
Table 308 - CIM Elements for Erasure	478
Table 309 - SMI Referenced Properties/Methods for CIM_AllocatedFromStoragePool	478
Table 310 - SMI Referenced Properties/Methods for CIM_LogicalDisk	478
Table 311 - SMI Referenced Properties/Methods for CIM_StoragePool	479
Table 312 - SMI Referenced Properties/Methods for CIM_StorageVolume	479
Table 313 - SMI Referenced Properties/Methods for SNIA_EraseCapabilities	479
Table 314 - SMI Referenced Properties/Methods for SNIA_EraseService	480
Table 315 - SMI Referenced Properties/Methods for SNIA_EraseSetting	480
Table 316 - Supported Common RAID Levels	488
Table 317 - CIM Elements for Extent Composition	504
Table 318 - SMI Referenced Properties/Methods for CIM_AssociatedComponentExtent (Pool Component to Concrete Pool)	505
Table 319 - SMI Referenced Properties/Methods for CIM_AssociatedRemainingExtent (Pool to its remaining extents)	505
Table 320 - SMI Referenced Properties/Methods for CIM_BasedOn (Mid level BasedOn)	506
Table 321 - SMI Referenced Properties/Methods for CIM_BasedOn (Top level BasedOn)	506
Table 322 - SMI Referenced Properties/Methods for CIM_CompositeExtent (Composite Intermediate)	507
Table 323 - SMI Referenced Properties/Methods for CIM_CompositeExtent (Composite Pool Component)	507
Table 324 - SMI Referenced Properties/Methods for CIM_CompositeExtentBasedOn	508
Table 325 - SMI Referenced Properties/Methods for CIM_ConcreteComponent (Pool Component to Concrete Pool)	509
Table 326 - SMI Referenced Properties/Methods for CIM_ConcreteComponent (Remaining Extent to Pool)	509
Table 327 - SMI Referenced Properties/Methods for CIM_FilterCollection (Extent Composition Predefined FilterCollection)	510

Table 328 - SMI Referenced Properties/Methods for CIM_HostedCollection (System to predefined IndicationFilters)	510
Table 329 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Extent Composition Filter Collection to FilterCollection)	510
Table 330 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Predefined Filter Collection to Extent Composition Filters)	511
Table 331 - SMI Referenced Properties/Methods for CIM_StorageExtent (Intermediate)	511
Table 332 - SMI Referenced Properties/Methods for CIM_StorageExtent (Pool Component)	512
Table 333 - SMI Referenced Properties/Methods for CIM_StorageExtent (Remaining)	512
Table 334 - SMI Referenced Properties/Methods for CIM_SystemDevice (Composite Extent System)	513
Table 335 - SMI Referenced Properties/Methods for CIM_SystemDevice (Storage Extent System)	513
Table 336 - SCSIProtocolController Property Description	522
Table 337 - Element to Service Mapping	525
Table 338 - Element to Element Name Mapping	525
Table 339 - ExposePath Use Cases	527
Table 340 - HidePaths Use Cases	529
Table 341 - Use Cases for ExposeDefaultLUs	531
Table 342 - Use Cases for HideDefaultLUs	533
Table 343 - CIM Elements for Masking and Mapping	545
Table 344 - SMI Referenced Properties/Methods for CIM_AuthorizedPrivilege	548
Table 345 - SMI Referenced Properties/Methods for CIM_AuthorizedSubject	548
Table 346 - SMI Referenced Properties/Methods for CIM_AuthorizedTarget	548
Table 347 - SMI Referenced Properties/Methods for CIM_ConcreteDependency (Associates ControllerConfigurationService and ProtocolController)	549
Table 348 - SMI Referenced Properties/Methods for CIM_ConcreteDependency (Associates PrivilegeManagementService and AuthorizedPrivilege)	549
Table 349 - SMI Referenced Properties/Methods for CIM_ConcreteDependency (Associates StorageHardwareIDManagementService and StorageHardwareID)	549
Table 350 - SMI Referenced Properties/Methods for CIM_ConcreteDependency (Associates StorageHardwareIDManagementService and SystemSpecificCollection)	550
Table 351 - SMI Referenced Properties/Methods for CIM_ControllerConfigurationService	550
Table 352 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (EnabledLogicalElementCapabilities to ControllerConfigurationService)	551
Table 353 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (EnabledLogicalElementCapabilities to ProtocolController)	551
Table 354 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StorageHardwareID)	551
Table 355 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StorageHardwareIDManagementService)	552
Table 356 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (EnabledLogicalElementCapabilities to SystemSpecificCollection)	552
Table 357 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (System to ProtocolControllerMaskingCapabilities)	552
Table 358 - SMI Referenced Properties/Methods for CIM_ElementSettingData (Associates ComputerSystem and StorageClientSettingData)	553
Table 359 - SMI Referenced Properties/Methods for CIM_ElementSettingData (Associates Port and StorageClientSettingData)	553
Table 360 - SMI Referenced Properties/Methods for CIM_ElementSettingData (Associates ProtocolController and StorageClientSettingData)	553
Table 361 - SMI Referenced Properties/Methods for CIM_ElementSettingData (Associates StorageHardwareID and StorageClientSettingData)	554
Table 362 - SMI Referenced Properties/Methods for CIM_EnabledLogicalElementCapabilities	554
Table 363 - SMI Referenced Properties/Methods for CIM_HostedCollection	554
Table 364 - SMI Referenced Properties/Methods for CIM_HostedService	

(Associates ComputerSystem and ControllerConfigurationService)	555
Table 365 - SMI Referenced Properties/Methods for CIM_HostedService (Associates ComputerSystem and PrivilegeManagementService)	555
Table 366 - SMI Referenced Properties/Methods for CIM_HostedService (Associates ComputerSystem and StorageHardwareIDManagementService)	555
Table 367 - SMI Referenced Properties/Methods for CIM_MemberOfCollection	556
Table 368 - SMI Referenced Properties/Methods for CIM_PrivilegeManagementService	556
Table 369 - SMI Referenced Properties/Methods for CIM_ProtocolController	557
Table 370 - SMI Referenced Properties/Methods for CIM_ProtocolControllerForUnit	557
Table 371 - SMI Referenced Properties/Methods for CIM_ProtocolControllerMaskingCapabilities	558
Table 372 - SMI Referenced Properties/Methods for CIM_SAPAvailableForElement	559
Table 373 - SMI Referenced Properties/Methods for CIM_StorageClientSettingData	559
Table 374 - SMI Referenced Properties/Methods for CIM_StorageHardwareID	559
Table 375 - SMI Referenced Properties/Methods for CIM_StorageHardwareIDManagementService	560
Table 376 - SMI Referenced Properties/Methods for CIM_SystemSpecificCollection	560
Table 377 - SMI Referenced Properties/Methods for SNIA_ProtocolControllerMaskingCapabilities	560
Table 378 - SMI Referenced Properties/Methods for SNIA_StorageHardwareID	561
Table 379 - SMI Referenced Properties/Methods for SNIA_StorageHardwareIDManagementService	561
Table 380 - CIM Elements for Storage Server Asymmetry	574
Table 381 - SMI Referenced Properties/Methods for CIM_AsymmetricAccessibility	577
Table 382 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (To Top-level ComputerSystem)	577
Table 383 - SMI Referenced Properties/Methods for CIM_HostedCollection (Top-Level System to Load Group)	578
Table 384 - SMI Referenced Properties/Methods for CIM_HostedCollection (Top-Level System to Port Group)	578
Table 385 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (SATA Target Port Group)	578
Table 386 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (SB Target Port Group)	579
Table 387 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (SCSI Target Port Group)	579
Table 388 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Storage Resource Load Group aggregating Storage Pools)	579
Table 389 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Storage Resource Load Group aggregating Storage Volumes)	580
Table 390 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (iSCSI Target Port Group)	580
Table 391 - SMI Referenced Properties/Methods for CIM_StorageConfigurationService	580
Table 392 - SMI Referenced Properties/Methods for CIM_StorageProcessorAffinity (StorageResourceLoadGroup)	581
Table 393 - SMI Referenced Properties/Methods for CIM_StorageProcessorAffinity (Target Port Group)	582
Table 394 - SMI Referenced Properties/Methods for CIM_StorageServerAsymmetryCapabilities	583
Table 395 - Block Service Management Rights	586
Table 396 - Supported Profiles for Storage Virtualizer	596
Table 397 - CIM Elements for Storage Virtualizer	598
Table 398 - SMI Referenced Properties/Methods for CIM_AssociatedComponentExtent (Pool Component to Primordial Pool)	602
Table 399 - SMI Referenced Properties/Methods for CIM_ComputerSystem (Shadow)	602
Table 400 - SMI Referenced Properties/Methods for CIM_ComputerSystem (Top Level System)	603
Table 401 - SMI Referenced Properties/Methods for CIM_ConcreteComponent (Imported Extents to Primordial Pool)	604
Table 402 - SMI Referenced Properties/Methods for CIM_Dependency (Systems)	604
Table 403 - SMI Referenced Properties/Methods for CIM_FilterCollection (Storage Virtualizer Predefined FilterCollection)	605

Table 404 - SMI Referenced Properties/Methods for CIM_HostedCollection (Allocated Resources)	605
Table 405 - SMI Referenced Properties/Methods for CIM_HostedCollection (Remote Resources).....	606
Table 406 - SMI Referenced Properties/Methods for CIM_HostedCollection (Storage Virtualizer to predefined FilterCollection).....	606
Table 407 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Storage Virtualizer LogicalPort OperationalStatus).....	606
Table 408 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Storage Virtualizer Storage Volume OperationalStatus)	607
Table 409 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Storage Virtualizer System Creation)	608
Table 410 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Storage Virtualizer System Deletion)	609
Table 411 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Storage Virtualizer System OperationalStatus).....	610
Table 412 - SMI Referenced Properties/Methods for CIM_IndicationFilter (WQL Storage Virtualizer FCPort OperationalStatus)	611
Table 413 - SMI Referenced Properties/Methods for CIM_IndicationFilter (WQL Storage Virtualizer Storage Volume OperationalStatus).....	612
Table 414 - SMI Referenced Properties/Methods for CIM_IndicationFilter (WQL Storage Virtualizer System OperationalStatus).....	613
Table 415 - SMI Referenced Properties/Methods for CIM_LogicalIdentity (Shadow Storage Volume)	614
Table 416 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Allocated Resources)	614
Table 417 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Predefined Filter Collection to Storage Virtualizer Filters).....	615
Table 418 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Remote Resources)	615
Table 419 - SMI Referenced Properties/Methods for CIM_ProtocolControllerForUnit (Arbitrary LU for All LUNs View)	616
Table 420 - SMI Referenced Properties/Methods for CIM_ProtocolControllerForUnit (Storage volumes for All LUNs View)	616
Table 421 - SMI Referenced Properties/Methods for CIM_RemoteServiceAccessPoint (Shadow)	617
Table 422 - SMI Referenced Properties/Methods for CIM_SAPAvailableForElement.....	617
Table 423 - SMI Referenced Properties/Methods for CIM_SCSIArbitraryLogicalUnit (Arbitrary LU).....	617
Table 424 - SMI Referenced Properties/Methods for CIM_SCSIProtocolController (All LUNs View)	618
Table 425 - SMI Referenced Properties/Methods for CIM_StorageExtent (Imported Extents).....	618
Table 426 - SMI Referenced Properties/Methods for CIM_StorageVolume (Shadow).....	619
Table 427 - SMI Referenced Properties/Methods for CIM_SystemDevice (Shadow StorageVolumes).....	620
Table 428 - SMI Referenced Properties/Methods for CIM_SystemDevice (System to SCSIArbitraryLogicalUnit)	620
Table 429 - SMI Referenced Properties/Methods for CIM_SystemDevice (System to SCSIProtocolController).....	621
Table 430 - SMI Referenced Properties/Methods for CIM_SystemDevice (System to StorageExtent)	621
Table 431 - SMI Referenced Properties/Methods for SNIA_AllocatedResources	622
Table 432 - SMI Referenced Properties/Methods for SNIA_RemoteResources	622
Table 433 - CompositionCharacteristics Property.....	625
Table 434 - Supported Profiles for Volume Composition	636
Table 435 - Method Summary.....	636
Table 436 - CreateOrModifyCompositeElement	638
Table 437 - RemoveElementsFromElement	640
Table 438 - ReturnElementToElements.....	640

Table 439 - GetAvailableElements.....	641
Table 440 - GetCompositeElements.....	642
Table 441 - GetSupportedStripeLengths	643
Table 442 - GetSupportedStripeLengthRange.....	643
Table 443 - GetSupportedStripeDepths.....	644
Table 444 - GetSupportedStripeDepthRange	645
Table 445 - CIM Elements for Volume Composition	650
Table 446 - SMI Referenced Properties/Methods for CIM_CompositeExtent.....	651
Table 447 - SMI Referenced Properties/Methods for CIM_CompositeExtentBasedOn (Volume Composition)	651
Table 448 - SMI Referenced Properties/Methods for CIM_ElementCapabilities	652
Table 449 - SMI Referenced Properties/Methods for CIM_ElementSettingData	652
Table 450 - SMI Referenced Properties/Methods for CIM_HostedService (Associates ComputerSystem and the ElementCompositionService)	652
Table 451 - SMI Referenced Properties/Methods for CIM_StorageElementCompositionCapabilities.....	653
Table 452 - SMI Referenced Properties/Methods for CIM_StorageElementCompositionService	653
Table 453 - SMI Referenced Properties/Methods for CIM_StorageSetting	654
Table 454 - SMI Referenced Properties/Methods for CIM_StorageVolume	656
Table 455 - Supported Profiles for Volume Management	659
Table 456 - CIM Elements for Volume Management	660
Table 457 - SMI Referenced Properties/Methods for CIM_AllocatedFromStoragePool (LogicalDisk from Pool)	661
Table 458 - SMI Referenced Properties/Methods for CIM_AllocatedFromStoragePool (Pool from Pool).....	662
Table 459 - SMI Referenced Properties/Methods for CIM_ComputerSystem	662
Table 460 - SMI Referenced Properties/Methods for CIM_ElementCapabilities	663
Table 461 - SMI Referenced Properties/Methods for CIM_ElementSettingData	663
Table 462 - SMI Referenced Properties/Methods for CIM_HostedStoragePool.....	663
Table 463 - SMI Referenced Properties/Methods for CIM_LogicalDisk.....	664
Table 464 - SMI Referenced Properties/Methods for CIM_StorageCapabilities	664
Table 465 - SMI Referenced Properties/Methods for CIM_StoragePool (Concrete)	665
Table 466 - SMI Referenced Properties/Methods for CIM_StoragePool (Primordial).....	665
Table 467 - SMI Referenced Properties/Methods for CIM_StorageSetting	666
Table 468 - SMI Referenced Properties/Methods for CIM_SystemDevice	666
Table 469 - Properties for StorageProtectionCapabilities	668
Table 470 - Properties for StorageProtectionSetting	669
Table 471 - Values for ProtectionControlled	670
Table 472 - Values for Access	670
Table 473 - Values for InquiryProtection.....	670
Table 474 - Values for DenyAsCopyTarget	670
Table 475 - Values for LUNMappingConfigurable	670
Table 476 - Values for ProtectExpirationSpecified.....	671
Table 477 - Values for RemainingProtectionTime	671
Table 478 - Methods of the Storage Element Protection Profile	678
Table 479 - CIM Elements for Storage Element Protection	683
Table 480 - SMI Referenced Properties/Methods for CIM_ElementCapabilities	683
Table 481 - SMI Referenced Properties/Methods for CIM_HostedService.....	684
Table 482 - SMI Referenced Properties/Methods for SNIA_ElementProtectionSettingData	684
Table 483 - SMI Referenced Properties/Methods for SNIA_StorageProtectionCapabilities.....	684
Table 484 - SMI Referenced Properties/Methods for SNIA_StorageProtectionService	685
Table 485 - SMI Referenced Properties/Methods for SNIA_StorageProtectionSetting	685
Table 486 - Supported Profiles for Replication Services.....	686
Table 487 - Key Classes	688

Table 488 - Comparing SyncTypes.....	689
Table 489 - CopyStates Values	702
Table 490 - Indications.....	710
Table 491 - Extrinsic Methods for Group Management	715
Table 492 - Extrinsic Methods for Replication Management.....	715
Table 493 - Extrinsic Methods for Getting Supported Capabilities.....	716
Table 494 - Selected CreateElementReplica optional parameters	720
Table 495 - Selected CreateGroupReplica optional parameters	722
Table 496 - Selected CreateListReplica optional parameters.....	724
Table 497 - SyncTypes	731
Table 498 - Modes	731
Table 499 - Local or Remote.....	731
Table 500 - ReplicationTypes	731
Table 501 - Features.....	732
Table 502 - Group Features.....	734
Table 503 - Consistency	735
Table 504 - Operations	736
Table 505 - Comparison of Similar Operations	737
Table 506 - SettingsDefineState Operations.....	738
Table 507 - Thin Provisioning Features	739
Table 508 - Components.....	739
Table 509 - Default Consistency	740
Table 510 - Group Persistency	740
Table 511 - Copy Methodologies	741
Table 512 - Target Element Suppliers	741
Table 513 - ThinProvisioningPolicy.....	741
Table 514 - Connection Features.....	742
Table 515 - Copy Services and Replication Services Methods Mapping.....	743
Table 516 - CIM Elements for Replication Services.....	744
Table 517 - SMI Referenced Properties/Methods for CIM_ConnectivityCollection.....	747
Table 518 - SMI Referenced Properties/Methods for CIM_ElementCapabilities	748
Table 519 - SMI Referenced Properties/Methods for CIM_GroupSynchronized	748
Table 520 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (ForProtocolEndpoint)	751
Table 521 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (ForRemoteServiceAccessPoint)	751
Table 522 - SMI Referenced Properties/Methods for CIM_HostedCollection (Allocated Resources)	752
Table 523 - SMI Referenced Properties/Methods for CIM_HostedCollection (Between ComputerSystem and ConnectivityCollection)	752
Table 524 - SMI Referenced Properties/Methods for CIM_HostedCollection (Between ComputerSystem and ReplicationGroup).....	752
Table 525 - SMI Referenced Properties/Methods for CIM_HostedCollection (Remote Resources).....	753
Table 526 - SMI Referenced Properties/Methods for CIM_HostedService.....	753
Table 527 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Allocated Resources)	754
Table 528 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (ProtocolEndpoints to ConnectivityCollection).....	754
Table 529 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Remote Resources)	754
Table 530 - SMI Referenced Properties/Methods for CIM_OrderedMemberOfCollection	755
Table 531 - SMI Referenced Properties/Methods for CIM_ProtocolEndpoint.....	755
Table 532 - SMI Referenced Properties/Methods for CIM_RemoteServiceAccessPoint.....	756
Table 533 - SMI Referenced Properties/Methods for CIM_ReplicaPoolForStorage.....	756

Table 534 - SMI Referenced Properties/Methods for CIM_ReplicationEntity	757
Table 535 - SMI Referenced Properties/Methods for CIM_ReplicationGroup	757
Table 536 - SMI Referenced Properties/Methods for CIM_ReplicationSettingData	758
Table 537 - SMI Referenced Properties/Methods for CIM_SAPAvailableForElement.....	759
Table 538 - SMI Referenced Properties/Methods for CIM_ServiceAffectsElement (Between ReplicationService and ConnectivityCollection)	760
Table 539 - SMI Referenced Properties/Methods for CIM_ServiceAffectsElement (Between ReplicationService and ReplicationEntity).....	760
Table 540 - SMI Referenced Properties/Methods for CIM_ServiceAffectsElement (Between ReplicationService and ReplicationGroup).....	761
Table 541 - SMI Referenced Properties/Methods for CIM_SettingsDefineState (Between ReplicationGroup and SynchronizationAspect)	761
Table 542 - SMI Referenced Properties/Methods for CIM_SettingsDefineState (Between storage object and SynchronizationAspect)	761
Table 543 - SMI Referenced Properties/Methods for CIM_SharedSecret	762
Table 544 - SMI Referenced Properties/Methods for CIM_StorageSynchronized.....	762
Table 545 - SMI Referenced Properties/Methods for CIM_SynchronizationAspect	765
Table 546 - SMI Referenced Properties/Methods for SNIA_AllocatedResources	766
Table 547 - SMI Referenced Properties/Methods for SNIA_RemoteResources	767
Table 548 - SMI Referenced Properties/Methods for SNIA_ReplicationService	767
Table 549 - SMI Referenced Properties/Methods for SNIA_ReplicationServiceCapabilities	769
Table 550 - CIM Elements for Thin Provisioning.....	789
Table 551 - SMI Referenced Properties/Methods for CIM_HostedStoragePool	790
Table 552 - SMI Referenced Properties/Methods for SNIA_LogicalDisk	791
Table 553 - SMI Referenced Properties/Methods for SNIA_StorageConfigurationCapabilities (Concrete).....	792
Table 554 - SMI Referenced Properties/Methods for SNIA_StorageConfigurationCapabilities (Global).....	793
Table 555 - SMI Referenced Properties/Methods for SNIA_StorageConfigurationCapabilities (Primordial)	794
Table 556 - SMI Referenced Properties/Methods for SNIA_StorageConfigurationService	795
Table 557 - SMI Referenced Properties/Methods for SNIA_StoragePool (Concrete)	795
Table 558 - SMI Referenced Properties/Methods for SNIA_StoragePool (Empty).....	796
Table 559 - SMI Referenced Properties/Methods for SNIA_StoragePool (Primordial)	797
Table 560 - SMI Referenced Properties/Methods for SNIA_StorageSetting	798
Table 561 - SMI Referenced Properties/Methods for SNIA_StorageVolume	800
Table 562 - CIM Elements for Pools from Volumes	811
Table 563 - SMI Referenced Properties/Methods for CIM_AllocatedFromStoragePool (Volume from Pool).....	812
Table 564 - SMI Referenced Properties/Methods for CIM_ElementCapabilities	812
Table 565 - SMI Referenced Properties/Methods for CIM_SystemDevice	813
Table 566 - SMI Referenced Properties/Methods for SNIA_StorageConfigurationCapabilities.....	813
Table 567 - Supported Profiles for Group Masking and Mapping Profile.....	815
Table 568 - Extrinsic Methods for Masking Group Management	823
Table 569 - Extrinsic Methods for Masking Views Management	823
Table 570 - CIM Elements for Group Masking and Mapping Profile.....	828
Table 571 - SMI Referenced Properties/Methods for CIM_AssociatedDeviceMaskingGroup	831
Table 572 - SMI Referenced Properties/Methods for CIM_AssociatedInitiatorMaskingGroup	831
Table 573 - SMI Referenced Properties/Methods for CIM_AssociatedTargetMaskingGroup.....	831
Table 574 - SMI Referenced Properties/Methods for CIM_AuthorizedPrivilege	832
Table 575 - SMI Referenced Properties/Methods for CIM_AuthorizedSubject.....	832
Table 576 - SMI Referenced Properties/Methods for CIM_AuthorizedTarget	832
Table 577 - SMI Referenced Properties/Methods for CIM_ConcreteDependency (Associates ControllerConfigurationService and ProtocolController)	833
Table 578 - SMI Referenced Properties/Methods for CIM_ConcreteDependency (Associates PrivilegeManagementService and AuthorizedPrivilege)	833

Table 579 - SMI Referenced Properties/Methods for CIM_ConcreteDependency (Associates StorageHardwareIDManagementService and StorageHardwareID)	833
Table 580 - SMI Referenced Properties/Methods for CIM_ConcreteDependency (Associates StorageHardwareIDManagementService and SystemSpecificCollection).....	834
Table 581 - SMI Referenced Properties/Methods for CIM_DeviceMaskingGroup.....	834
Table 582 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (EnabledLogicalElementCapabilities to ControllerConfigurationService).....	835
Table 583 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (EnabledLogicalElementCapabilities to ProtocolController)	835
Table 584 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StorageHardwareID)	835
Table 585 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StorageHardwareIDManagementService)	836
Table 586 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (EnabledLogicalElementCapabilities to SystemSpecificCollection)	836
Table 587 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (System to ProtocolControllerMaskingCapabilities)	836
Table 588 - SMI Referenced Properties/Methods for CIM_ElementSettingData (Associates ComputerSystem and StorageClientSettingData)	837
Table 589 - SMI Referenced Properties/Methods for CIM_ElementSettingData (Associates Port and StorageClientSettingData).....	837
Table 590 - SMI Referenced Properties/Methods for CIM_ElementSettingData (Associates ProtocolController and StorageClientSettingData)	837
Table 591 - SMI Referenced Properties/Methods for CIM_ElementSettingData (Associates StorageHardwareID and StorageClientSettingData)	838
Table 592 - SMI Referenced Properties/Methods for CIM_EnabledLogicalElementCapabilities.....	838
Table 593 - SMI Referenced Properties/Methods for CIM_GroupMaskingMappingCapabilities	839
Table 594 - SMI Referenced Properties/Methods for CIM_GroupMaskingMappingService.....	842
Table 595 - SMI Referenced Properties/Methods for CIM_HostedCollection	843
Table 596 - SMI Referenced Properties/Methods for CIM_HostedService (Associates ComputerSystem and ControllerConfigurationService)	843
Table 597 - SMI Referenced Properties/Methods for CIM_HostedService (Associates ComputerSystem and PrivilegeManagementService)	843
Table 598 - SMI Referenced Properties/Methods for CIM_HostedService (Associates ComputerSystem and StorageHardwareIDManagementService)	844
Table 599 - SMI Referenced Properties/Methods for CIM_InitiatorMaskingGroup.....	844
Table 600 - SMI Referenced Properties/Methods for CIM_MemberOfCollection	844
Table 601 - SMI Referenced Properties/Methods for CIM_PrivilegeManagementService	845
Table 602 - SMI Referenced Properties/Methods for CIM_ProtocolController	845
Table 603 - SMI Referenced Properties/Methods for CIM_ProtocolControllerForUnit	846
Table 604 - SMI Referenced Properties/Methods for CIM_SAPAvailableForElement.....	846
Table 605 - SMI Referenced Properties/Methods for CIM_ServiceAffectsElement (Between GroupMaskingMappingService and MaskingGroup).....	847
Table 606 - SMI Referenced Properties/Methods for CIM_StorageClientSettingData	847
Table 607 - SMI Referenced Properties/Methods for CIM_StorageHardwareID	847
Table 608 - SMI Referenced Properties/Methods for CIM_StorageHardwareIDManagementService	848
Table 609 - SMI Referenced Properties/Methods for CIM_SystemSpecificCollection.....	848
Table 610 - SMI Referenced Properties/Methods for CIM_TargetMaskingGroup	849
Table 611 - SMI Referenced Properties/Methods for SNIA_ProtocolControllerMaskingCapabilities	849
Table 612 - SMI Referenced Properties/Methods for SNIA_StorageHardwareID	850
Table 613 - SMI Referenced Properties/Methods for SNIA_StorageHardwareIDManagementService	850
Table B.1 - Registry of StorageExtent Definitions.....	852
Table B.2 - Example Valid Combinations of Extent Definitions.....	856
Table B.3 - Extent Combinations not defined in this Release of the Standard	856

INFORMATION TECHNOLOGY – STORAGE MANAGEMENT – Part 4: Block devices

FOREWORD

- 1) ISO (the International Organization for Standardization) and IEC (the 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 through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.
- 2) 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 National Committees and ISO member bodies.
- 3) IEC, ISO and ISO/IEC Publications have the form of recommendations for international use and are accepted by IEC National Committees 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.
- 4) In order to promote international uniformity, IEC National Committees 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 or ISO/IEC publication and the corresponding national or regional publication should be clearly indicated in the latter.
- 5) ISO and IEC do not provide any attestation of conformity. Independent certification bodies provide conformity assessment services and, in some areas, access to IEC marks of conformity. ISO or IEC are not responsible for any services carried out by independent certification bodies.
- 6) All users should ensure that they have the latest edition of this publication.
- 7) 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 National Committees 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.
- 8) 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.
- 9) Attention is drawn to the possibility that some of the elements of this ISO/IEC Publication 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 24775-4 was prepared by subcommittee 25: Interconnection of information technology equipment, of ISO/IEC joint technical committee 1: Information technology.

This International Standard, together with ISO/IEC 24775-1 to ISO/IEC 24775-3 and ISO/IEC 24775-5 to ISO/IEC 24775-8, replaces ISO/IEC 24775, second edition, published in 2011, and constitutes a technical revision.

This edition includes the following significant technical changes with respect to the previous edition:

- a) reorganization into eight parts;
- b) maturity identification using stages; and
- c) new profiles.

The list of all currently available parts of the ISO/IEC 24775 series, under the general title *Information technology – Storage management*, can be found on the IEC web site.

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.

IMPORTANT – The “colour inside” logo on the cover page of this publication indicates that it contains colours which are considered to be useful for the correct understanding of its contents. Users should therefore print this publication using a colour printer.

IECNORM.COM : Click to view the full PDF of ISO/IEC 24775-4:2014

Introduction

The Block Devices part of the series of International Standards, *Information technology – Storage management*, contains the profiles for devices that serve block storage. These devices include RAID arrays, Storage Virtualizers, host volume managers, and disk drives. This part also contains supporting profiles, such as the Block Services package.

Parts of this Standard

This International Standard is subdivided into the following parts:

- *Information technology – Storage management – Part 1: Overview*
- *Information technology – Storage management – Part 2: Common architecture*
- *Information technology – Storage management – Part 3: Common profiles*
- *Information technology – Storage management – Part 4: Block devices*
- *Information technology – Storage management – Part 5: Filesystems*
- *Information technology – Storage management – Part 6: Fabric*
- *Information technology – Storage management – Part 7: Host elements*
- *Information technology – Storage management – Part 8: Media libraries*

Changes since the Last Edition

This third edition cancels and replaces the second edition published in 2011 and constitutes a technical revision. This edition includes the following significant changes with respect to the previous edition. The changes fall into three broad categories:

- **Improved organization.** The International Standard has been reorganized into eight parts to provide more information more easily. The parts are:
 - *Part 1 Overview:* The overview book provides a high level overview of the standard.
 - *Part 2 Common Architecture:* This part covers general information about the interface, such as security and protocols.
 - *Part 3 Common Profiles:* This part covers component profiles that extend profiles in other books, such as target ports and job control.
 - *Part 4 Block Devices:* This part covers storage profiles that support various forms of disk storage.
 - *Part 5 Filesystems:* This part covers profiles that support filesystems, such as NAS (Network Attached Storage).
 - *Part 6 Fabric:* This part covers profiles that deal with interconnection of host servers and storage devices, such as switches.
 - *Part 7 Host Elements:* This part covers profiles for storage software on host servers, such as disk partitioning and Host Hardware RAID controllers.
 - *Part 8 Media Libraries:* This part covers profiles that deal with removable media such as tape libraries.
- **Maturity identification.** As material is added to the standard it goes through various stages of maturity. The initial stage is *Experimental*, which is material that has not yet been implemented and is subject to change. The other stages indicate the degree of implementations. The stages are:
 - **Experimental:** Full design review, no commercial implementations.
 - **Implemented:** Initial implementations available, may be removed at minor revision.
 - **Stable:** Three or more vendors have implemented the identified material, backward compatibility assured, removed only at major revision.
 - **Finalized:** Relies solely on Finalized content, deprecated only at major revision.
 - **Deprecated:** Obsolete material, may be removed in future revisions.

For a more detailed explanation of each maturity level and its typographical indication, see Clause 4 Typographical Conventions.

- **Expanded scope.** The range of SAN components modeled by the profiles defined in the parts has been greatly expanded.
- New profiles include:
 - *Part 3 Common Profiles:* Serial Attached SCSI (SAS) Target Port, Serial ATA (SATA) Target Ports, SB Target Port, SAS Initiator Ports, ATA Initiator Ports, FC-SB-x Initiator Ports, FCoE Initiator Ports, Power Supply, Fan, Sensors, Base Server, Media Access Device, Storage Enclosure, Software Inventory, Profile Registration, Proxy Server System Management, Operational Power.
 - *Part 4 Block Devices:* Block Storage Views, CKD Block Services, Erasure, Storage Server Asymmetry, Volume Composition, Storage Element Protection, Replication Services, Pools from Volumes, Group Masking and Mapping, Thin Provisioning.
 - *Part 5 Filesystems:* File Export, File Server Manipulation, File Storage, Filesystem, Filesystem Copy Services, Filesystem Performance, Filesystem Quotas, NAS Network Port, Host Filesystem, Filesystem Remote Copy Services.
 - *Part 6 Fabric:* Fibre Channel Security, Fabric Views, Virtual Fabrics, Switch Partitioning, SAS Expander, N Port Virtualizer, Inter Fabric Routing.
 - *Part 7 Host Elements:* Storage HBA, Host Hardware RAID Controller.
 - *Part 8 Media Libraries:* Partitioned Tape Library, Virtual Tape Library, Virtual Tape Library Copy and Library Views.
- The following experimental profiles were removed from the International Standard:
 - *Part 3 Common Profiles:* Security, 3rd Party Authentication, Authorization, Credential Management, Identity Management, Security Role Based Access Control and Security Resource Ownership.
 - *Part 4 Block Devices:* Pool Management Policy.
- The following profiles were deprecated:
 - *Part 3 Common Profiles:* Cascading (replaced by direct use of cascading classes).
 - *Part 4 Block Devices:* Volume Management (and not replaced).
 - *Part 6 Fabric:* Router (and not replaced).
 - *Part 7 Host Elements:* FC HBA (replaced by Storage HBA), SB Multipath Management (and not replaced).
 - *Part 8 Media Libraries:* InterLibraryPort Connection (and not replaced).
- In addition, many of the existing profiles have been enhanced.

This International Standard was prepared by the SNIA (Storage Networking Industry Association)¹. The standard is often referred to as **SMI-S** (*Storage Management Initiative Specification*).

1. Storage Networking Industry Association, 425 Market Street, Suite 1020, San Francisco, CA 94105, U.S.A., <http://www.snia.org>

INFORMATION TECHNOLOGY – STORAGE MANAGEMENT – Part 4: Block devices

1 Scope

This part of ISO/IEC 24775 defines an interface for the secure, extensible, and interoperable management of a distributed and heterogeneous storage system. This interface uses an object-oriented, XML-based, messaging-based protocol designed to support the specific requirements of managing devices and subsystems in this storage environment. Using this protocol, this International Standard describes the information available to a WBEM Client from an Information technology – Storage management compliant CIM WBEM Server.

2 Normative References

The following documents, in whole or in part, are normatively referenced in the document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document, including any amendments, applies.

ISO/IEC 24775-2, *Information technology – Storage management – Part 2: Common architecture*

ISO/IEC 24775-3, *Information technology – Storage management – Part 3: Common profiles*

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 24775-2 as well as the following apply.

3.1

Constituent Volume

StorageVolumes used to create a concrete StoragePool

3.2

Pool Volume

StorageVolume created from a Constituent Pool

3.3

Constituent Pool

concrete StoragePool created from constituent volumes

4 Typographical Conventions

4.1 Maturity Model

In addition to informative and normative content, this International Standard includes guidance about the maturity of emerging material that has completed a rigorous design review but has limited implementation in commercial products. This material is clearly delineated as described in Clause 4. The typographical convention is intended to provide a sense of the maturity of the affected material, without altering its normative content. By recognizing the relative maturity of different sections of the International Standard, an implementer should be able to make more informed decisions about the adoption and deployment of different portions of the International Standard in a commercial product.

This International Standard has been structured to convey both the formal requirements and assumptions of the Information Technology – Storage Management API and its emerging implementation and deployment lifecycle. Over time, the intent is that all content in the International Standard will represent a mature and stable design, be verified by extensive implementation experience, assure consistent support for backward compatibility, and rely solely on content material that has reached a similar level of maturity. Unless explicitly labeled with one of the subordinate maturity levels defined for this International Standard, content is assumed to satisfy these requirements and is referred to as “Finalized”. Since much of the evolving International Standard content in any given release will not have matured to that level, this International Standard defines three subordinate levels of implementation maturity that identify important aspects of the content’s increasing maturity and stability. Each subordinate maturity level is defined by its level of implementation experience, its stability and its reliance on other emerging standards. Each subordinate maturity level is identified by a unique typographical tagging convention that clearly distinguishes content at one maturity model from content at another level.

4.2 Experimental Maturity Level

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

EXPERIMENTAL
Experimental content appears here.
EXPERIMENTAL

Figure 1 - Experimental Maturity Level Tag

4.3 Implemented Maturity Level

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

International Standard prior to its removal from subsequent releases. Figure 2 is a sample of the typographical convention for Implemented content.

IMPLEMENTED
Implemented content appears here.
IMPLEMENTED

Figure 2 - Implemented Maturity Level Tag

4.4 Stable Maturity Level

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

STABLE
Stable content appears here.
STABLE

Figure 3 - Stable Maturity Level Tag

4.5 Finalized Maturity Level

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

4.6 Deprecated Material

Non-Experimental material can be deprecated in a subsequent revision of the International Standard. Sections identified as "Deprecated" contain material that is obsolete and not recommended for use in new development efforts. Existing and new implementations may still use this material, but shall move to the newer approach as soon as possible. The maturity level

of the material being deprecated determines how long it will continue to appear in the International Standard. Implemented content shall be retained at least until the next revision of the specialization, while Stable and Finalized material shall be retained until the next major revision of the International Standard. Providers shall implement the deprecated elements as long as it appears in the International Standard in order to achieve backward compatibility. Clients may rely on deprecated elements, but are encouraged to use non-deprecated alternatives when possible.

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



Figure 4 - Deprecated Tag

STABLE

5 Array Profile

5.1 Description

The Array Profile describes RAID array systems. The RAID systems supported by this profile are standalone and use local disks to store the data. Systems that use external storage or a combination of local and external storage are "Storage Virtualizers". Systems that plug into backplanes or are on mother boards should use Clause 9 Host Hardware RAID Controller Profile in *Information Technology – Storage Management, Part 7 Host Elements*.

The model consists of multiple subprofiles and packages. The main component profiles are:

- The Array Profile contains a CIM_ComputerSystems object that represents the array as a whole. It is the top level object for the profile.
- Block Services Package is the main part of the model. It contains the StorageExtents that represent the physical storage, StoragePools that gather together the extents and supports allocation and QoS (Quality of Service) settings, and StorageVolumes that represent the logical devices allocated from the pools.
- Target Ports component profile model the ports (e.g., Fibre Channel or iSCSI) through which the LUNs are made available to hosts.

Figure 5: "Array Profile Instance Diagram" is a simplified instance diagram of an array.

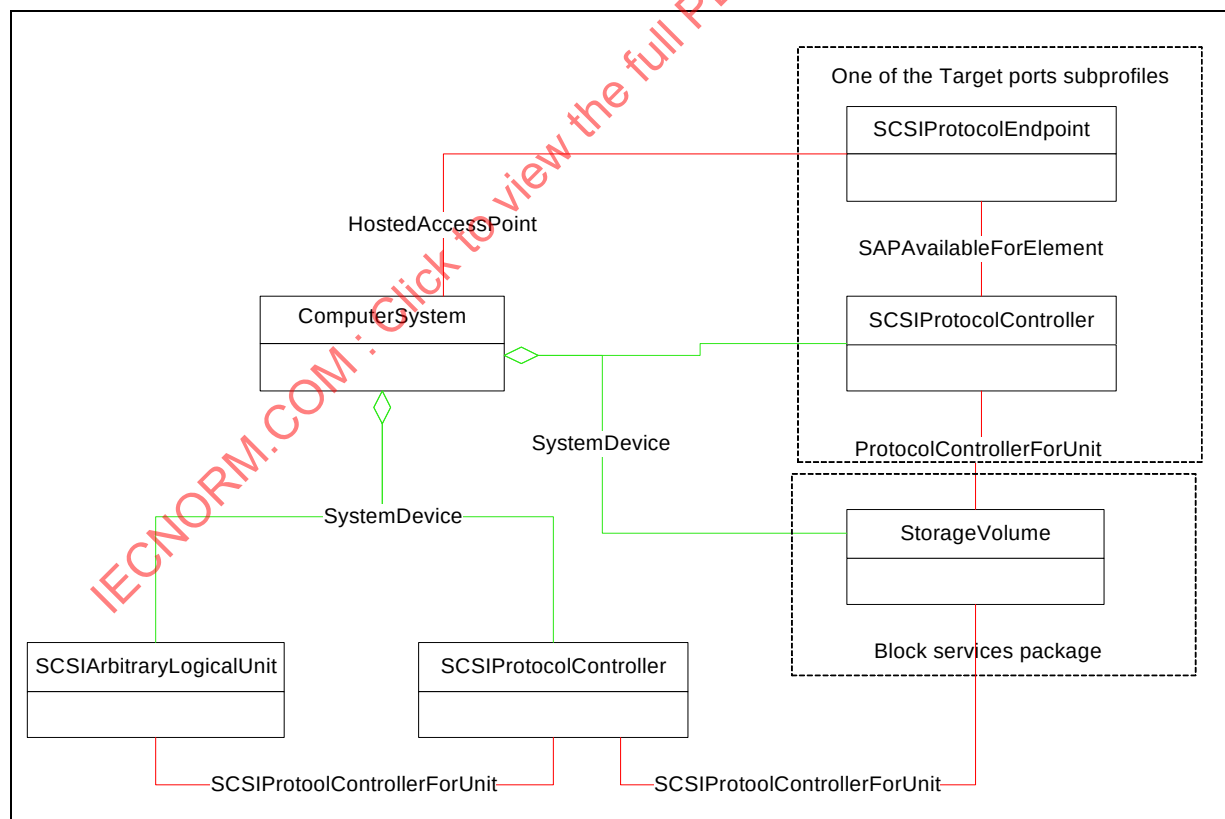


Figure 5 - Array Profile Instance Diagram

At the minimum, the Array Profile provides a high level read-only ‘view’ of an array. Clause 6 Block Services Package includes the basic description of how storage is managed.

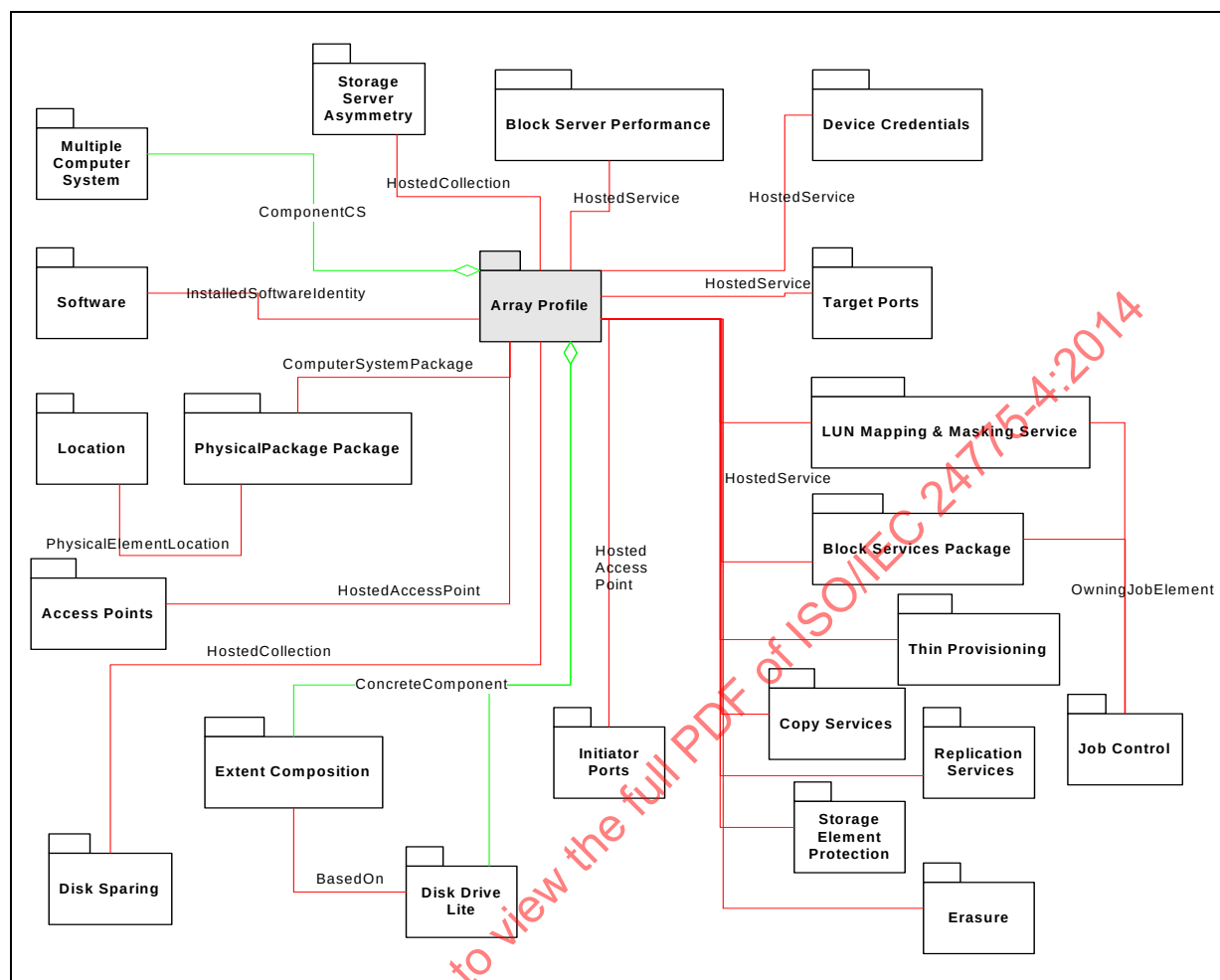


Figure 6 - Array Package Diagram

The various subprofiles indicated in Figure 6: "Array Package Diagram" cover other areas of functionality like location, software/firmware versions, and access to the management interfaces of the array.

The base "Array" Profile only contains the CIM_ComputerSystem object representing the array. This object is attached to the other subprofiles and packages through a set of associations.

The Block Services Package (see 6 Block Services Package) supports configuration of the storage using a QoS (Quality of Service) model. The model is further extended by the “Extent Composition Subprofile” (see 15 Extent Composition Subprofile) to model the details of how the RAID sets are composed. This subprofile supports the detailed configuration of storage by the selection of disk drives and partitions that make-up the RAID sets.

Target Ports model the array ports that provide block data service to the host systems. These ports shall be modeled.

The Generic Initiator Ports Profile (see Clause 15 Generic Initiator Ports Profile) and the Disk Drive Lite Subprofile (see Clause 12 Disk Drive Lite Subprofile) are used to model the physical disk drives and how they are attached to the array system. This part of the model is optional.

Information Technology – Storage Management, Part 3 Common Profiles, 31 Multiple Computer System Subprofile models multiple controllers in a single array system. The model provides a way to model failover and other redundant behavior of a multiple controller system. This subprofile is optional.

The Array Profile includes the “Copy Services” Subprofile to model and configure local and remote snapshots, clones, mirrors, and other array based copying. The copy services will be enhanced in the future to model remote replication. The enhancement is included as experimental in this version of SMI-S. This part of the model is optional.

Information Technology – Storage Management, Part 3 Common Profiles 32 Physical Package Package describes the physical layout of the array and includes product identification information.

5.2 Health and Fault Management

Health and Fault management is described in the referenced subprofiles and packages.

5.3 Cascading Considerations

Not defined in this International Standard.

5.4 Supported Subprofiles and Packages

Table 1 describes the supported profiles for Array.

Table 1 - Supported Profiles for Array

Profile Name	Organization	Version	Requirement	Description
Access Points	SNIA	1.3.0	Optional	
Block Server Performance	SNIA	1.5.0	Optional	
Cluster	SNIA	1.0.2	Optional	Deprecated.
Extra Capacity Set	SNIA	1.0.2	Optional	Deprecated.
Disk Drive	SNIA	1.0.2	Optional	Deprecated.
Disk Drive Lite	SNIA	1.5.0	Optional	
Extent Mapping	SNIA	1.0.2	Optional	Deprecated.
Extent Composition	SNIA	1.5.0	Optional	
Location	SNIA	1.4.0	Optional	
Software	SNIA	1.4.0	Optional	
Copy Services	SNIA	1.5.0	Optional	
Pool Manipulation Capabilities and Settings	SNIA	1.0.2	Optional	Deprecated.
LUN Creation	SNIA	1.0.2	Optional	Deprecated.
Device Credentials	SNIA	1.3.0	Optional	
LUN Mapping and Masking	SNIA	1.0.2	Optional	Deprecated.
Masking and Mapping	SNIA	1.4.0	Optional	
Disk Sparing	SNIA	1.5.0	Optional	
Block Services	SNIA	1.5.0	Mandatory	
CKD Block Services	SNIA	1.5.0	Optional	Experimental.
Indication	SNIA	1.5.0	Mandatory	
Experimental Indication	SNIA	1.5.0	Optional	Experimental.
Physical Package	SNIA	1.5.0	Mandatory	
Health	SNIA	1.2.0	Mandatory	
Multiple Computer System	SNIA	1.2.0	Optional	
Block Storage Views	SNIA	1.5.0	Optional	Experimental.

Table 1 - Supported Profiles for Array

Profile Name	Organization	Version	Requirement	Description
Volume Composition	SNIA	1.5.0	Optional	Experimental.
Job Control	SNIA	1.5.0	Optional	
Storage Element Protection	SNIA	1.4.0	Optional	Experimental.
Storage Server Asymmetry	SNIA	1.4.0	Optional	Experimental.
Erasure	SNIA	1.2.0	Optional	Experimental.
Thin Provisioning	SNIA	1.5.0	Optional	Experimental.
Replication Services	SNIA	1.5.0	Optional	Experimental.
Operational Power	SNIA	1.5.0	Optional	Experimental.
Launch In Context	DMTF	1.0.0	Optional	Experimental. See DSP1102, version 1.0.0
FC Target Ports	SNIA	1.4.0	Support for at least one is mandatory.	
iSCSI Target Ports	SNIA	1.2.0		
SAS Target Ports	SNIA	1.4.0		
SB Target Ports	SNIA	1.2.0		
FC Initiator Ports	SNIA	1.4.0		
SAS Initiator Ports	SNIA	1.4.0		
ATA Initiator Ports	SNIA	1.4.0		
Backend Ports	SNIA	1.0.2		Deprecated.

5.5 Methods of the Profile

None.

5.6 Client Considerations and Recipes

None.

5.7 Registered Name and Version

Array version 1.5.0 (Autonomous Profile).

5.8 CIM Elements

5.8.1 General

Table 2 describes the CIM elements for Array.

Table 2 - CIM Elements for Array

Element Name	Requirement	Description
5.8.2 CIM_ComputerSystem (Top Level System)	Mandatory	'Top level' system that represents the whole array. Associated to RegisteredProfile.
5.8.3 CIM_FilterCollection (Array Predefined FilterCollection)	Conditional	Experimental. Conditional requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='5' (Predefined Filter Collections). This is a collection of predefined IndicationFilters to which a client may subscribe.
5.8.4 CIM_HostedCollection (Array to predefined FilterCollection)	Conditional	Experimental. Conditional requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='5' (Predefined Filter Collections).
5.8.5 CIM_IndicationFilter (Array System Creation)	Conditional	Experimental. Conditional requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='3' (Predefined Filters). This is the 'pre-defined' CIM_IndicationFilter instance for the addition of a new array system instance.
5.8.6 CIM_IndicationFilter (Array System Deletion)	Conditional	Experimental. Conditional requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='3' (Predefined Filters). This is the 'pre-defined' CIM_IndicationFilter instance for the removal of a new array system instance.
5.8.7 CIM_MemberOfCollection (Predefined Filter Collection to Array Filters)	Conditional	Experimental. Conditional requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='5' (Predefined Filter Collections). This associates the Array predefined FilterCollection to the predefined Filters supported by the Array.
5.8.8 CIM_ProtocolControllerForUnit (Arbitrary LU for All LUNs View)	Conditional	Conditional requirement: Elements that are mandatory if Masking and Mapping is not implemented.
5.8.9 CIM_ProtocolControllerForUnit (Storage volumes for All LUNs View)	Conditional	Conditional requirement: Elements that are mandatory if Masking and Mapping is not implemented.
5.8.10 CIM_SCSIArbitraryLogicalUnit (Arbitrary LU)	Optional	A SCSI Logical Unit that exists only for management of the array.
5.8.11 CIM_SCSIProtocolController (All LUNs View)	Conditional	Conditional requirement: Elements that are mandatory if Masking and Mapping is not implemented.

Table 2 - CIM Elements for Array

Element Name	Requirement	Description
5.8.12 CIM_SystemDevice (System to SCSIArbitraryLogicalUnit)	Conditional	Conditional requirement: Elements that are mandatory if SCSIArbitraryLogicalUnit is instantiated. This association links SCSIArbitraryLogicalUnit to the scoping system.
5.8.13 CIM_SystemDevice (System to SCSIProtocolController)	Conditional	Conditional requirement: Elements that are mandatory if Masking and Mapping is not implemented. This association links SCSIProtocolController to the scoping system.
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_ComputerSystem	Mandatory	Addition of a new array instance. See 5.8.5 CIM_IndicationFilter (Array System Creation).
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_ComputerSystem	Mandatory	Deletion of an array instance. See 5.8.6 CIM_IndicationFilter (Array System Deletion).

5.8.2 CIM_ComputerSystem (Top Level System)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

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

Table 3 describes class CIM_ComputerSystem (Top Level System).

Table 3 - SMI Referenced Properties/Methods for CIM_ComputerSystem (Top Level System)

Properties	Flags	Requirement	Description and Notes
CreationClassName		Mandatory	
Name		Mandatory	Unique identifier for the array. E.g. IP address.
ElementName		Mandatory	User friendly name.
OtherIdentifyingInfo	C	Mandatory	
IdentifyingDescriptions	C	Mandatory	
OperationalStatus		Mandatory	Overall status of the array.
NameFormat		Mandatory	Format for Name property.
Dedicated		Mandatory	Indicates that this computer system is dedicated to operation as a storage array.
PrimaryOwnerContact	M	Optional	Contact a details for owner.
PrimaryOwnerName	M	Optional	Owner of the array.

5.8.3 CIM_FilterCollection (Array Predefined FilterCollection)

Experimental. This is a collection of predefined IndicationFilters to which a client may subscribe. An Array implementation shall indicate support for predefined FilterCollections by the SNIA_IndicationConfigurationCapabilities.FeaturesSupported = '5' (Predefined Filter Collections).

Requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='5' (Predefined Filter Collections).

Table 4 describes class CIM_FilterCollection (Array Predefined FilterCollection).

**Table 4 - SMI Referenced Properties/Methods for CIM_FilterCollection
(Array Predefined FilterCollection)**

Properties	Flags	Requirement	Description and Notes
InstanceID		Mandatory	Shall specify the unique identifier for an instance of this class within the Implementation namespace.
CollectionName		Mandatory	The value of CollectionName shall be 'SNIA:Array'.

5.8.4 CIM_HostedCollection (Array to predefined FilterCollection)

Experimental.

Requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='5' (Predefined Filter Collections).

Table 5 describes class CIM_HostedCollection (Array to predefined FilterCollection).

**Table 5 - SMI Referenced Properties/Methods for CIM_HostedCollection
(Array to predefined FilterCollection)**

Properties	Flags	Requirement	Description and Notes
Dependent		Mandatory	Reference to the predefined FilterCollection for the Array.
Antecedent		Mandatory	Reference to the 'Top level' Array System.

5.8.5 CIM_IndicationFilter (Array System Creation)

Experimental. This is the 'pre-defined' CIM_IndicationFilter instance for the addition of a new array system instance. This would represent the addition of a controller computer system to the array. This is a special case of the CIM_IndicationFilter (pre-defined) class as defined in the Indication Profile.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='3' (Predefined Filters).

Table 6 describes class CIM_IndicationFilter (Array System Creation).

Table 6 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Array System Creation)

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
CreationClassName		Mandatory	See the CreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
SystemName		Mandatory	See the SystemName definition in section <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
Name		Mandatory	This shall be 'SNIA:Array:SystemCreation'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in section <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in section <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
Query		Mandatory	SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_ComputerSystem.
QueryLanguage		Mandatory	This should be 'DMTF:CQL' for CQL queries, but may be 'WQL' or 'SMI-S V1.0'. WQL and SMI-S V1.0 are deprecated in favor of 'DMTF:CQL'.
ElementName	N	Optional	See the ElementName definition in section <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).

5.8.6 CIM_IndicationFilter (Array System Deletion)

Experimental. This is the 'pre-defined' CIM_IndicationFilter instance for the removal of a new array system instance. This would represent the removal of a controller computer system from the array. This is a special case of the CIM_IndicationFilter (pre-defined) class as defined in the Indication Profile.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='3' (Predefined Filters).

Table 7 describes class CIM_IndicationFilter (Array System Deletion).

Table 7 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Array System Deletion)

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
CreationClassName		Mandatory	See the CreationClassName definition in section <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
SystemName		Mandatory	See the SystemName definition in section <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
Name		Mandatory	This shall be 'SNIA:Array:SystemDeletion'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in section <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in section <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
Query		Mandatory	SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_ComputerSystem.
QueryLanguage		Mandatory	This should be 'DMTF:CQL' for CQL queries, but may be 'WQL' or 'SMI-S V1.0'. WQL and SMI-S V1.0 are deprecated in favor of 'DMTF:CQL'.
ElementName	N	Optional	See the ElementName definition in section <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).

5.8.7 CIM_MemberOfCollection (Predefined Filter Collection to Array Filters)

Experimental. This associates the Array predefined FilterCollection to the predefined Filters supported by the Array.

Requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='5' (Predefined Filter Collections).

Table 8 describes class CIM_MemberOfCollection (Predefined Filter Collection to Array Filters).

Table 8 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Predefined Filter Collection to Array Filters)

Properties	Flags	Requirement	Description and Notes
Collection		Mandatory	Reference to the Array predefined FilterCollection.
Member		Mandatory	Reference to the predefined IndicationFilters of the Array.

5.8.8 CIM_ProtocolControllerForUnit (Arbitrary LU for All LUNs View)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Elements that are mandatory if Masking and Mapping is not implemented.

Table 9 describes class CIM_ProtocolControllerForUnit (Arbitrary LU for All LUNs View).

Table 9 - SMI Referenced Properties/Methods for CIM_ProtocolControllerForUnit (Arbitrary LU for All LUNs View)

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

5.8.9 CIM_ProtocolControllerForUnit (Storage volumes for All LUNs View)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Elements that are mandatory if Masking and Mapping is not implemented.

Table 10 describes class CIM_ProtocolControllerForUnit (Storage volumes for All LUNs View).

Table 10 - SMI Referenced Properties/Methods for CIM_ProtocolControllerForUnit (Storage volumes for All LUNs View)

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

5.8.10 CIM_SCSIArbitraryLogicalUnit (Arbitrary LU)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 11 describes class CIM_SCSIArbitraryLogicalUnit (Arbitrary LU).

Table 11 - SMI Referenced Properties/Methods for CIM_SCSIArbitraryLogicalUnit (Arbitrary LU)

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	Opaque identifier.
ElementName		Mandatory	User-friendly name.
Name		Mandatory	
OperationalStatus		Mandatory	

5.8.11 CIM_SCSIProtocolController (All LUNs View)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Elements that are mandatory if Masking and Mapping is not implemented.

Table 12 describes class CIM_SCSIProtocolController (All LUNs View).

Table 12 - SMI Referenced Properties/Methods for CIM_SCSIProtocolController (All LUNs View)

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

5.8.12 CIM_SystemDevice (System to SCSIArbitraryLogicalUnit)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Elements that are mandatory if SCSIArbitraryLogicalUnit is instantiated.

Table 13 describes class CIM_SystemDevice (System to SCSIArbitraryLogicalUnit).

Table 13 - SMI Referenced Properties/Methods for CIM_SystemDevice (System to SCSIArbitraryLogicalUnit)

Properties	Flags	Requirement	Description and Notes
PartComponent		Mandatory	
GroupComponent		Mandatory	

5.8.13 CIM_SystemDevice (System to SCSIProtocolController)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Elements that are mandatory if Masking and Mapping is not implemented.

Table 14 describes class CIM_SystemDevice (System to SCSIProtocolController).

**Table 14 - SMI Referenced Properties/Methods for CIM_SystemDevice
(System to SCSIProtocolController)**

Properties	Flags	Requirement	Description and Notes
PartComponent		Mandatory	
GroupComponent		Mandatory	

STABLE

IECNORM.COM : Click to view the full PDF of ISO/IEC 24775-4:2014

STABLE
6 Block Services Package**6.1 Description****6.1.1 General**

Many devices and applications provide their storage capacity to external devices and applications (block consumers) through block-based I/O. This subprofile defines a standard expression of existing storage capacity, the assignment of capacity to StoragePools, and allocation of capacity to be used by external devices or applications.

A block is

- the unit in which data is stored and retrieved on disk and tape devices,
- a unit of application data from a single information category that is transferred within a single sequence.

6.1.2 Storage Capacity States

Figure 7: "Storage Capacity State" illustrates the state of a block of storage.

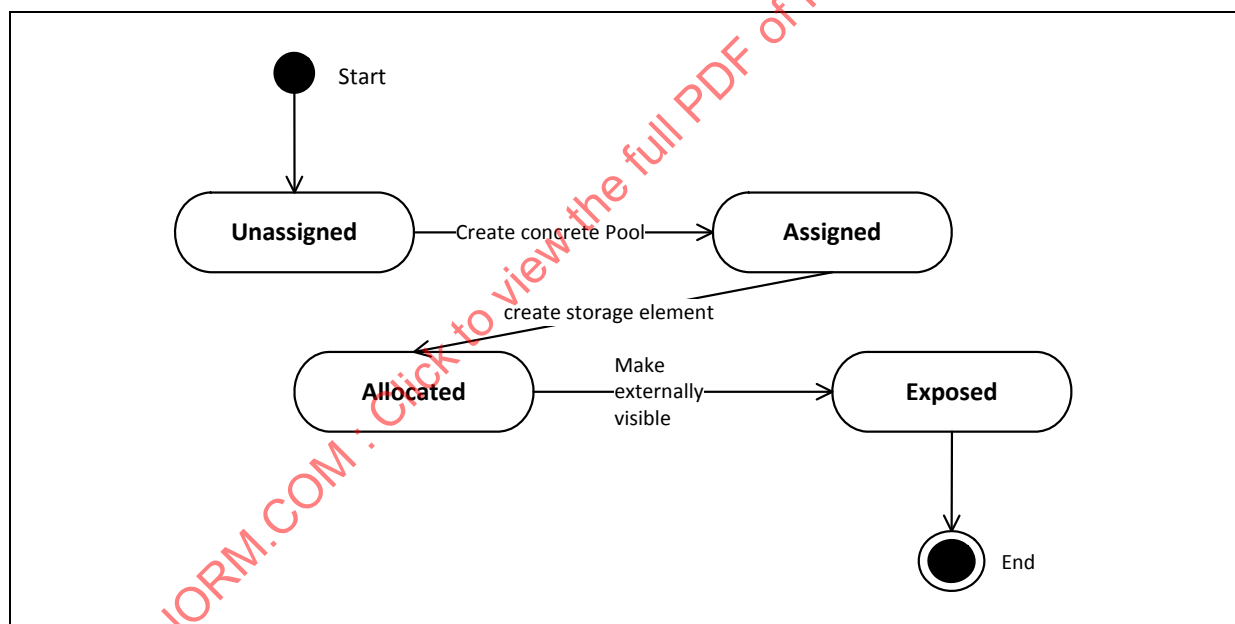


Figure 7 - Storage Capacity State

Each block of capacity within a storage device or application has a state. StorageVolumes and LogicalDisks, the storage elements described in this subclause, are distinct groupings of blocks. An unconfigured storage device or application may not have its capacity organized into concrete StoragePools. All blocks within that unconfigured device or application start in an unassigned state. Once a block is a member of a concrete StoragePool, storage capacity can be assigned. Once a block is a member of a storage element, like a StorageVolume or LogicalDisk, the storage capacity has been allocated for use by a block consumer. Once a block is visible to one or more block consumers, that capacity is exposed.

6.1.3 StoragePools

6.1.3.1 General

A StoragePool is a storage element. Its storage capacity has a given set of capabilities. Those 'StorageCapabilities' indicate the 'Quality of Service' requirements that can be applied to objects created from the StoragePool.

A StoragePool is a mandatory part of modeling disk storage systems that support the Block Services package. However, user manipulation of StoragePools is optional and may not be supported by all disk storage systems. This profile defines the support required to expose functions for creating and modifying StoragePools.

StoragePools are scoped relative to the ComputerSystem (indicated by the HostedStoragePool association). Objects created from a StoragePool have the same Computer System scope.

Child objects (e.g., StorageVolumes, LogicalDisks, or StoragePools) created from a StoragePool are linked back to the parent StoragePool using an AllocatedFromStoragePool association.

There are two properties of StoragePools that describe the size of the 'underlying' storage:

- TotalManagedStorage describes the total storage in the StoragePool;
- RemainingManagedStorage describes the storage currently remaining in the StoragePool.

The Usage property indicates if a storage pool is reserved for use by the array itself, or if the storage pool is reserved for certain operations such as "Reserved for Local Replication Services".

6.1.3.2 Primordial StoragePool

A primordial StoragePool is a type of StoragePool that contains unformatted, unprepared, or unassigned capacity. Storage capacity is drawn from the primordial StoragePool to create concrete StoragePools. A primordial StoragePool aggregates storage capacity not assigned to a concrete StoragePool. StorageVolumes and LogicalDisks are allocated from concrete StoragePools.

At least one primordial StoragePool shall always exist on the block storage system 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 storage of the storage system. The primordial property shall be true for primordial StoragePools.

A primordial StoragePool can be used to determine the amount of capacity on the block storage system that is not assigned to a concrete StoragePool.

6.1.3.3 Concrete StoragePool

A concrete StoragePool is a type of StoragePool. This concrete StoragePool is the only type of StoragePool created or modified by behaviors described in this package. A concrete StoragePool subdivides the storage capacity available in a block server to enable creation or modification of StorageVolumes and LogicalDisks. Concrete StoragePools can be used to assign capacity based on such factors as QoS, cost per megabyte, or ownership of storage. A concrete StoragePool may aggregate the capacity of one or many RAID groups or RAID ranks. A RAID group or rank may be created when the StorageVolume or LogicalDisk is created.

6.1.4 Blocks, Metadata, and Capacity Reported

This subprofile uses the term *metadata* to signify the capacity drawn for the creation of stripes, data copies, and similar items. The capacity removed for such constructs when creating storage elements, like StoragePools, StorageVolumes, and LogicalDisks, is reported in the difference between the capacity of the parent StoragePool and the capacity of the child storage element allocated from that parent. The TotalManagedSpace property represents the capacity that may be used to create or expand child storage elements. The RemainingManagedSpace

property represents capacity left to create a new storage element or expand an existing storage element. One may use this profile to calculate capacity used for metadata.

There is likely to be a difference between a) the capacity calculated by adding up the capacity of all the disks, as reported by the manufacturers, or by adding up the LUNs consumed by a block server, as reported by the block server that exposes them, and b) the capacity that can be used to create other storage organizations or constructs from this capacity, like StoragePools, StorageVolumes, and LogicalDisks. This difference in capacity can be used for disk formatting, for example. The difference in the capacity of the primordial StoragePool and the capacity used to produce the primordial StoragePool is not reported through this subprofile.

6.1.5 StoragePool Management Instance Diagram

Figure 8: "StoragePool Manipulation Instance Diagram" shows an instance diagram for StoragePool manipulation.

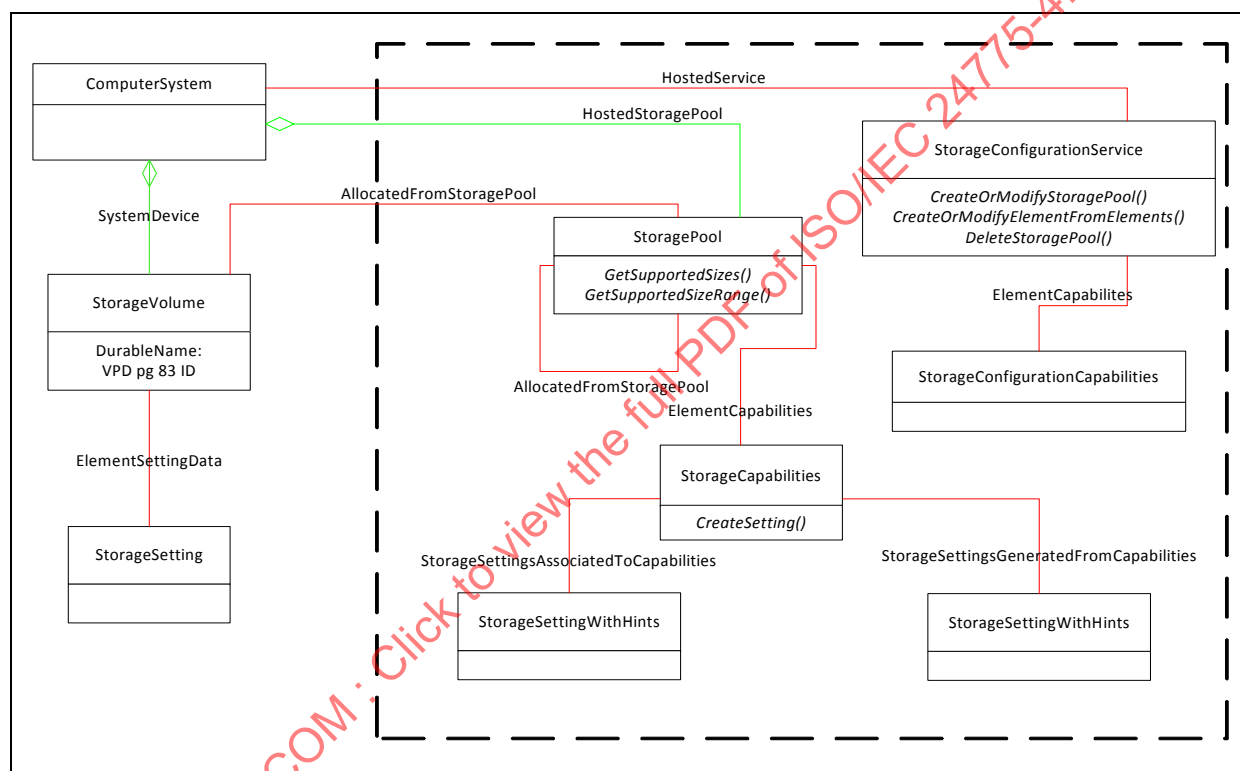


Figure 8 - StoragePool Manipulation Instance Diagram

6.1.6 StoragePool, StorageVolume and LogicalDisk Manipulation

6.1.6.1 General

StorageVolumes are allocations of storage capacity that shall be exposed from a system through an external interface. In the CIM class hierarchy, they are a subclass of a StorageExtent. In SCSI terms, they are logical units.

LogicalDisks are the manifestations of the consumption of storage capacity on a general purpose computer, i.e., a host, as revealed by the operating system or a Volume Manager. In the CIM class hierarchy, they are also a subclass of a StorageExtent. LogicalDisks are a mandatory part of modeling host-based StorageVolume managers.

StorageVolumes and LogicalDisks are consumable storage capacity. These storage elements are the only StorageExtents available to consumers of the block service and a block device.

However, creation or modification of StorageVolumes or LogicalDisks from StoragePools is optional and may not be supported by a given disk storage system. This subprofile defines the

support mandatory if the storage system exposes functions for creating StorageVolumes from StoragePools.

EXPERIMENTAL

The Usage property indicates if a volume or a logical disk is reserved for a special purpose. For example, a volume may be reserved for use by the array itself ("Reserved by the ComputerSystem"), or a volume may have been "set aside" for use by the Migration Services, in which case the usage property of the volume is set to "Reserved by Migration Services".

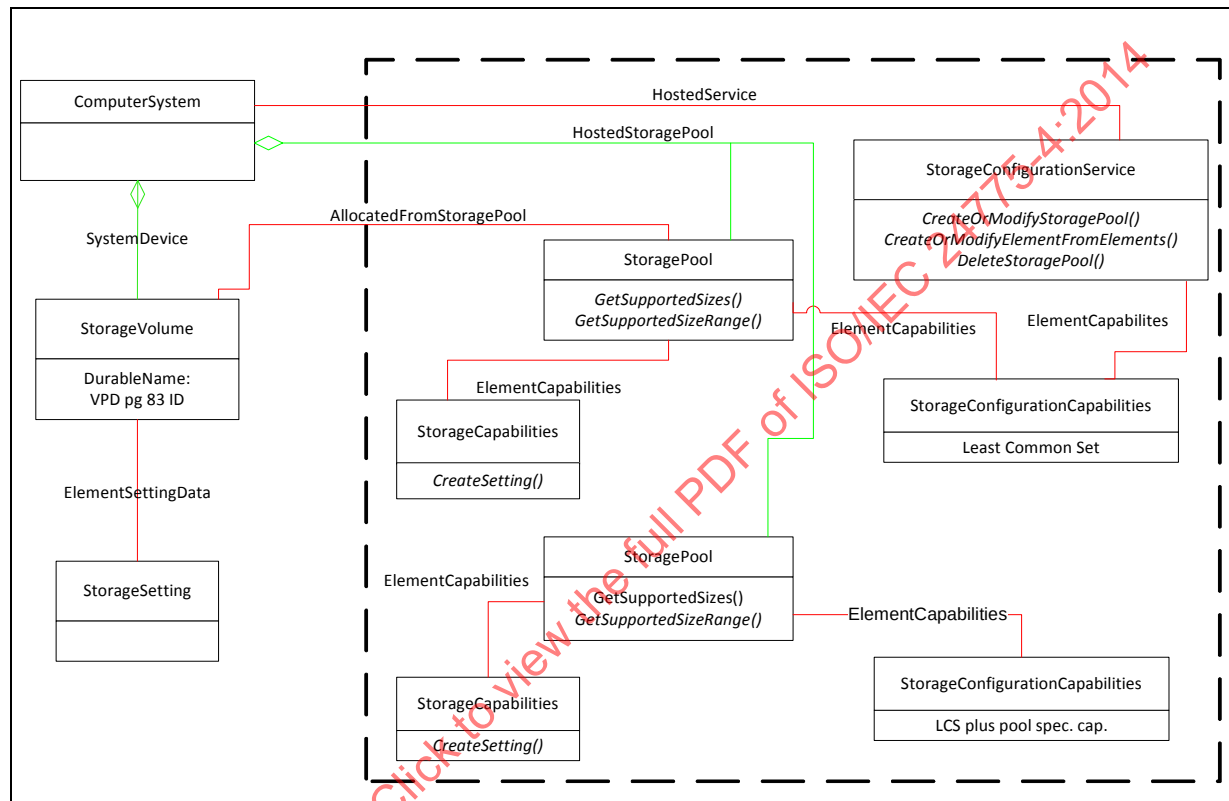


Figure 9 - Capabilities Specific to a StoragePool

Figure 9 illustrates a situation where there are two StoragePools present in an implementation. The top most StoragePool supports the same capabilities as is declared for the entire implementation. The bottom most StoragePool supports the same capabilities as expressed by a different StorageConfigurationCapabilities instance, but with an expanded set of capabilities. For example, the implementation may generally support the creation of StoragePools from StoragePools, but the bottom most StoragePool in the diagram does not.

EXPERIMENTAL

EXPERIMENTAL

Some implementations may impose conditions on when a StorageVolume may be deleted by a user. One example of this is that the storage device may implement a rule that StorageVolumes may only be deleted in the reverse order of creation. Under this rule, all StorageVolumes except the last one created would be marked as not being able to be deleted. Some conditions where a StorageVolume can not be deleted may be related to the Usage property value of the StorageVolume. However this is determined by the implementation.

To enable clients to know which volumes may be deleted, a new property, CanDelete, has been added to SNIA_StorageVolume class. If SNIA_StorageVolume.CanDelete is null or set to true, then the client shall be able to delete the volume, subject to any additional constraints that may be defined in the profiles that would otherwise prevent the volume from being deleted. If SNIA_StorageVolume.CanDelete is set to false, then any client attempt to delete the volume shall be denied (failed) by the implementation, even if there are no constraints on that volume.

In the context of this profile, the value of CanDelete shall be determined by the implementation and shall not be modifiable by the client. The reason is that there are implementation-specific rules that shall be followed and that clients are not allowed to change, even outside the SMI-S.

The value of CanDelete shall be set or cleared dynamically. For example, in the Pools from Volumes case, if a volume that is contributing capacity to a pool is actively in use, it can not be deleted. However, if the same volume is no longer contributing capacity to a pool, then it can be deleted. In other words, the expectation is that the value of CanDelete shall change dynamically.

EXPERIMENTAL

6.1.6.2 StoragePool Manipulation Methods

The StorageConfigurationService, in conjunction with the capacity grouping concept of a StoragePool, allows SMI-S clients to configure StoragePools within block storage systems without specific knowledge about the block storage system configuration. The service has the following StoragePool manipulation methods:

- CreateOrModifyStoragePool: Create a StoragePool with a set of capabilities defined by the input StorageSetting, with possible sources being other StoragePool(s) or StorageExtents. Or modify a StoragePool to increase or decrease its capacity.
- DeleteStoragePool: Delete a StoragePool and return the freed-up storage to the underlying entities.

6.1.6.3 Storage Element Manipulation Methods

The StorageConfigurationService allows SMI-S clients to configure block storage systems with StorageVolumes (ex. LUNs) without specific knowledge about the storage system capacity. The service has the following methods for storage element manipulation:

- CreateOrModifyElementFromStoragePool: Create StorageVolume or LogicalDisk, possibly with a specific StorageSetting, from a source StoragePool. Also modify a StorageVolume or LogicalDisk to increase or decrease its capacity.
- CreateOrModifyElementFromElements: Create a StorageVolume or LogicalDisk using ComponentExtents of a parent and source StoragePool. Also alter the set of member StorageExtents of a StorageVolume or LogicalDisk or change the consumption of an existing set of member StorageExtents.
- ReturnToStoragePool: Return an element previously created with CreateOrModifyElementFromStoragePool to the originating StoragePool.

EXPERIMENTAL

- To locate Pools, Volumes, or Logical Disks based on their current usage, use the method StorageConfigurationService.GetElementsBasedOnUsage.

EXPERIMENTAL

6.1.6.4 Storage Capability Methods

The StorageCapabilities instances provide the ability to create and modify settings for use in StorageVolume creation using the following methods (part of the StorageCapabilities class):

- CreateSetting: Creates a setting consistent with the StorageCapabilities, may be modified before use in creating a StoragePool, StorageVolume, or LogicalDisk.
- GetSupportedStripeLengths and GetSupportedStripeLengthRange: Returns the possible stripe lengths for that capability.
- GetSupportedStripeDepths and GetSupportedStripeDepthRange: Returns the possible stripe depths for that capability.
- GetSupportedParityLayouts: Returns the possible parity layouts, rotated or non-rotated, for that capability.

See 6.5.3 for details on the associations from Setting to Capabilities.

6.1.6.5 Storage Element Size Retrieval

The StoragePool instances provide the ability to retrieve the possible sizes for the StorageVolume or LogicalDisk creation or modification given a StorageSetting as a goal:

- GetSupportedSizes: Returns a list of discrete sizes, given a goal. Also can return the discontinuous capacity in the StoragePool not yet assigned to a concrete StoragePool or allocated to a storage element.
- GetSupportedSizeRange: Returns the range of possible sizes, given a goal.
- GetAvailableExtents: Returns an array of StorageExtent references that matches a given goal and are components of the StoragePool and are not already members of an existing consumable storage element, child StoragePool, StorageVolume, or LogicalDisk.

6.1.7 Declaring Storage Configuration Options

If no StorageConfigurationService is present, then the implementation offers no standard configuration capability (see 6.1.4 "Blocks, Metadata, and Capacity Reported"). If the implementation includes an instance of StorageConfigurationService, it shall also instantiate exactly one StorageConfigurationCapabilities instance associated to the service, referred to as the Global StorageConfigurationCapabilities. The global StorageConfigurationCapabilities shall identify the capabilities of the implementation unless overridden by other provisions. For example, SMI-S does not allow creation of StorageVolumes (or LogicalDisks) from Primordial StoragePools. So even if the StorageConfigurationCapabilities indicates that creation of StorageVolumes are supported, this is overridden by the SMI-S rule that StorageVolumes (or LogicalDisks) shall not be created from Primordial Pools.

The Global StorageConfigurationCapabilities defines the overall capabilities that are supported by the implementation. This instance of StorageConfigurationService shall represent the methods and capabilities of the entire implementation. The Global StorageConfigurationService instance shall state what operation can be done at some time on some set of StoragePools, even if the implementation does not permit some of these operations for some subset of all StoragePools. For example, if create volume is allowed for some StoragePool, then the Global instance shall advise that the create volume operation is supported.

EXPERIMENTAL

Each individual StoragePool may limit these capabilities using another instance of the StorageConfigurationCapabilities associated to that StoragePool via ElementCapabilities. This instance of StorageConfigurationCapabilities represents what configuration operations are permitted for that StoragePool. The StoragePool specific instance of StorageConfigurationCapabilities shall not be associated to the StorageConfigurationService

also. If no StorageConfigurationCapabilities are instantiated for a StoragePool, the client can assume that the Global StorageConfigurationCapabilities apply.

EXPERIMENTAL

Table 15 defines how the SupportedSynchronousActions and SupportedAsynchronousActions array values map to methods in the StorageConfigurationService class. The presence of an 'Action' from Table 15 in the SupportedSynchronousActions array indicates that the associated 'SCS Method' does not produce a Job as a side-effect. Likewise, the presence of an 'Action' from Table 15 in the SupportAsynchronousActions array indicates that the associated 'SCS Method' may produce a Job as a side-effect and a client may use the Job to monitor the progress of the work being done. If an 'Action' may be present in both arrays, the implementation may or may not produce a Job as a side effect.

EXPERIMENTAL

When a StorageConfigurationCapabilities is associated to a StoragePool, the application of the capability is in the context of the StoragePool to which the capabilities are associated. Table 15 also gives the specific meanings of a supported actions in the context of the associated pool ("Pool x").

EXPERIMENTAL

Table 15 - Mapping: Supported Actions to Methods

Action	Associated to "Pool x" Meaning	SCS Method
2 "Storage Pool Creation", 4 "Storage Pool Modification"	"Pool x" may be used as the InPools parameter of CreateOrModifyStoragePool	CreateOrModifyStoragePool
3 "Storage Pool Deletion"	"Pool x" may be used as the Pool parameter of DeleteStoragePool	DeleteStoragePool
5 "Storage Element Creation" 7 "Storage Element Modification"	"Pool x" may be used as the InPool parameter of CreateOrModifyElementFromStoragePool	CreateOrModifyElementFromStoragePool
6 "Storage Element Return"	No meaning specified.	ReturnToStoragePool
12 "Storage Element from Element Creation"	A Storage Element may be created from StorageExtents that are components of "Pool x" (the StorageExtents have a ConcreteComponent or AssociatedComponentExtent association to "Pool x").	CreateOrModifyElementFromElements
13 "Storage Element From Element Modification"	"Pool x" may be used for Storage Element modification using CreateOrModifyElementFromElements. "Pool x" would be TheElement parameter of the method call.	
14 "Element Usage Modification"	No meaning specified.	RequestUsageChange
15 "StoragePool Usage Modification"	"Pool x" may be used as the TheElement parameter of RequestUsageChange	

The SupportedStorageElementTypes array declares what type of storage element may be created or modified by this implementation. For example, support of the StoragePool methods (CreateOrModifyStoragePool and DeleteStoragePool) implies support of creation or modification of storage elements of type StoragePool.

EXPERIMENTAL

When a StorageConfigurationCapabilities are associated to a StoragePool, the valid values of properties differ between Concrete StoragePools and Primordial StoragePools. The valid values and their interpretation are summarized in Table 16.

Table 16 - Valid Values for StorageConfigurationCapabilities associated to a Pool

ConfigurationCapabilities Property	Valid Values for Primordial Pools	Valid Values for Concrete Pools
SupportedStorageElementTypes	none	"2" (StorageVolume) or "4" (LogicalDisk)
SupportedStoragePoolFeatures	"2" (InExtents) or "3" (Single InPool) NOTE This is in reference to creation of pools from the Primordial Pool.	"2" (InExtents), "3" (Single InPool), "5" (Storage Pool QoS Change), "6" (Storage Pool Capacity Expansion) or "7" (Storage Pool Capacity Reduction) NOTE The first two values is in reference to creation of pools from the Concrete Pool. The second three are in reference to the associated pool (e.g., expansion of the pool associated to this capabilities).
SupportedStorageElementFeatures	none	"3" (StorageVolume Creation) or "8" (LogicalDisk Creation)
SupportedSynchronousActions	"2" (Storage Pool Creation), "12" (Storage Element from Element Creation) or "15" (StoragePool Usage Modification)	"2" (Storage Pool Creation), "3" (Storage Pool Deletion), "4" (Storage Pool Modification), "5" (Storage Element Creation), "12" (Storage Element from Element Creation), "13" (Storage Element from Element Modification) or 15" (StoragePool Usage Modification)
SupportedAsynchronousActions	"2" (Storage Pool Creation), "12" (Storage Element from Element Creation) or "15" (StoragePool Usage Modification)	"2" (Storage Pool Creation), "3" (Storage Pool Deletion), "4" (Storage Pool Modification), "5" (Storage Element Creation), "12" (Storage Element from Element Creation), "13" (Storage Element from Element Modification) or 15" (StoragePool Usage Modification)
SupportedStorageElementUsage	none	none
ClientSettableElementUsage	none	none
SupportedStoragePoolUsage	any	any
ClientSettablePoolUsage	any	any

EXPERIMENTAL

EXPERIMENTAL

The arrays SupportedStorageElementUsage and SupportedStoragePoolUsage express what usage values apply to the storage element types. That is, the storage element shall have one of the stated usages.

The arrays ClientSettableElementUsage and ClientSettablePoolUsage express what usage values may be manipulated by SMI-S Clients. That is, only storage elements of the given type may have their usage change changed.

EXPERIMENTAL

The SupportedStoragePoolFeatures array declares what StoragePool behavior is supported, as shown in Table 17.

Table 17 - SupportedStoragePoolFeatures Array

Supported StoragePool Behavior	Explanation
2 "InExtents"	A StoragePool may be created from StorageExtents.
3 "Single InPools", 4 "Multiple InPools"	A StoragePool may be the source of capacity for StoragePool creation or modification, i.e., concrete StoragePools may be created from other StoragePools.
5 "StoragePool QoS Change"	A new setting may be used to modify the quality of service of a StoragePool.
6 "StoragePool Capacity Expansion"	A StoragePool may be expanded
7 "StoragePool Capacity Reduction"	A StoragePool may be shrunk. This operation may be destructive

EXPERIMENTAL

Support for 3 "Single InPools" is fully defined in this International Standard, but 4 "Multiple InPools" is not fully defined and is considered experimental.

EXPERIMENTAL

The SupportedStorageElementFeatures array declares which special features the configuration methods support, shown in Table 18.

Table 18 - SupportedStoragePoolFeatures Array

Supported Special Features	Explanation
3 "StorageVolume Creation", 5 "StorageVolume Modification"	The SMI-S implementation can create or modify StorageVolumes respectively.
8 "LogicalDisk Creation", 9 "LogicalDisk Modification"	The SMI-S implementation can create or modify LogicalDisks respectively.
6 "Single InPool", 7 "Multiple InPools"	If a SMI-S implementation supports the creation or modification of storage elements, then the implementation shall support this creation or modification of concrete StoragePools from either a single StoragePool only or from multiple input StoragePools.
11 "Storage Element QoS Change", 12 "Storage Element Capacity Expansion", 13 "Storage Element Capacity Reduction"	The SMI-S implementation can change the quality of service, grow the capacity of a StorageVolume or LogicalDisk, and shrink the capacity of a StorageVolume or LogicalDisk respectively.
3 "StorageVolume Creation", 5 "StorageVolume Modification"	The SMI-S implementation can create or modify StorageVolumes respectively.

Support for 6 “Single InPools” is fully defined in this International Standard, but 7 “Multiple InPools” is not fully defined and is considered experimental.

EXPERIMENTAL

The SupportedStoragePoolFeatures array indicates which storage elements may be manipulated by SMI-S Clients and thereby which elements can be modified in the ways expressed by these features.

6.1.8 StorageVolume Creation Instance Diagram

Figure 10: “StorageVolume Creation Instance Diagram” shows an instance diagram from StorageVolume creation.

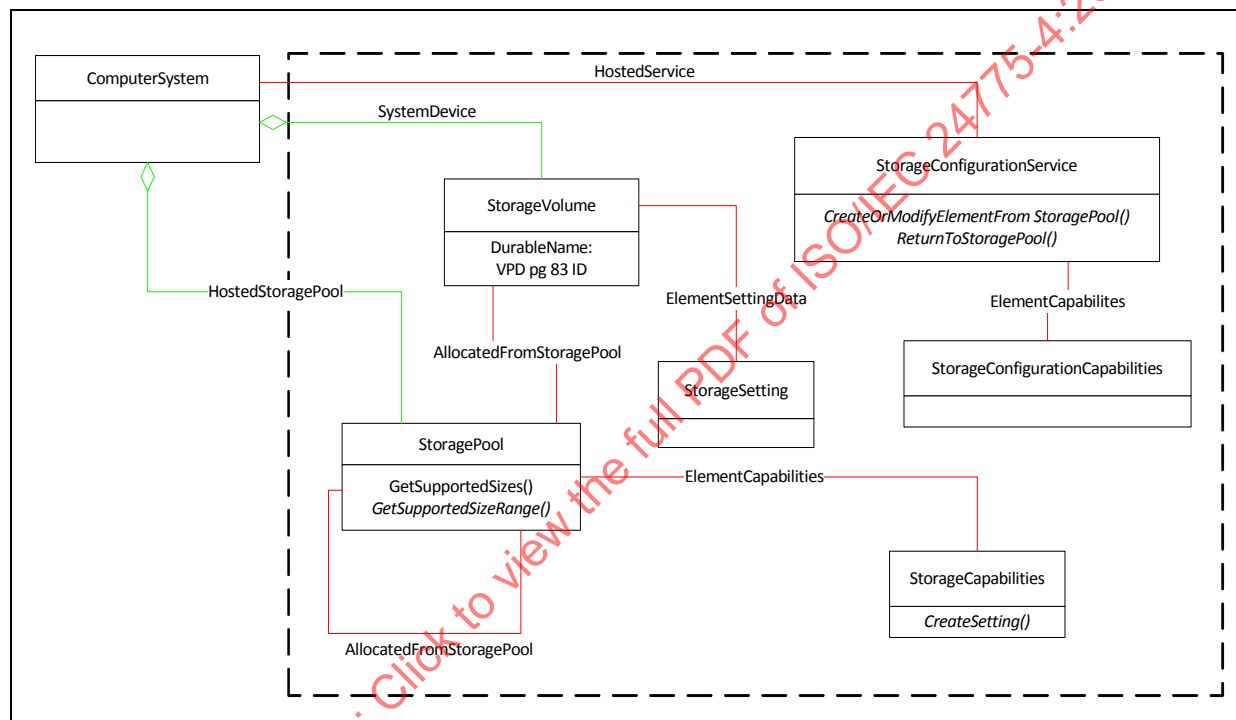


Figure 10 - StorageVolume Creation Instance Diagram

6.1.9 Backward Compatibility

This package is designed to be backward compatible with the “Pool Manipulation Capabilities, and Settings” Subprofile and the “LUN Creation” Subprofile from SMI-S 1.0.x. These subprofiles are deprecated. The Block Services package subsumes all the functionality from these subprofiles. However, to maintain backward compatibility, implementations of this package produce RegisteredProfile instances for these deprecated subprofiles as supporting SMI-S 1.0.3 with one exception. If the BlockServices implementation produces LogicalDisks and not StorageVolumes, then advertising support for these deprecated subprofiles is discouraged. If the implementation supports SLP and the deprecated subprofile RegisteredProfile instances are produced, then these deprecated subprofiles shall be advertised via SLP. See *Information Technology – Storage Management, Part 3 Common Profiles* 41, “Server Profile”.

EXPERIMENTAL

Since the Usage property on StoragePool, StorageVolume, or LogicalDisk did not exist in SMI-S 1.1 and prior versions of SMI-S, the Usage property may be Null. A client may try to utilize a

storage element that is reserved for a restricted usage. In this case, the operation may fail because the supplied volume can not be used for this purpose or as a target for the operation.

EXPERIMENTAL

IMPLEMENTED

SMI-S 1.3. added the ability for an implementation to have the configuration capabilities of a given StoragePool to be more restrictive than what is permitted globally by that implementation. If a given StoragePool cannot support an operation advised as permitted by the global StorageConfigurationCapabilities, then the implementation shall advise clients that attempt the creation or modification of storage elements that there is no capacity for these operations. In order the client that support earlier version of SMI-S are not confused, the result of GetSupportSizes(), GetSupportSizeRange(), and GetAvailableExtents() shall report no available capacity, in the form of no sizes reports or no extents reports, for the StoragePools for which creation or modification operations are not permitted. Previous to SMI-S 1.3., the "In Use" return value was not explained. With SMI-S 1.3., this return code was defined. This code is used to communicate why a storage element may not be deleted.

IMPLEMENTED

6.1.10 Capacity Management

Capacity characteristics of storage systems vary greatly in cost and performance. Storage capacity may need to be partitioned. StoragePools provide a means to aggregate this storage according to characteristics determined by the storage administrator or by the factory when the storage system is assembled.

A StoragePool is an aggregation of storage suitable for configuration and allocation or "provisioning". A StoragePool may be preformatted into a form (such as a RAID group) that makes StorageVolume creation easier.

StoragePools can be drawn from a StoragePool; the result is indicated with the AllocatedFromStoragePool association).

A StoragePool has a set of capabilities held in the StorageCapabilities class. These capabilities reflect the configuration parameters that are possible for elements created from this StoragePool. 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 can delineate the capabilities and define the ranges of these capabilities, as appropriate. Some implementations 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 StoragePools can also change over time.

These storage capabilities are given the scope of the storage system when they are associated to the StorageConfiguratonService or the scope of a single StoragePool when associated to same. The capabilities expressed at the service scope are equal to the union of all primordial StoragePool 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. Once storage elements are drawn from the StoragePool, the administrator can be assured that the elements produced will have the capabilities previous 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 then create StoragePools and/or storage elements, as necessary. Inventories may be made of the capacity by capabilities.

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 characteristics can also be the basis for capacity pooling decisions. A sample configuration is illustrated in Figure 11: "Storage Configuration".

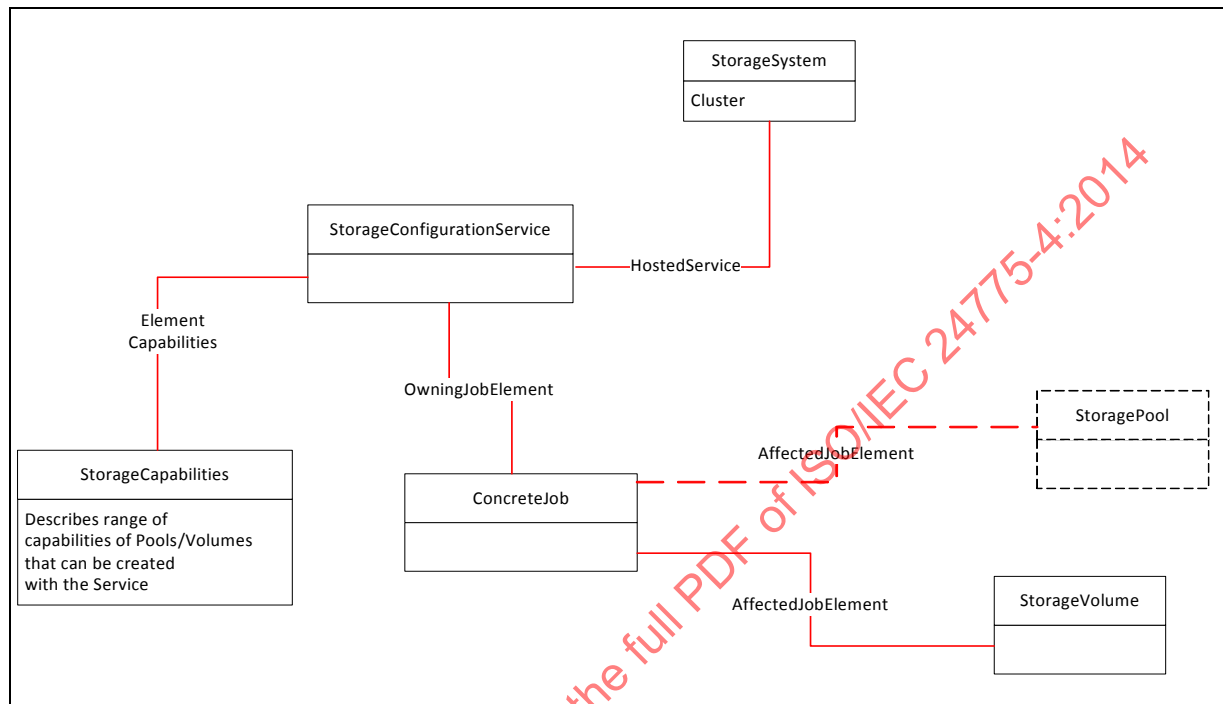


Figure 11 - Storage Configuration

See *Information Technology – Storage Management, Part 3 Common Profiles 27 Job Control Subprofile* for details on the usage of the **StorageConfigurationJob**, **AssociatedStorageConfigurationJob**, and **OwningJobElement** associations.

The definition of storage capabilities intentionally avoids vendor specific details of **StorageVolume** configuration such as RAID types. Although RAID types imply performance and availability levels, these levels cannot be easily compared between vendor implementation—particularly in comparisons with reliability of non-RAID storage (i.e., certain virtualization appliances). 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 **StoragePool** can support storage configured with No Single Points of Failure within the storage system. This parameter does not include the path from the system to the host.
- **Data Redundancy**. Describes the number of complete copies of data maintained. Examples include RAID5 where one copy is maintained and mirroring where two or more copies are maintained.
- **Package Redundancy**. Describes how many physical components (packages), such as disk drives, can fail without data loss (including a spare, but not more than a single global spare). Examples include RAID5 with a Package Redundancy of 1, RAID6 with 2, RAID6 with 2 global (to the system) spares would be 3.

- **ExtentStripeLength.** Describes the number of underlying StorageExtents across which data is striped in the common striping-based storage organizations. Also the number of 'members' or 'columns'. For non-striped organizations (e.g., mirror or JBOD), the ExtentStripeLength shall be 1.
- **UserDataStripeDepth.** Describes the number of bytes forming a stripe in common striping-based storage organizations. The stripe is defined as the size of the portion of a stripe that lies on one StorageExtent. ExtentStripeLength times UserDataStripeDepth yields the size of one stripe of user data.
- **ParityLayout.** Specifies whether a parity-based storage organization is using rotated or non-rotated parity.

Package Redundancy and Data Redundancy values associated to RAID levels are indicated in Table 19.

6.1.11 Mapping of RAID levels to Data Redundancy and Package Redundancy

Table 19 reflects available definitions of RAID levels.

Table 19 - RAID Mapping

RAID Level	Package Redundancy	Data Redundancy	StorageExtent Stripe Length	User Data Stripe Depth	Parity Layout
JBOD	0	1	1	NULL	NULL
0 (Striping)	0	1	2 to N ¹	Vendor Dependent	NULL
1	1	2 - N	1	NULL	NULL
10	1	2 - N	2 to N	Vendor Dependent	NULL
0+1	1	2 - N	2 to N	Vendor Dependent	NULL
3 or 4	1	1	3 to N	Vendor Dependent	1
4DP	2	1	4 to N	Vendor Dependent	1
5 (3/5) ²	1	1	3 to N	Vendor Dependent	2
6, 5DP ³	2	1	3 to N	Vendor Dependent	2
15	2	2 - N	3 to N	Vendor Dependent	2
50	1	1	3 to N	Vendor Dependent	2
51	2	2 - N	3 to N	Vendor Dependent	2

1. The character 'N' represents the variable for the total number of StorageExtents.
2. '3/5' indicate RAID5 implementations that are sometimes called RAID5.
3. 'DP' is double parity.

It is the nature of RAID technology that even though RAID levels are 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 mean for a particular implementation. Instead, RAID level defines storage provided in storage-level terms. If a single storage device implements RAID levels that have the same package redundancy and data redundancy, the implementor should have the SMI-S Client differentiate via StorageSettingsWithHints. Additionally, the SMI-S Provider author can predefine StorageCapabilities that match best practice RAID Levels, including differentiation with StorageSettingWithHints when the StorageVolume or LogicalDisk exists. In this case, the ElementName property is used to correlate between the capability and device documentation. Alternatively, the capability may be expressed in broader ranges for more flexible storage systems.

StorageSetting instances whose "ChangeableType" property is "0", "Fixed - Not Changeable", (identifying the StorageSettings which represent certain non-changeable sets of preset storage property data, describing "fixed", or pre-defined Settings, corresponding to preset RAID levels), the Element name should contain a string value from a comprehensive list of well-known RAID configuration names. The ElementName string value should be the name of the RAID level, from this list, which most closely describes the storage characteristics of the StorageSetting in question. This list of RAID level strings includes, but is not limited to: "JBOD", "RAID0", "RAID1", "RAID0+1", "RAID01E", "RAID10", "RAID3", "RAID4", "RAID4DP", "RAID5", "RAID3/5", "RAID5DP", "RAID6", "RAID15", "RAID50", "RAID51". In addition, the "Description" property of the pre-defined StorageSettings should (optionally) contain similar RAID level information in a more free-form text format, including vendor-specific and/or value-added annotations, for example: "RAID3, with spares", or "RAID5, 7D + 1P".

6.1.12 Storage Setting Associations to Storage Capabilities

A Storage Setting instance can be associated to its parent StorageCapabilities instance through either the StorageSettingsAssociatedToCapabilities or StorageSettingsGeneratedFromCapabilities association instances. The nature of the associated setting is different depending on the association instance used.

A Storage Setting associated via a StorageSettingsAssociatedToCapabilities instance shall not be modifiable by the client (ChangeableType = 0 "Fixed - Not Changeable"). These types of settings are used to define the possible configurations of StoragePools, StorageVolumes or LogicalDisks where the number of possibilities are small because the capabilities of the device itself are likewise limited. When an instance of a Capability class is created as a side effect of creating a concrete StoragePool, this type of Storage Setting may also be created or an existing Storage Setting associated to this new Capabilities instance as well. A client can use the StorageSettingsAssociatedToCapabilities association to find the default goal for the Capabilities instance, using the DefaultSetting property. There shall be one default per combination of a StoragePool instance, associated StorageCapabilities instances, and associated StorageSetting instances.

A Storage Setting associated via a StorageSettingsGeneratedFromCapabilities instance may be modified by a client (ChangeableType = 1 "Changeable - Transient" or Changeable = 2 "Changeable - Persistent"). When a Setting is created from a Capabilities instance, it is transient (e.g., ChangeableType = 1), i.e., the Setting instance may not remain for long. This Setting may be removed from the CIMOM (Common Information Model Object Manager) after reboots or after a set period of time. The client should create and use the Setting as soon as possible. Alternatively, some implementations will allow the client to request that the Setting be retained. This request is made by changing the ChangeableSettingType to 3 "Changeable - Persistent". SMI-S does not define normative behavior for the changing of the ChangeableType property.

EXPERIMENTAL

6.1.13 The Usage Property

The intended usage of storage elements and storage pools is specified in the Usage property of these components. For the most part, the usage of these components is 2 "Unrestricted". However, a system manager and/or a client may decide that certain storage elements are to be set aside for a specific application. For example, a number of volumes are created for the sole purpose of being used for Migration Services. In this case, the volumes are created using a storage setting with the StorageElementInitialUsage of "Reserved by Migration Services". Alternatively, a client may request an "Unrestricted" volume to be converted to "Reserved by Migration Services" by invoking the method StorageConfigurationService.RequestUsageChange. The Provider shall honor the request if the client has access to the storage element and the requested change is valid. The property ClientSettableUsage indicates what usage values are valid for a given component.

The companion property `OtherUsageDescription` may be used to indicate a component's usage that is not covered by the usage value map. The Usage property value in this case shall be set to 1 "Other".

The Usage and `OtherUsageDescription` properties are maintained by the Provider. Restricted values may already exist for static elements that pre-exist when the Provider is discovered.

The Usage and `OtherUsageDescription` property values may change as a side effect of other method calls, e.g. a `StorageVolume` that may have been a replica target candidate at one time, may no longer be a replica target candidate once it is active as a replica target.

Storage elements that support the Usage property will also have a property called `ClientSettableUsage`. This property indicates which usage values may be manipulated by a client using the method `StorageConfigurationService.RequestUsageChange`.

Using the method `StorageConfigurationService.GetElementsBasedOnUsage`, clients are able to retrieve storage elements and storage pools based on their current usage values. For example, a client can retrieve all the volumes that are candidate to be used as a Local Replica Target. Using the same method `StorageConfigurationService.GetElementsBasedOnUsage` with the criteria parameter set to 2 "Available Only", clients are able to retrieve the available (i.e., not in use) storage elements and storage pools based on their current usage value.

Some methods change the usage of a storage element. For example, a client supplies a volume to be used as a target in the call to the method `CreateReplica`.

Table 20 describes some of the representative values of the Usage property (storage element refers to a `StorageVolume`, `LogicalDisk`, or `StoragePool`).

Table 20 - Meaning of Usage values

Usage Value	Description
Reserved by the ComputerSystem	The storage element is used by the array itself for firmware, storage processor software, etc.
Reserved for Local Replication Services	The storage element is designated for activities related to the CopyServices. For example, SNAP cache.
Local Replica Target	The storage element is suitable to be used as replica target.
Element Component	The <code>StorageVolume</code> or <code>LogicalDisk</code> is now acting as a <code>StorageExtent</code> . In this case, the storage element no longer appears in the list of these element types. Use the method <code>GetElementsBasedOnUsage</code> to locate such storage elements.

EXPERIMENTAL

6.1.14 Read-Only Model Requirements

This package defines classes and behavior to express the assignment and allocation of storage capacity and the mechanism for configuring the storage capacity. The expression of the assignment and allocation of storage capacity through the `StoragePool`, `StorageVolume`, `LogicalDisk` and related associations is mandatory. An implementation may also offer the configuration of one or more classes of storage elements. The expression of the support for the configuration of storage is through the instantiation of the `StorageConfigurationService` and its associated `GlobalStorageConfigurationCapabilities`. If an instance of the `StorageConfigurationService` class is not provided, then a client can assume that no configuration operations are supported. An implementation shall not provide an instance of the `StorageConfigurationService` if none of the extrinsic methods of the service are supported.

If the implementation is only supporting read-only information about the capacity assignment and allocation but does not offer modification of the capacity configuration, then that implementation is said to be a *read-only* implementation. In such a model, only classes listed in

Table 21 are required. Classes not explicitly listed are not required for *read-only* implementations.

Table 21 - Classes Required In *Read-Only* Implementation

Required Classes	Reason for Requirement
StoragePool, StorageVolume and/or LogicalDisk, HostedStoragePool and AllocatedFromStoragePool	Reporting of unassigned, assigned, and allocated capacity.
StorageCapabilities and ElementCapabilities	Reporting of storage pool capabilities.
StorageSetting and ElementSettingData used is associated to StorageVolume and LogicalDisk	Reporting of the capabilities of existing StorageVolumes and LogicalDisks.
SystemDevice	Reporting the system to which a StorageVolume or LogicalDisk is scoped.

6.1.15 StorageExtent Conservation

6.1.15.1 General

StorageExtent Conservation is the construct where the remaining capacity after the partial use of a StorageExtent is itself represented as a StorageExtent, based on the antecedent StorageExtent. Note that the StorageExtent class itself does not report the amount of capacity that is used by another StorageExtent that draws capacity from it. In order to calculate the remaining space from a StorageExtent model without StorageExtent Conservation, the client would have to calculate the existence of remaining capacity through finding unused ranges of blocks as expressed by the StorageExtent's BasedOn associations.

This notion allows a client to use those remaining StorageExtents to determine the physical components like disk drives and network ports that are associated to this remaining space in order to pick the StorageExtent best suiting its needs for, for example, storage network redundancy or performance history.

6.1.15.2 Requirements for the General use of StorageExtents

The general use of StorageExtents, which is optional for this subprofile, is subject to the following requirements:

- Allocating capacity from a StoragePool shall not reduce the total size of the StoragePool.
- A given StorageExtent instance shall not be a component of more than one StoragePool. However, an given block may be accounted for in the range of blocks represented by more than one StorageExtent instance. In other words, a given block may be associated to more than one StoragePool.
- The use of all or some of the capacity of an StorageExtent directly, by passing the reference to the StorageExtent in a method call, or indirectly, by passing the size of the desired storage element, shall result in the creation of new StorageExtents that are components of the new StorageVolume or LogicalDisk.
- Any remaining capacity from the StorageExtent shall be represented by a new ComponentExtent of the source StoragePool that is based on the partitioned StorageExtent. This StorageExtent is called a *remaining StorageExtent*.
 - 1) If the Size requested is smaller than the total consumable size of the StorageExtents or StoragePools, then these resources are partially used. In this case, the model shall reflect what capacity was used and what capacity remains of the StorageExtents or StoragePools passed as arguments to CreateOrModifyStoragePool and CreateOrModifyElementFromElements methods.
 - 2) Once the capacity represented by a remaining StorageExtent is used to assign or allocate capacity, the remaining StorageExtent either shrinks in size or is removed from the model. A remaining StorageExtent shall not be molded to have other StorageExtents based on it.

- A StorageExtent that was split or partially used may be made whole by the deletion of the storage element whose creation or modification gave rise to the partial use of the StorageExtent in the first place.

Figure 12: "StorageExtent Conservation - Step 1", Figure 13: "StorageExtent Conservation - Step 2", and Figure 14: "StorageExtent Conservation - Step 3" illustrate the use of StorageExtents to represent the partitioning of a StorageExtent's capacity. An implementation of this subprofile may also implement Clause 15 Extent Composition Subprofile. Extent Conservation requires the instantiation of additional ComponentExtents that represent remaining space. These ComponentExtents are in addition to those modeled by the Extent Composition Subprofile. Available StorageExtents, including remaining space StorageExtents, which meet specific goal requirements, are found using the GetAvailableExtents method of the StoragePool.

The modeling of remaining StorageExtents is not within the scope of the Extent Composition Subprofile. However, the recipes written for Clause 15 Extent Composition Subprofile will tolerate these additional extents. The modeling of free/unused extents is defined only in 6.1.15 StorageExtent Conservation.

Support of the GetAvailableExtents and CreateOrModifyElementFromElements methods are not required by the Block Services package nor Clause 15 Extent Composition Subprofile. An implementation may support the representation of StorageVolume or LogicalDisk structure through Clause 15 Extent Composition Subprofile but not support these methods.

If an implementation supports the GetAvailableExtents and CreateOrModifyElementFromElements methods and the Block Services Package, then it shall also implement Clause 15 Extent Composition Subprofile. See 6.5.3. Additionally, the implementation shall implement either both methods (if it implements one of the methods) or neither method.

The most virtualized Storage Extents are those that have no dependent storage extents that are either StorageVolumes or LogicalDisks. There are three associations that may represent the most virtualized storage components of a StoragePool:

- ConcreteComponent
- AssociatedComponentExtent
- AssociatedRemainingExtent.

IMPLEMENTED

If there are StorageExtents associated to a StoragePool via ConcreteComponent, these StorageExtents shall also be associated to the same StoragePool via AssociatedComponentExtent or AssociatedRemainingExtent. The set of instances associated to this StoragePool via ConcreteComponent shall equal the union of the sets of StorageExtents associated to the same StoragePool via AssociatedComponentExtent and AssociatedRemainingExtent. The subset of AssociatedRemainingExtent StorageExtents represents remaining capacity, as defined in preceding paragraphs. These StorageExtents are remaining StorageExtents. The subset of AssociatedComponentExtent StorageExtents represents capacity that has not yet been allocated, is allocated in part, or is allocated in its entirety.

IMPLEMENTED

6.1.15.3 The Three Steps of StorageExtent Conservation

Figure 12: "StorageExtent Conservation - Step 1", Figure 13: "StorageExtent Conservation - Step 2", and Figure 14: "StorageExtent Conservation - Step 3" show how StorageExtents are partitioned to represent the partial usage of the capacity in the construction of a concrete

StoragePool and a concrete StorageVolume. For the purposes of illustration all the numbers in the figures are expressed in blocks even though some of the class properties are in blocks and others are in bytes. The solid line box around the elements in the diagram groups those classes that are defined in Clause 15 Extent Composition Subprofile.

The initial state in Figure 12: "StorageExtent Conservation - Step 1" starts with a primordial StoragePool that is realized by a primordial StorageExtent. This StorageExtent is part of the initial capacity of the device or added to the device in a process defined outside of this subprofile. The process of assigning capacity to a StoragePool and allocating capacity to a StorageVolume or LogicalDisk is defined inside of this subprofile. To simplify the diagram, the StoragePool has only one ComponentExtent box that represents many StorageExtents. The "SUM_" prefix indicates that the size of the StorageExtents are a summation. Both the StoragePool and StorageExtent start with 1000 blocks of storage capacity.

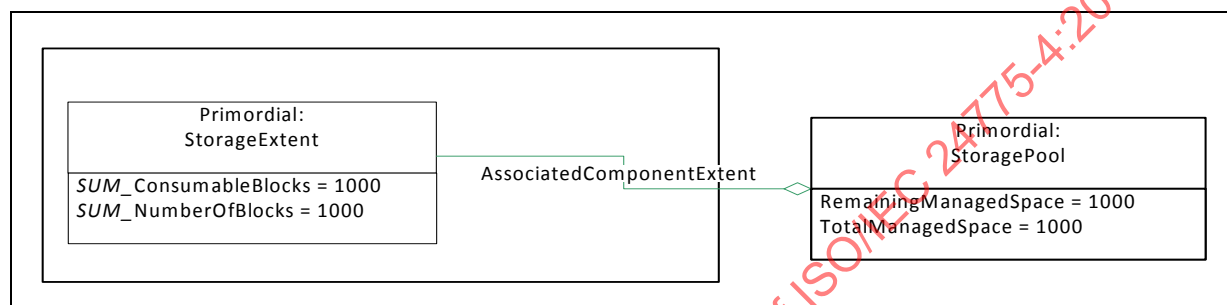


Figure 12 - StorageExtent Conservation - Step 1

A concrete StoragePool is drawn from the primordial StoragePool in step 2, shown in Figure 13: "StorageExtent Conservation - Step 2". Figure 13: "StorageExtent Conservation - Step 2" groups the instances modeled using Clause 15 Extent Composition Subprofile with a dark box. The concrete StoragePool takes only half the capacity of the parent StoragePool. In this particular example, the metadata required by the implementation is written to the storage after this step. Another StorageExtent is created to represent the remaining capacity of the primordial StoragePool that was not used in the creation of the concrete StoragePool. ConsumableBlocks remain constant after the creation of the StorageExtent as a representation of the space actually available for use is other storage elements that are based on the StorageExtent. The remaining space StorageExtent can be used for the creation of other StorageVolumes or Logical Devices. If GetAvailableExtents were called on the primordial StoragePool at this point, a reference to the remaining StorageExtent shall be returned. A reference to the original primordial StorageExtent shall not be returned because the StorageExtent is entirely allocated.

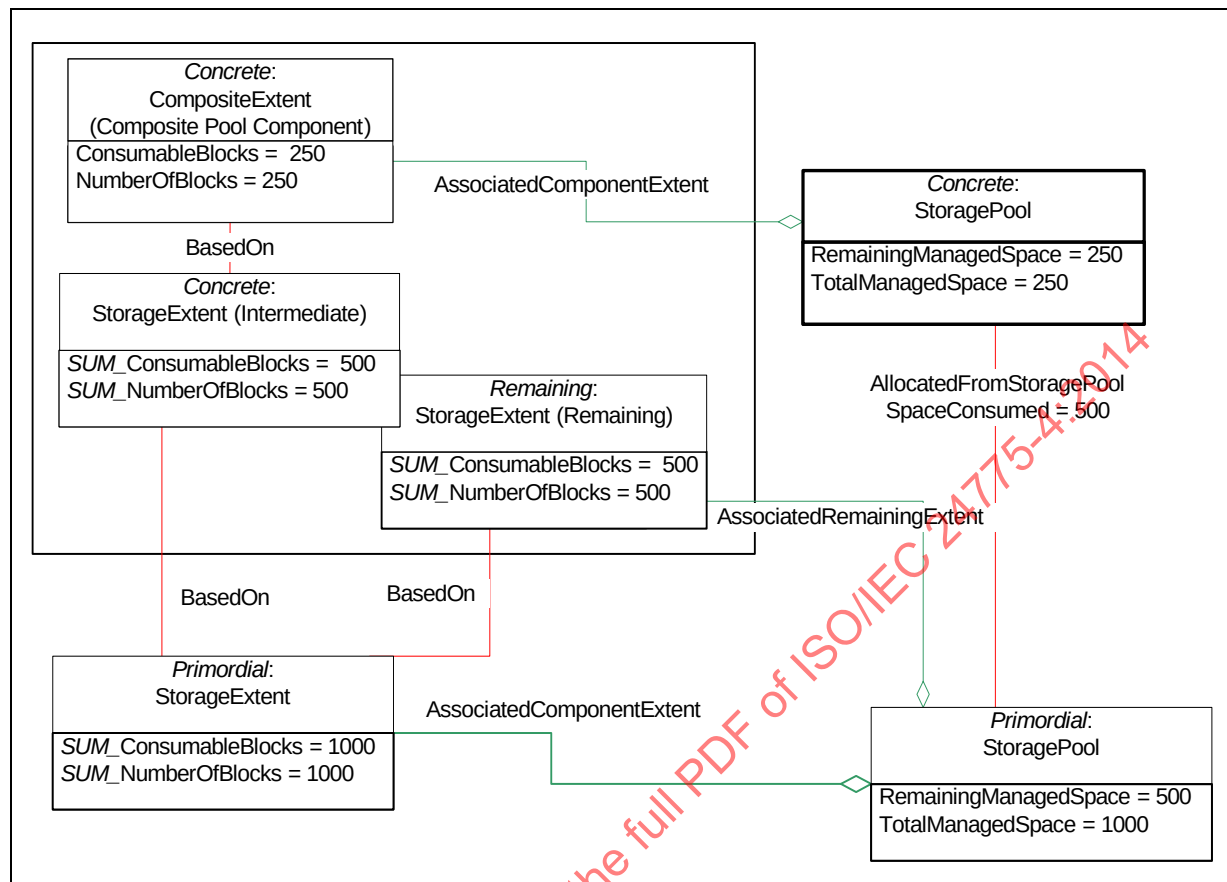


Figure 13 - StorageExtent Conservation - Step 2

Figure 14: "StorageExtent Conservation - Step 3" shows a StorageVolume creation. Figure 14: "StorageExtent Conservation - Step 3" groups the instances modeled using Clause 15 Extent Composition Subprofile with a dark box. This particular implementation draws storage capacity for metadata (for its own house-keeping) during the creation of the StorageVolume. Not shown is the case where the metadata is drawn from capacity during the creation of the concrete StoragePool. A RAID1 stripe is written over three StorageExtents. These StorageExtents are likely to be disk drives. Again, a remaining StorageExtent is created to represent the capacity of the parent concrete StoragePool that is not used in the creation of the StorageVolume. A call to the concrete StoragePool's GetAvailableExtents method yields a reference to the remaining StorageExtent.

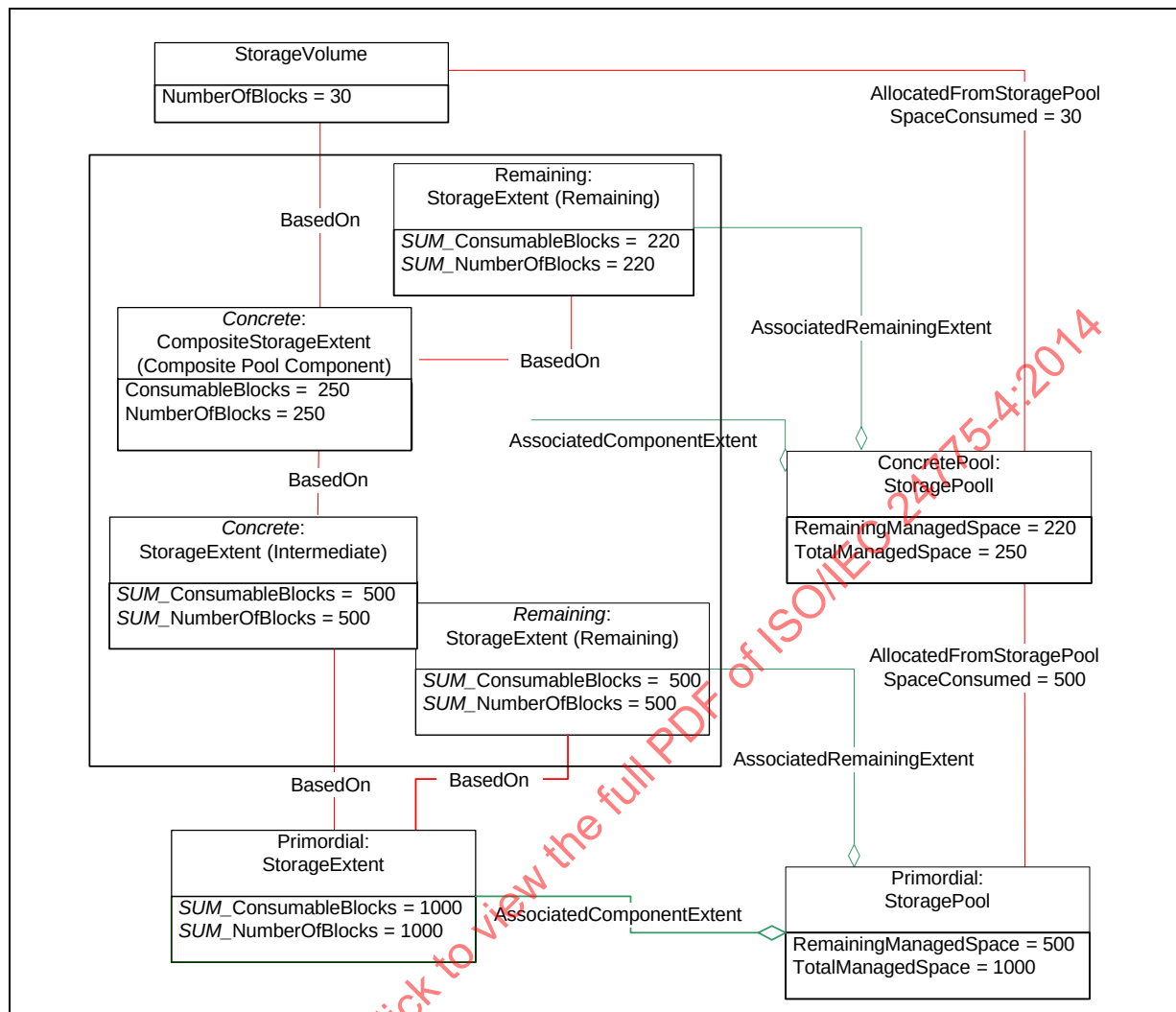


Figure 14 - StorageExtent Conservation - Step 3

In all cases, the TotalManagedSpace and RemainingSpace attributes reflect the total capacity and the capacity that can be drawn from a StoragePool, respectively. In this figure, the metadata is drawn from the capacity in the creation of the storage element.

- The capacity drawn by the metadata from the parent StoragePool is reflected by the sum of associated AllocatedFromStoragePool.SpaceConsumed minus the StoragePool.TotalManagedSpace of the child StoragePool.
- The capacity drawn by the metadata from each StorageVolume or LogicalDisk is reflected by SpaceConsumed minus NumberOfBlocks times BlockSize.

6.1.16 Formulas For Calculating Capacity

These formulas define calculations that shall be valid in a conformant implementation:

- RemainingManagedSpace plus AllocatedFromStoragePool.SpaceConsumed from all of the StorageVolumes, LogicalDisks, and StoragePools allocated from the StoragePool shall equal TotalManagedSpace.
- A parent StoragePool's TotalManagedSpace equals RemainingManagedSpace plus the sum of all related AllocatedFromStoragePool SpaceConsumed.
- If Clause 15 Extent Composition Subprofile is implemented:

IMPLEMENTED

- 1) The StoragePool's TotalManagedSpaces shall be equal to the sum of all the Associated-ComponentExtent StorageExtent's BlockSize times ConsumableBlocks.

IMPLEMENTED

- 2) Using the BasedOn association from the StoragePool's component StorageExtents (found using ConcreteComponent or AssociatedComponentExtent, or AssociatedRemainingExtent), the sum of the Dependent StorageExtent's NumberOfBlocks shall be equal to the ConsumableBlocks of the Antecedent StorageExtent.

IMPLEMENTED

- 3) The StoragePool's RemainingManagedSpace shall be equal to the sum of BlockSize times ConsumableBlocks for the union of the following sets of StorageExtents:
 - The set of StorageExtents associated to the StoragePool via AssociatedComponentExtent where each StorageExtent does not participate in an Antecedent relationship via one or more BasedOn associated with either a StorageVolume or a LogicalDisk.
 - The set of StorageExtents associated to the StoragePool via AssociatedRemainingExtent.

IMPLEMENTED**6.1.17 Storage Element Manipulation**

The StorageConfigurationService class contains methods to allow creation, modification and deletion of StorageVolumes or LogicalDisks. The capabilities of a StorageConfigurationService or StoragePool to provide storage are indicated using the StorageCapabilities class. This class allows the Service or StoragePool to advertise its capabilities (using implementation independent attributes) and a client to set the attributes it desires.

The concept of "hints" is included. Hints allow a client to provide general requirements to the system as to how it expects to use the storage. Hints allow a client to provide extra information to "tune" a StorageVolume or LogicalDisk. If a client chooses to supply these hints when creating a StorageVolume or LogicalDisk, the StorageSystem can either use the hints to determine a matching configuration or ignore them.

When creating a StorageVolume or LogicalDisk, a reference to an instance of StorageSetting is passed as a parameter to the StorageConfigurationService.CreateOrModifyElementFromStoragePool or CreateOrModifyElementFromElements methods. This reference provides a goal for that element.

The current 'service level' being achieved is reported via the StorageVolume or LogicalDisk class itself. For example, data redundancy reported in the Setting associated to the storage element may be different from the data redundancy reported in the storage element itself. This difference indicates that a copy of the data is no longer available.

StorageVolumes or LogicalDisks are created from StoragePools via a StorageConfigurationService's CreateOrModifyElementFromStoragePool() method. A StorageVolume creation operation takes time, and a Client needs to be aware that the operation is not complete until the StorageVolume.OperationalStatus is OK. A Client may also monitor the progress of the operation using the ConcreteJob class and its properties.

The name of a StorageVolume, LogicalDisk, or StoragePool can be changed. The existence of the EnabledLogicalElementCapabilities instance associated to the storage element indicates

that the storage element can be named. If `ElementNameEditSupported` is set to `TRUE`, then the `ElementName` of the associated storage element name can be modified. The `ElementNameMask` property provides the regular expression that indicates the name limits. See Table 24, "CIM Elements for Block Services" for details for this property.

This model does not help in communicating whether or not the element name can be provided in the creation or the modification of the storage element through these `StorageConfigurationService` methods (if there are no existing storage elements of a given type):

- `CreateOrModifyStoragePool()`
- `CreateOrModifyElementFromStoragePool()`
- `CreateOrModifyElementFromElements()`

First, there shall be a single `EnabledLogicalElementCapabilities` for each storage element type.

Note that the `ElementType` parameter of these methods requests the element to be created or modified. There shall be a single mask for each storage element type. Each of these instances shall be associated to the `StorageConfigurationService` via the `ElementCapabilities` association. Each of these `EnabledLogicalElementCapabilities` instances may also be used to express the capabilities of storage elements. The `ElementNames` of these `EnabledLogicalElementCapabilities` instances that define the possibility of naming `StoragePools`, `StorageVolumes`, and `LogicalDisks` type shall be of the values of "StoragePool Enabled Capabilities", "StorageVolume Enabled Capabilities", and "LogicalDisk Enabled Capabilities" respectively. If the implementation supports the creation or modification of a given element type and the modification of the name of the storage element, then it shall produce the aforementioned `EnabledLogicalElementCapabilities` instances.

If a storage element's name is modifiable through one of the aforementioned `StorageConfigurationService` methods, it shall also be modifiable through instance modification. However, a storage element's name may be modifiable through instance modification, but may not be modifiable through these service methods.

EXPERIMENTAL

By default, storage elements are created with the 2 "Unrestricted" value for their `Usage` property. To specify a different value for the `Usage` property, set the appropriate `StorageExtentInitialUsage` or `StoragePoolInitialUsage` of the applicable `StorageSetting` before creating the storage element. Subsequently, the `Usage` property can be modified by calling the `StorageConfigurationService.RequestUsageChange` method.

EXPERIMENTAL

EXPERIMENTAL

6.1.18 Block Services Predefined Indications

If the optional Experimental Indication profile is supported by an implementation, there shall be an implementation of the `SNIA_IndicationConfigurationService` and its associated `SNIA_IndicationConfigurationCapabilities` associated to the `ComputerSystem` of the referencing profile associated with the Block Services package. If the implementation supports predefined `IndicationFilters` or predefined `IndicationFilterCollections` this shall be indicated in the `SupportedFeatures` property of the `SNIA_IndicationConfigurationCapabilities`. If a value "3" is present, it means the implementation supports predefined `IndicationFilters`. If a value of "5" is present, it means the implementation supports predefined `IndicationFilterCollections`.

Figure 15 illustrates classes that shall be populated by the Block Services Package if both “3” and “5” are present in the SupportedFeatures property.

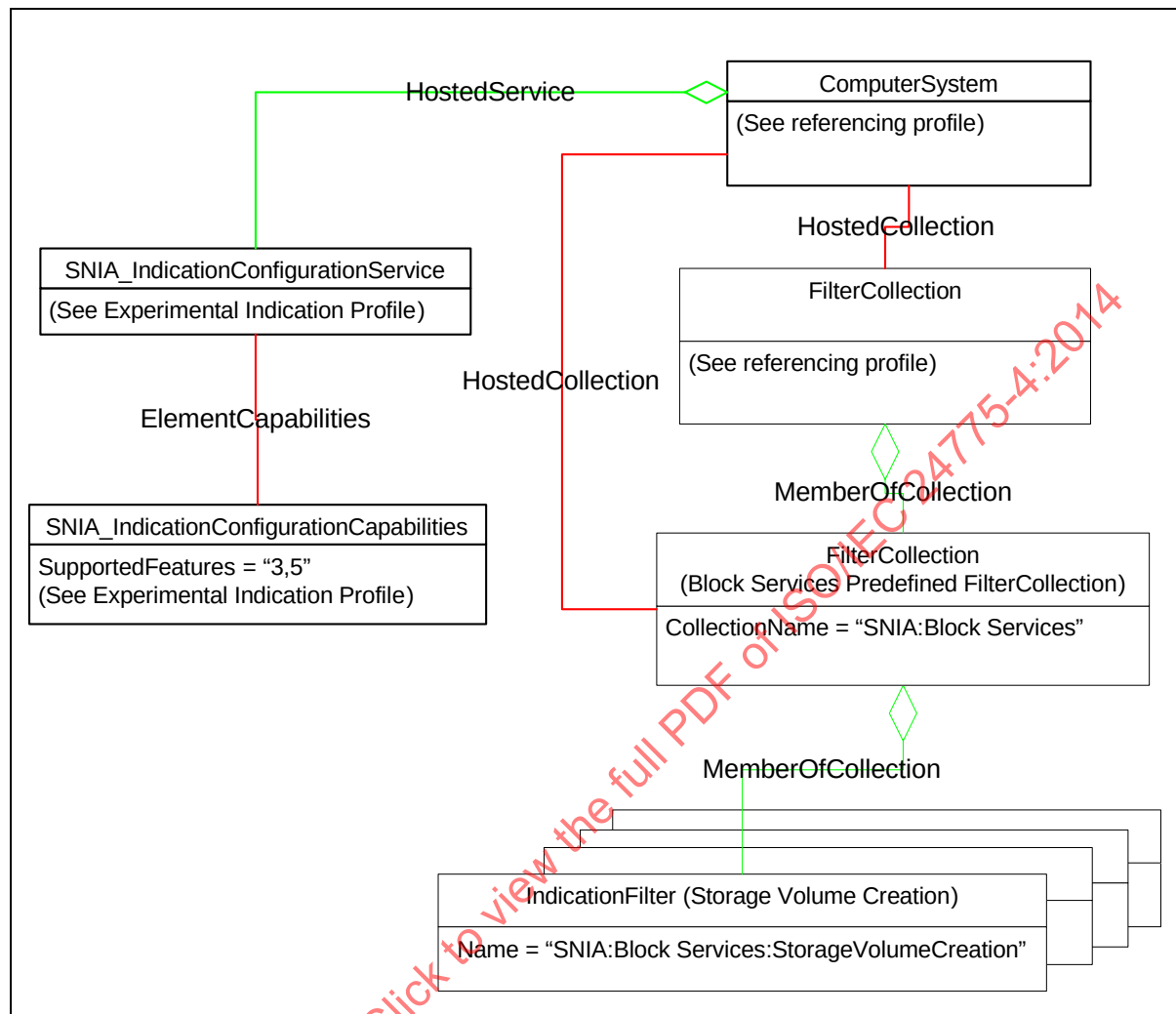


Figure 15 - Block Services Predefined FilterCollection

The SNIA_IndicationConfigurationService is hosted on the ComputerSystem for the referencing profile associated to the Block Services component profile. The FilterCollection for block services is also hosted on the same ComputerSystem. The block services FilterCollection is a member (MemberOfCollection) in the FilterCollection of the referencing profile. The block Services FilterCollection has members which are all the predefined IndicationFilters supported by the implementation. This shall include all Mandatory IndicationFilters of the Block Services Package. But it should also include any IndicationFilter that has been predefined by the implementation. This may include conditional, optional or vendor specific IndicationFilters supported by the implementation.

The block services FilterCollection shall have the CollectionName “SNIA:Block Services”. Each of the predefined filters shall have the Name property as defined for the IndicationFilter. In Figure 15 the name of the IndicationFilter (Storage Volume Creation) is “SNIA:Block Services:StorageVolumeCreation”. For vendor specific IndicationFilters (not defined in this International Standard), the Name of the filter would be of the form ORG_ID”:Block Services:”UNIQUE_ID, where ORGID is the designation of the vendor that is providing the implementation.

EXPERIMENTAL

6.2 Health and Fault Management Considerations

The extrinsic methods should produce Errors (instances of CIM_Error) instead of some of the failure return codes. CIM errors include parameter errors, hardware efforts, and time-out errors. See *Information Technology – Storage Management, Part 3 Common Profiles*, Clause 26 Health Package for details.

EXPERIMENTAL

The standard messages specific to this profile are listed in Table 22. See *Information Technology – Storage Management, Part 2 Common Architecture*, Clause 9 Standard Messages for a description of standard messages and the list of all standard messages.

Table 22 - Standard Messages for Block Services Package

Message ID	Message Name
MP17	Invalid property combination during instance creation or modification.
DRM19	Stolen capacity.
DRM20	Invalid extent passed.
DRM21	Invalid deletion attempted.

EXPERIMENTAL

6.3 Cascading Considerations

Not defined in this International Standard.

6.4 Supported Profile, Subprofiles and Packages

Table 23 describes the supported profiles for Block Services.

Table 23 - Supported Profiles for Block Services

Profile Name	Organization	Version	Requirement	Description
Job Control	SMA	1.5.0	Optional	
Extent Composition	SNIA	1.5.0	Optional	

6.5 Methods of this Profile

6.5.1 Extrinsic Methods on StorageCapabilities

6.5.1.1 CreateSetting

CreateSetting is a method in StorageCapabilities and is invoked in the context of a specific StorageCapabilities instance.

```
uint32 CreateSetting(
    [In] uint16 SettingType,
    [Out] CIM_StorageSetting REF NewSetting)
```

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 define what types of storage can be allocated. The client shall determine what subset of the parent StoragePool capabilities to use, albeit a primordial StoragePool or a concrete StoragePool. The StorageSetting provided to the StoragePool creation method defines what 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' ("Default") 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, with using '3' ("Goal"), the new StorageSetting attributes are set to the related attributes of the parent StorageCapabilities, e.g., Min to Min and Max to Max. The method CreateSetting should return a unique instance of StorageSetting so that the ModifyInstance operation by one client shall not impact another client's instance of 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 implementation shall not generate a Setting whose values fall outside of the range of the parent Capabilities.

The StorageSetting cannot be used to create storage that is more capable than the parent StorageCapabilities. The ModifyInstance and SetProperties CIM Operations shall fail 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 or LogicalDisk shall not be modified or deleted directly.

Once this type of StorageSetting is used as the Goal for the creation or modification of a StoragePool, the Goal setting properties are copied into a new StorageCapabilities instance. The new StorageCapabilities instance is associated to the newly created or modified StoragePool. If the StoragePool was modified, then the previous StorageCapabilities shall be removed. The new StorageCapabilities instance, associated with the new StoragePool, should describe the parameters used in its creation or modification.

Once this type of StorageSetting is used as the Goal for the creation or modification of a StorageVolume or LogicalDisk, the Goal StorageSetting shall be duplicated, with the exception of the instance keys. The duplicate Setting is associated to the newly created or modified StoragePool, StorageVolume, or LogicalDisk. The generated Setting may be removed thereafter. The new StorageSetting instance, associated with the new storage element, should describe the parameters used in its creation or modification.

The following set of methods (6.5.1.2, 6.5.1.3, and 6.5.1.4) can be implemented to allow a client to be more specific about the configuration of the stripe length, stripe depth, and parity in a Setting. Thereby the client can get specific RAID levels or quality of service characteristics.

The stripe length, stripe depth, and party extrinsic methods may be supported. These methods may be supported in the content of one capabilities and not in another within the same implementation. Sometimes the block striping is done as part of the creation of the concrete StoragePool, and sometimes the block striping is done as part of the creation of a StorageVolume or LogicalDisk. There may be implementations that allow striping to be done in both steps.

A client may use `StorageSettingHints` to imply desired striping (or other) characteristics are desired. The striping and parity methods and properties may be used in combination with hints. The hints express a ranking of preference. While the striping and parity methods and properties are much more explicit. When the hints and the stripe and parity Settings properties are used in combination, the striping and parity properties of the Setting are also considered hints, and the implementation may still create or modify the `StoragePool` or storage element using its best effort.

This International Standard does not define how the ranking of hints relates to the exact nature of the `StoragePool` or storage element created or the nature of their modification.

6.5.1.2 Getting Stripe Length

```
uint32 GetSupportedStripeLengths(
    [Out] uint16 StripeLengths[])
```

This method is used to report discrete `ExtentStripeLengths` for `StorageVolume`, `LogicalDisk`, or `StoragePool` creation. Some systems may support only discrete stripe lengths.

```
uint32 GetSupportedStripeLengthRange(
    [Out] uint16 MinimumStripeLength,
    [Out] uint16 MaximumStripeLength,
    [Out] uint32 StripeLengthDivisor)
```

This method is used to report a range of possible `ExtentStripeLengths` for `StorageVolume`, `LogicalDisk`, or `StoragePool` creation. Some systems may support only a range of sizes. This method reports the continuum of discrete sizes between the minimum and maximum as defined by intervals of the divisor (e.g., if given a min of 10 and a max of 50, the discrete values would be 20, 30, 40, and 50).

Either method may be supported. Return codes are:

- 0, "Method completed OK", means success.
- 1, "Method not supported",
- 2, "Choices not available for this Capability." Although the method may be supported by Capabilities in this implementation, it is not supported for this Capability. Usually, this return code indicates that the stripe length has already been set in the parent `StoragePool` and may not be changed.
- 3, "Use [GetSupportedStripeLengths|GetSupportStripeLengthRange] instead". This return code tells the client that this stripe method is not supported, but the other stripe method is supported.

6.5.1.3 Getting Stripe Depth

```
uint32 GetSupportedStripeDepths(
    [Out] uint64 StripeDepths)
```

This method is used to report discrete `UserDataStripeDepths` for `StorageVolume`, `LogicalDisk`, and `StoragePool` creation. Some systems may support only discrete depth byte sizes.

```
uint32 GetSupportStripeDepthRange(
    [Out] uint64 MinimumStripeDepth,
    [Out] uint64 MaximumStripeDepth,
    [Out] uint64 StripeDepthDivisor)
```

This method is used to report a range of possible `UserDataStripeDepths` for `StorageVolume`, `LogicalDisk`, or `StoragePool` creation. Some systems may support only a range of sizes. The method reports the continuum of discrete sizes between the minimum and maximum as defined by intervals of the devisor (e.g., if given a min of 10 and a max of 50, the discrete values would be 20, 30, 40, and 50).

Either method may be supported. Return codes are:

- 0, "Method completed OK", means success.

- 1, "Method not supported"
- 2, "Choices not available for this Capability". Although the method may be supported by Capabilities in this implementation, it is not supported for this Capability. Usually, this return code indicates that the stripe depth has already been set in the parent StoragePool and may not be changed.
- 3, "Use [GetSupportedStripeDepths | GetSupportStripeDepthRange] instead". This return code tells the client that this stripe method is not supported, but the other stripe method is supported.

6.5.1.4 Getting Parity

```
uint32 GetSupportedParityLayouts(
    [Out] ParityLayout[])
```

This method is used to return the type of parity, non-rotated or rotated, that the capability supports.

Return codes:

- 0, "Method completed OK" means success.
- 1, "Method not supported".
- 2, "Choice not available for this Capability." Although the method may be supported by Capabilities in this implementation, it is not supported for this Capability. Usually, this return code indicates that the parity has already been set in the parent StoragePool and may not be changed.

6.5.2 Intrinsic Methods on StorageSetting

The following Intrinsic write methods are supported on StorageSetting:

- DeleteInstance,
- ModifyInstance.

6.5.3 Extrinsic Methods on StorageConfiguration

6.5.3.1 The RAID characteristics of the new or modified StoragePool

This design supports the implementation choice of the application of RAID striping during either the creation or modification of a StoragePool, StorageVolume, or LogicalDisk. Generally, without the implementation of Clause 15 Extent Composition Subprofile, a client cannot determine the storage elements that are used to represent the RAID striping without at least one StorageVolume or LogicalDisk. Even if the subprofile is supported, the client can make this determination only after each of the supported element types are created.

Once each of the storage element types are created, the client can use the StorageExtents on which the storage element is based to determine the RAID striping type applied. The RAID group is represented by a CompositeStorageExtent instance.

If the ExtentStripeLength property is not supported by an implementation, this design does not provide for interoperable behavior in the creation or modification of StoragePools, StorageVolumes, or LogicalDisks to provide reference to member StorageExtents.

6.5.3.2 Element Naming

Several of the following methods allow a client to 1) specify a name for the storage element that is being created or 2) change the name of a storage element being modified.

If the implementation supports the naming of storage elements, then the ElementName property reports the name assigned to the storage element. If the implementation creates a name even when the client does not specify one, then this element contains that system defined name. If the implementation does not create a name for the storage element when the client does not specify a name, then this property should be null. If the implementation does not support the naming of elements and the client provides a value in the ElementName

parameter of one of the following methods that specify an `ElementName` parameter, then the implementation shall reject the method call.

EXPERIMENTAL

The possible `ExtentStripeLengths`, `ExtentStripeDepths`, and `ParityLayouts` for a given `StorageCapabilities` may be fetched using these methods in that class:

- `GetSupportedStripeLengths()`
- `GetSupportedStripeLengthRange()`
- `GetSupportedParityLayouts()`
- `GetSupportedStripeDepths()`
- `GetSupportedStripeDepthRange()` methods

These methods are useful when the `ExtentStripeLength`, `ExtentStripeDepth`, and `ParityLayout` values in the given instance of `StorageCapabilities` are expressed in a range, where the minimum and the maximum are not equal.

EXPERIMENTAL

6.5.3.3 CreateOrModifyStoragePool

```
uint32 CreateOrModifyStoragePool(
    [In] string ElementName
    [Out] CIM_ConcreteJob ref Job,
    [In] CIM_StorageSetting ref Goal,
    [In,out] UInt64 Size,
    [In] string InPools[ ],
    [In] string InExtents[ ],
    [Out] CIM_StoragePool ref Pool);
```

This method is used to create a `StoragePool` from either a source `StoragePool` or a list of `StorageExtents`. Any required associations (such as `HostedStoragePool`) are created in addition to the instance of `StoragePool`. The parameters are as follows:

- **Job:** If a `Job` was created as a side-effect of the execution of the method, then a reference to that `Job` is returned through this parameter.
- **Goal:** This is the `Service Level` that the `StoragePool` is expected to provide. This may be a null value in which case a default setting is used.
- **Size:** As an input this shall be the desired size of the `StoragePool`. It may be null, in which case all passed-in capacity (as specified by `InExtents` and `InPools`) shall be used to create the pool. If it is not possible to create a `StoragePool` of at least the desired size, a return code of "Size not supported" shall be returned with size set to the nearest supported size.
- **InPools[]:** This is an array of strings containing `Object` references (see 4.11.5 of *DMTF DSP0200 CIM Operations over HTTP* for format) to source `StoragePools`.
- **InExtents[]:** This is an array of strings containing `Object` references (see 4.11.5 of *DMTF DSP0200 CIM Operations over HTTP* for format) to source `StorageExtents`. An array of source `StoragePools` or an array of source `StorageExtents` or both can be defined. See 6.1.15.
- **Pool:** If the method completes without creating a `Job`, then the `Pool` parameter is the storage element that is created. Otherwise, the `Pool` parameter may or may not be `NULL`. When the `Pool` parameter is `NULL`, then the storage element created can be determined by using the `Job` model.

6.5.3.4 The CreateOrModifyStoragePool method and the primordial StoragePool

A client may pass a reference to a primordial StoragePool in order to be explicit in indicating from which primordial StoragePool a concrete StoragePool needs to be created. If no StoragePool references are passed in the creation of a StorageVolume or LogicalDisk, the implementation shall determine the parent StoragePool based on the Goal and the Size.

A client may also pass a reference to a primordial StoragePool to express from what reserve to draw capacity if the capacity needed is greater than the total capacity represented by the input StoragePools and StorageExtents. Any capacity request, using the Size parameter, not satisfied by the referenced StoragePools and StorageExtents is drawn from the primordial StoragePool referenced. If no primordial StoragePool reference is passed and the capacity requested is greater than the referenced StoragePools and StorageExtents, then the method shall fail with the "Size not supported" return code. The use of a primordial StoragePool reference in this manner is not recommended, but the behavior is retained to maintain backward compatibility. The client should align the size requested to what can be satisfied by the concrete StoragePools and StorageExtents referenced.

A client should pass only concrete StoragePools when creating a StoragePool from several StoragePools.

6.5.3.5 DeleteStoragePool

```
uint32 DeleteStoragePool(
    [Out] CIM_ConcreteJob ref Job,
    [In] CIM_StoragePool ref Pool);
```

This method allows a client to delete a previously created StoragePool. All associations to the deleted StoragePool are also removed as part of the action. In addition, the RemainingManagedStorage of the associated parent primordial StoragePool will change accordingly.

NOTE This method will be denied ("Failed") if there are any AllocatedFromStoragePool associations where the deleted StoragePool is the Antecedent.

6.5.3.6 CreateOrModifyElementFromStoragePool

```
uint32 CreateOrModifyElementFromStoragePool (
    [In,
    string ElementName
    Values {"StorageVolume", "StorageExtent",
    "LogicalDisk"},
    ValueMap{"2", "3", "4"}]
    Uint16 ElementType;
    [Out] CIM_ConcreteJob ref Job,
    [In] CIM_StorageSetting ref Goal,
    [In, Out] Uint64 Size,
    [In] CIM_StoragePool ref InPool,
    [In, Out] CIM_LogicalElement ref TheElement );
```

This method allows an element of a type specified by the enumeration ElementType to be created from the input StoragePool. The parameters are:

- **ElementType:** This enumeration specifies what type of object to create.
- **Job:** If a Job was created as a side-effect of the execution of the method, then a reference to that Job is returned through this parameter. See *Information Technology – Storage Management, Part 3 Common Profiles*, Clause 27 Job Control Subprofile.
- **Goal:** This is the Service Level that the element is expected to provide. The Setting shall be a subset of the Capabilities available from the parent StoragePool. Goal may be a null value, in which case the default Setting for the StoragePool is used.
- **Size:** As an input this shall be the desired size of the element. It may be null, in which case all passed in capacity (as specified by InPool) shall be used to create the element. If it is not possible

to create an element of at least the desired size, a return code of "Size not supported" shall be returned with size set to the nearest supported size.

- InPool: This shall contain the reference to the source StoragePool.
- TheElement:
 - As Input: If the TheElement parameter is not null, then this method shall attempt to modify the reference element. Otherwise, this method shall attempt to create a new element.
 - As Output: If the method completes without creating a Job, then the TheElement is the storage element that is created. Otherwise, TheElement may be NULL. When the TheElement is NULL, the storage element that is created can be determined by using the Job model.

6.5.3.7 CreateOrModifyElementFromElements

```
uint32 CreateOrModifyElementFromElements(
    [In,
        Values {"Storage Volume", "Storage Pool",
            "Logical Disk"},
        ValueMap{"2","4", "5"}]
    unit16 ElementType,
    [In, Out] CIM_ConcreteJob REF Job,
    [In] CIM_ManagedElement REF Goal,
    [In, Out] unit64 Size,
    [In] CIM_StorageExtent REF InElements[],
    [In, Out] CIM_LogicalElement REF TheElement);
```

The parameters are:

- ElementType: This enumeration specifies the type of object to create.
- Job: If a Job was created as a side-effect of the execution of the method, then a reference to that Job is returned through this parameter. See *Information Technology – Storage Management, Part 3 Common Profiles*, Clause 27 Job Control Subprofile.
- Goal: This is the Service Level that the element is expected to provide. The Setting shall be a subset of the Capabilities available from the parent StoragePool. Goal may be a null value, in which case the default Setting for the StoragePool is used.
- Size: As an input this shall be the desired size of the element. It may be null, in which case all passed in capacity (as specified by InElements) shall be used to create the element. If it is not possible to create an element of at least the desired size, a return code of "Size not supported" shall be returned with size set to the nearest supported size.
- InElements: References to the StorageExtents to be used for the storage element creation or modification. The referenced StorageExtents shall be ComponentExtents of a single StoragePool, a parent of new or existing storage element. The parent StoragePool shall be a direct parent or an indirect parent, a grandparent, of the storage element. The InElements parameter of the CreateOrModifyElementFromElements() parameter is used to provide new StorageExtents to be used for this storage element. Therefore, the use of the parameter in the reduction of capacity for TheElement is invalid.
- TheElement:
 - As Input: If the TheElement parameter is not null, then this method shall attempt to modify the reference element. Otherwise, this method shall attempt to create a new element.
 - As Output: If the method completes without creating a Job, then the TheElement is the storage element that is created. Otherwise, TheElement may be NULL. When the TheElement is NULL, the storage element created can be determined by using the Job model.

6.5.3.8 ReturnToStoragePool

```
uint32 ReturnToStoragePool (
    [Out] CIM_ConcreteJob ref Job,
    [In] CIM_LogicalElement ref TheElement);
```

This method allows a client to delete a previously created element such as a StorageVolume.

EXPERIMENTAL

If TheElement is a SNIA_StorageVolume and SNIA_StorageVolume.CanDelete is set to false, then ReturnToStoragePool shall fail and shall return an error code of 6 ("In Use") or 4 ("Failed").

EXPERIMENTAL**EXPERIMENTAL****6.5.3.9 RequestUsageChange**

```
uint32 RequestUsageChange (
    [In,
        ValueMap { "2", "3" },
        Values { "Set", "Modify \Other\"
description only" }]
    uint16 Operation,
    [In] uint16 UsageValue,
    [In] string OtherUsageDescription,
    [Out] CIM_ConcreteJob ref Job,
    [In] CIM_LogicalElement ref TheElement);
```

The parameters are:

- Operation: This International Standard defines the usage of the 2 "Set" value for the parameters, which means to set the Usage to one of the possible usage values. This parameter is required.
- UsageValue: The usage value possible for the type of storage element, whose reference is passed to this method. This parameter is required.
- OtherUsageDescription: Not defined in this International Standard. This parameter is not required.
- Job: If a Job was created as a side-effect of the execution of the method, then a reference to that Job is returned through this parameter. See *Information Technology – Storage Management, Part 3 Common Profiles*, Clause 27 Job Control Subprofile.
- TheElement: This requirement parameter contains a reference to the storage element whose usage is to be changed.

If the storage element can not be changed to the requested usage because it is invalid to do so, then the implementation shall return an invalid parameter error.

EXPERIMENTAL**6.5.3.10 Return Values**

Each method has this set of defined return codes:

```
ValueMap {"0", "1", "2", "3", "4", "5", "6", "..", "4096", "4097"},
Values {"Job completed with no error", "Not Supported", "Unknown",
    "Timeout", "Failed", "Invalid Parameter", "In
Use", "DMTF Reserved", "Method parameters
checked - job started", "Size not supported"}]
```

Only the following return codes shall be supported:

- 0 - "Job completed with no error"
The method has completed immediately with no errors (and with no asynchronous execution required).
- 1 - "Not Supported"
This method is not supported at this time.
- 3 - "Timeout" or 4 - "Failed"
The provider has problems accessing the hardware (or other implementation-specific reasons)'. The provider should return a standard message communicating the nature of the value rather than returning this code.
- 5 - "Invalid Parameter"
One or more of the parameters are invalid (invalid object paths, for instance). The provider should return a standard message, communicating which parameters are invalid and why, rather than returning this code.
- 6 - "In Use"
The storage element is used for the basis for another storage element. For example, a client request that a StoragePool be deleted, but that StoragePool is the basis for another storage element. This return code may also indicate that the deletion of the specified storage element is not permitted because it is being used for another reason. This reason may be that the StoragePool on which this method is called does not permit this action. The reason may also be that the implementation does not allow this action for proprietary reasons.
- 4096 - "Method parameters checked - job started"
The method parameters have been checked, and the method is being executed asynchronously.
- 4097 - "Size not supported"
For a Create/Modify method, the requested size is not supported. The Size parameter and the size of the storage element is set to the nearest supported and larger size.). Only the methods that create or modify storage elements, other than their usage, shall return this code.

A vendor shall not extend the Value map to express vendor specific error situations not catered for by the standard messages.

EXPERIMENTAL

6.5.3.11 GetElementsBasedOnUsage

```
uint GetElementsBasedOnUsage(
    [In,
        ValueMap { "2", "3", "4", "5"}
        Values { "StorageVolume", "StorageExtent",
            "StoragePool", "Logical Disk",}]
    uint16 ElementType,
    [In] uint16 Usage,
    [In,
        ValueMap { "2", "3", "4" },
        Values { All, "Available Only", "In Use
            Only" }]
    uint16 Criterion,
    [In] CIM_StoragePool ref ThePool,
    [Out] CIM_ManagedSystemElement ref
    TheElements[ ]);
```

All input parameters are required. The parameters are:

- ElementType: This enumeration specifies the type of object to create.
- UsageValue: The usage value possible for the type of storage element as indicated by the ElementType parameter.

- Criterion: Specifies whether to retrieve all elements - 2 "All", available elements only - 3 "Available Only", or the elements that are in use - 4 "In Use Only".
- ThePool: Limits the search for the elements that satisfy the criteria in this StoragePool only. If null, all appropriate storage pools shall be included in the search.
- TheElements: Contains the array of references found to the storage element instances retrieved.

This method returns the following statuses:

- 0 - "Completed with No Error":
The method has completed immediately with no errors
- 1 - "Not Supported"
This method is not supported at this time.
- 3 - "Timeout" or 4 - "Failed"
The provider has problems accessing the hardware (or other implementation-specific reasons)'. The provider should return a standard message communicating the nature of the value rather than returning this code.
- 5 - "Invalid Parameter"
One or more of the parameters are invalid (invalid object paths, for instance). The provider should return a standard message, communicating which parameters are invalid and why, rather than returning this code.

EXPERIMENTAL

6.5.4 Extrinsic Methods on StoragePool

6.5.4.1 General

The Extrinsic methods on StoragePool return sizes in units of bytes. These methods, each described in this section, are:

- GetSupportedSizes
- GetSupportedSizeRange
- GetAvailableExtents

6.5.4.2 GetSupportedSizes

```
uint32 GetSupportedSizes(
    [In] uint16 ElementType,
    [In] CIM_StorageSetting ref Goal,
    [Out] uint64 Sizes[ ]);
```

The parameters are:

- ElementType: This enumeration specifies what type of object to create.
- Goal: The Service Level the element is expected to provide. The setting shall be a subset of the Capabilities available from the parent StoragePool. Goal may be a null value, in which case the default Setting for the StoragePool shall be used by the implementation.
- Sizes: An array containing all the possible sizes of an element in a creation or modification operation.

For a given Goal, this method returns discrete possible sizes of child elements, e.g., StoragePool, StorageVolume or LogicalDisk, that can be created or modified using capacity from the StoragePool. If the Goal is not supplied, the default Setting for the StoragePool shall be used by the implementation. This method is used to return the sizes of contiguous ranges of blocks of the pool that can be used individually or in combination with other extents to create or

modify storage pool or storage elements. For example, an implementation can use this method to return the sizes of disks, imported extents, or remaining extents that can be used in the storage assignment operation. This method is also useful if the possible sizes do not differ by a fixed size and thus cannot be reported by the `GetSupportedSizeRange` method. A summation in this case is the integer resulting from the addition any of the elements. The summations of the possible sizes shall not be returned from this method. The implementation should return the sizes of unassigned or remaining component extents that are appropriate for that Goal.

For example, if the returned sizes in gigabytes are {10, 15, 17, 21}, the summations include {25, 27, 31, 32, 36, 63}. It is the responsibility of the client to calculate the summations.

Any one of the returned sizes or any one of the summations of the returns shall be acceptable by the implementation as a possible size for a supported storage assignment using the element type and goal. If the size of unassigned or remaining storage extents is repeated in this set of storage extents, the repetition of size shall be reflected in the sizes returned. It is necessary to duplicate sizes so that the client can calculate the summations.

If the implementation supports zero size StoragePools (also known as an "empty" storage pool) or StorageVolumes, the returned Sizes parameter will have an entry with the value of 0. For example, if the `GetSupportedSizes` method is called with `ElementType` set to `StoragePool`, and an array of Sizes containing [0, 20, 22, 25] is returned, it indicates it is possible to create a 0 size (i.e. an empty) StoragePool, as well as other StoragePool sizes – namely 20, 22, and 25.

6.5.4.3 GetSupportedSizeRange

6.5.4.3.1 General

```
uint32 GetSupportedSizeRange(  
    [In] uint16 ElementType,  
    [In] CIM_StorageSetting ref Goal,  
    [Out] uint64 MinimumVolumeSize,  
    [Out] uint64 MaximumVolumeSize,  
    [Out] uint64 VolumeSizeDivisor);
```

- `ElementType`: This enumeration specifies what type of object to create.
- `Goal`: The service level the element is expected to provide. The Setting shall be a subset of the Capabilities available from the parent StoragePool. Goal may be a null value, in which case the default Setting for the StoragePool shall be used by the implementation.
- `MinimumVolumeSize`: The minimum size an element can take on either as a creation or modification operation.
- `MaximumVolumeSize`: The maximum size an element can take on either as a creation or modification operation
- `VolumeSizeDivisor`: The value used to determine sizes between `MinimumVolumeSize` and `MaximumVolumeSize`.

This method is used to determine the possible sizes of child element, e.g., StoragePool, LogicalDisk, and StorageVolume, that can be created or modified using capacity drawn from the StoragePool. This method is useful when the number of possible sizes is so voluminous that reporting each discrete size would be impractical. This method reports the continuum of discrete sizes between the minimum and maximum size as defined by intervals of the divisor.

The range of possible values between the values reported by `MinimumVolumeSize` and `MaximumVolumeSize` shall be defined as:

- next integer value greater than `MinimumVolumeSize` that is divisible by `VolumeSizeDivisor`,
- next integer value less than `MaximumVolumeSize` that is divisible by `VolumeSizeDivisor`,

- and every integer in between these integers that is divisible by VolumeSizeDivisor.

The possible values returned from this method shall include the MinimumVolumeSize, MaximumVolumeSize, and the range of values in between. Neither the MinimumVolumeSize nor the MaximumVolumeSize are required to be divisible by the VolumeSizeDivisor. For example, if given a MinimumVolumeSize of 10, a MaximumVolumeSize of 50, and VolumeSizeDivisor of 10, the possible size values would be 10, 20, 30, 40, and 50.

A client can calculate the discrete sizes by calculating the ceiling of the MinimumVolumeSize or the floor MaximumVolumeSize, then using one of these calculated values and the VolumeSizeDivisor to determine the discrete possible values within the range.

For example, given

MinimumVolumeSize = 35 GB
MaximumVolumeSize = 225 GB
VolumeSizeDivisor = 10 GB

ceiling(35/10) = 4
floor(225/10) = 22

the next possible size after the minimum, 35, is 4 * VolumeSizeDivisor, or 40 GB.
the next possible size after that is 5 * VolumeSizeDivisor, or 50 GB.
the next possible size before the maximum, 225, is 22 * VolumeSizeDivisor, or 220 GB.

sizes = {35, 40, 50, 60 ... 210, 220, 225}

Any one of the returned sizes shall be acceptable by the implementation as a possible size for a supported storage assignment using the element type and goal. The result size of the storage assignment or allocation may be greater than the size requested by the client. The result size should be greater than or equal to the requested size. The result size should be less than the next size greater than requested size that is divisible by the VolumeSizeDivisor.

It is not required that there be a relationship between the sizes returned from this method and the component extent sizes of the implementation as report by implementing the Extent Composition.

Both or either method may be supported by a storage subsystem, either as a decision made at implementation time or as a variable that depends on the state of the StoragePool. For example, when a StoragePool is first created allowing for possible sizes to be in 1 024-byte blocks, the GetSupportedSizeRange method should be used to report possible sizes. This example StoragePool does not relocate blocks to avoid fragmentation of the capacity. As StorageVolumes or LogicalDisks are drawn from and returned to the StoragePool, the capacity becomes fragmented. In this case, the GetSupportedSizes method should be used to report the non-continuous regions of capacity that may be used for element creation. There are storage systems that can allocate the StorageVolume or LogicalDisk only in whole disks that need not be of uniform size; such storage systems support only the GetSupportedSizes method.

Both methods may be supported at the same time and may report different values when discontinuous and contiguous capacity is present in the StoragePool. In this case, the GetSupportSizes method is used to report the fragments of available capacity. The remaining contiguous capacity is reported as the largest element size possible. The GetSupportSizeRange is used to report element sizes that may be drawn from the contiguous capacity.

If there is no notion of continuity as being a stable state of the system, e.g., capacity is continuously and automatically being defragmented, the GetSupportSizeRange method should be used.

If the implementation supports zero size StoragePools (also known as an "empty" storage pool) or StorageVolumes, the returned MinimumVolumeSize parameter will have the value of 0.

6.5.4.3.2 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 do not complete successfully, then either the methods are not supported or the other method should be used. The GetSupportSizes method can notify the SMI-S client that it should use the GetSupportSizeRanges instead; the GetSupportedSizeRange method can notify the SMI-S client that it should use the GetSupportedSizes method instead.

6.5.4.3.3 GetAvailableExtents

```
uint32 GetAvailableExtents(
    [In] CIM_StorageSetting REF Goal,
    [Out] CIM_StorageExtent REF AvailableExtents[
    ]);
```

This method is used to retrieve the available StorageExtents—ComponentExtents of the StoragePool—that do not form the basis for StorageVolumes and LogicalDisks allocated from the StoragePool. If a NULL is passed for a Goal, then all the available ComponentExtents of the StoragePool are returned.

The StorageExtent references returned from this method refer to a subset of the StorageExtents associated to the StoragePool via ConcreteComponent, AssociatedComponentExtent, and AssociatedRemainingExtent. The StorageExtents referenced by the output of this method may not equal the set of Component StorageExtents because of any of the following reasons:

- The excluded StorageExtents may not be used with the Goal.
- The excluded StorageExtents may not be used for vendor-specific reasons.
- The excluded StorageExtents may not be used because of a usage restriction.

This method is designed as a companion for the CreateOrModifyElementFromElements method. A client may fetch the StoragePool's available ComponentExtents and attempt to call CreateOrModifyElementFromElement, or the client may use this method and have the agent provide the available StorageExtents. However, note it is possible that even though a StorageExtent may appear to be available from the implementation's model, the implementation may not allow the StorageExtent to be used for vendor specific reasons.

6.5.4.4

6.5.4.4.1 General

Each method has this set of defined return codes:

```
ValueMap {"0", "1", "2", "3", "4", "5"},
Values {"Job completed with no error", "Not Supported", "Unknown",
"Timeout", "Failed", "Invalid Parameter"}]
```

- 0 - "Job completed with no error"
The method completes immediately with no errors (and with no asynchronous execution required).
- 1 - "Not Supported"
The implementation does not support the method.
- 5 - "Invalid Parameter"
One of the method parameters is incorrect (for instance invalid object paths).
- 3 - "Timeout" or 4 - "Failed"
The provider had problems accessing the hardware, or there were implementation-specific problems.

6.5.4.4.2 Storage Element Modification

Concrete StoragePools may be expanded, shrunk, or have their quality of service (QoS) changed (the Goal parameter) by a client.

This package does not define how primordial StoragePools are modified (if they can be modified) within a particular implementation.

The current capacity of a StoragePool is the value of the TotalManagedSpace property.

StorageVolumes and LogicalDisks may be expanded, shrunk, or have their quality of service (QoS) changed (the Goal parameter) by a client.

The current capacity of the StorageVolume, LogicalDisk, or StorageExtent is the ConsumableBlocks times the BlockSize.

Storage elements are StoragePools, StorageVolumes, and LogicalDisks.

Return values are:

- 5 "StoragePool QoS Change," 6 "StoragePool Capacity Expansion," 7 "StoragePool Capacity Reduction"

Within SupportedStoragePoolFeatures array within the StorageConfigurationCapabilities instance, indicates the types of StoragePool modification allowed.

- 11 "Storage Element QoS Change, 12 "Storage Element Capacity Expansion", and 13 "Storage Element Capacity Reduction"

Within the SupportedStorageElementFeatures array within the StorageConfigurationCapabilities instance, indicates the types of StorageVolume and LogicalDisk modifications allowed.

An implementation may support one or more of these options. If the implementation supports capacity expansion or capacity reduction options and the QoS change option, then it shall support the capacity change and the QoS change simultaneously in the modification of a given storage element.

A client can determine the resultant usable capacity to which a storage element may be changed by using the GetSupportedSizes() and GetSupportedSizeRange() methods on the parent StoragePool. These methods provide the possible storage capacity for new storage elements and for the modification of existing storage elements given a QoS goal. To obtain a size to use for storage element modification, the client simply select a size returned from the GetSupportedSizes() method or a size within the range returned from GetSupportedSizeRange() method.

Generally, the attempted StoragePool modification shall be characterized as a storage capacity expansion if the new capacity (the Size parameter) is greater than the current value of the TotalManagedSpace property of the StoragePool to be modified. Likewise, the attempted StoragePool modification shall be characterized as a storage capacity reduction if the desired new capacity (the Size parameter) is less than the current value of the TotalManagedSpace property of the StoragePool to be modified.

Generally, the attempted StorageVolume or LogicalDisk modification shall be characterized as a storage capacity expansion if the new capacity (the Size parameter) is greater than its current capacity. Likewise, the attempted StorageVolume or LogicalDisk modification shall be characterized as a storage capacity reduction if the desired new capacity (the Size parameter) is less than its current capacity.

A storage element may also be modified by providing the references to component StorageExtents. The list candidate component StorageExtents shall be provided through the execution of the GetAvailableExtents() method on the parent StoragePool. For example, the SMI-S Client determines which StorageExtents to use from the returned list based on their performance characteristics or their relationship to network ports or primordial storage.

A StoragePool's capacity may be expandable by providing the references to existing component StorageExtents of the StoragePool and additional references to component StorageExtents. A StoragePool's capacity may be reducible by providing references to some, but not all, of the current component StorageExtents of the StoragePool. If the summary of the capacity of the referenced input StorageExtents is greater than the TotalManagedSpace of the StoragePool, then this action shall be characterized as a capacity expansion. If this summary is less than the TotalManagedSpace of the StoragePool, then this action shall be characterized as capacity reduction.

A StorageVolume's or LogicalDisk's capacity may be expandable by providing references to additional component StorageExtents of the parent StoragePool. The capacity of a StorageVolume or LogicalDisk shall not be reducible by providing references to StorageExtents.

The capacity of storage elements that have only one member StorageExtent can only be reduced by passing a reference to the existing member and specifying a capacity, using the Size parameter, that is smaller than the current size of the storage element.

The specified Size parameter (in bytes), along with the International Standard of member StorageExtents, indicates how much of the provided StorageExtents is to be used for the storage element. The specified size represents the desired consumable capacity of the storage element. The capacity of the StorageExtent may be equal to either the capacity drawn in its creation from a parent StorageExtent or StoragePool or to the capacity that may be drawn from it in the creation of a dependent storage element. No direct comparison may be made by the client between the desired capacity and the capacity of the StorageExtents.

If the capacity desired is equal to the capacity of the storage element and the QoS is not altered, then the implementation shall return no error and start no job.

If the capacity requested is larger than is consumable given a QoS (new or existing) from the referenced StorageExtents or StoragePools, then that capacity shall be drawn from the parent primordial StoragePool. The effect of passing a capacity less than the current capacity of the storage element shall be to make available or free the capacity in the member StorageExtents to the difference between the current capacity of the storage element and the new capacity of the storage element. The amount of capacity freed depends on the virtualization (e.g., RAID method) employed in the previous configuration of the storage element. An invalid parameter error shall be produced if the capacity in bytes passed is less than the current capacity but greater than the capacity realizable from the StorageExtents referenced given a QoS. The size of a StorageExtent is the NumberOfBlocks times the BlockSize. The capacity of the StorageExtents references can be calculated. It is the sum of the sizes of all StorageExtents.

The number of StorageExtents desired, including existing and additional StorageExtents, for a StorageElement minus the PackageRedundancy shall be equal to the ExtentStripeLength times the DataRedundancy specified in the existing QoS goal. Clause 15 Extent Composition Subprofile defines how to determine the number of primordial StorageExtents used.

The quality of service (QoS) of a storage element may be modified. Generally, a QoS change indicates a reorganization of computing resources to meet the new requirements—either additional or fewer computing resources are used.

If the QoS is being modified, then clients may not be able to determine if desired size of the storage element constitutes an expansion or reduction, as specified previously. Such a modification shall be non-destructive to the data stored.

The QoS of a StoragePool shall not be changeable if that StoragePool has children storage elements. However, the package redundancy of parental StoragePools may be changed by changing the number of spare StorageExtents. See Clause 13 Disk Sparing Subprofile.

In the totality of this design, a SMI-S Client may change one of the following:

- The QoS,

- The Size (capacity),
- The Size and the member StorageExtents,
- Only the member StorageExtents.

A SMI-S Client may not change the QoS and the member StorageExtents. There is no mechanism for a SMI-S Client to determine the quorum of StorageExtents for a given QoS if ExtentStripeLength is not provided.

6.6 Client Considerations and Recipes

6.6.1 Representative Instance Diagram

Figure 16: "Representative Block Service Instance Diagram" shows the classes and associations needed to model a single StoragePool with two StorageVolumes.

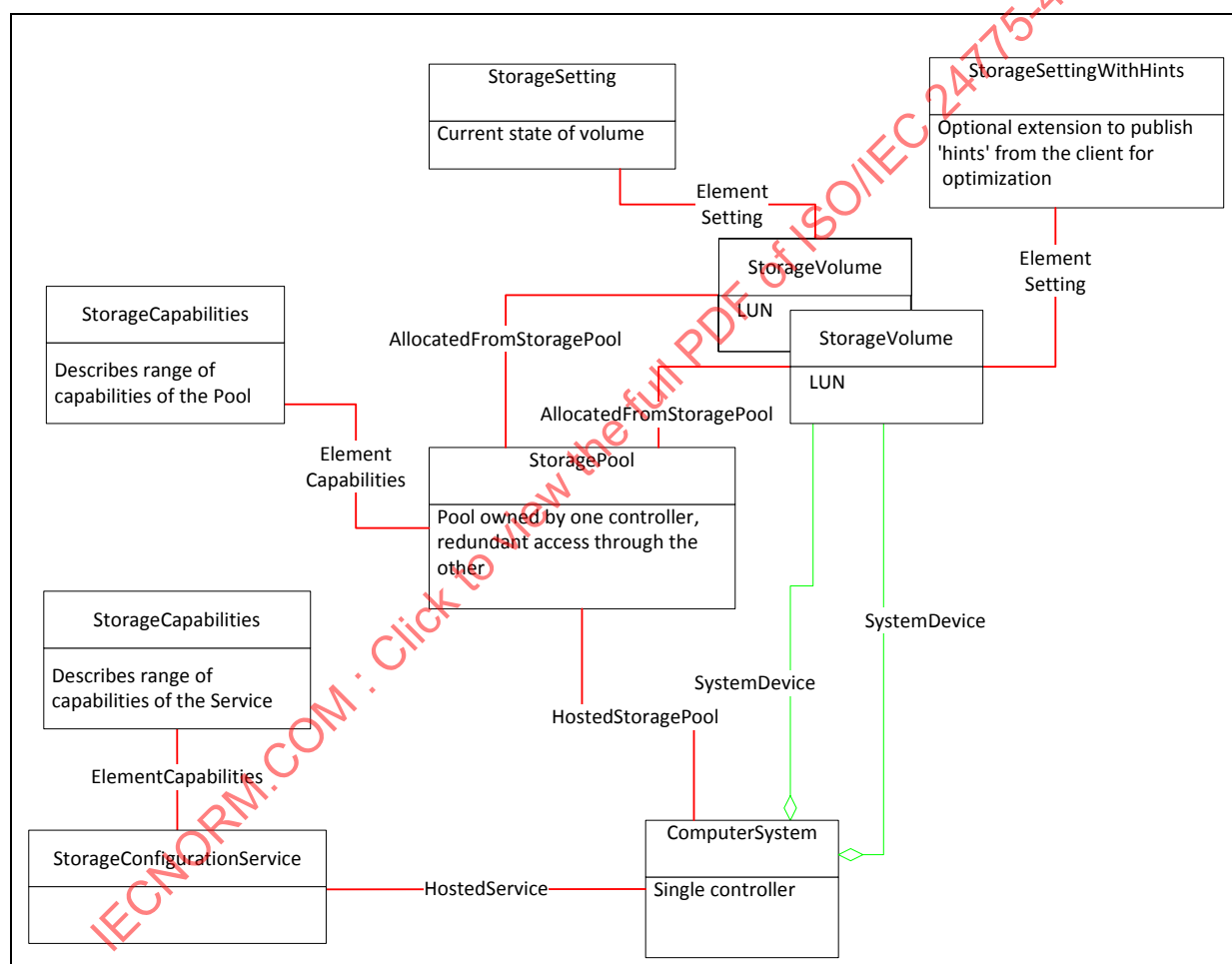


Figure 16 - Representative Block Service Instance Diagram

6.6.2 Goals and Settings

A implementation may persist the properties of the Setting as they were when the Setting was used to perform a configuration operation. However, the implementation may also construct the Setting given the current quality of service provided. An implementation of this package should retain the properties of the Setting as they were when the Setting was used as a Goal. For example, a client requests a package redundancy 2, the implementation is restarted and therefore cannot retrieve; the implementation sets this value to the current value of 1. Unless the client maintained the state of Setting as well, it will not be able to detect the difference between the initial Setting state and the current state for package redundancy, in the StorageVolume or LogicalDisk, for example.

If a client specifies a goal asking for no single point of failure, the implementation shall return an error if the system is not capable of supporting that goal. However, if a client specifies that single points of failure are allowed, the implementation may return storage that has potential single points of failure or it may return storage that has no single points of failure. In other words, the system may return a storage that is more capable than what the client has asked for.

A client may request more data redundancy and package redundancy than what is required for the particular RAID level. An implementation may provide more of these redundancies than is required for its RAID levels. If allowed, the client request of additional data redundancy indicates that additional copies of the data are requested. If allowed, the client request of additional package redundancy results in additional drives, for example, being assigned to this storage element. The redundant package may be overassigned (e.g., assigned as extra packages for more than one storage element), or it may be dedicated. See Clause 13 Disk Sparing Subprofile for details on modeling the sparing functionality itself. In other words, these Goal properties can be used to assign additional copies of the data and redundancy at creation or modification time of a StoragePool, StorageVolume, or LogicalDisk.

6.6.3 Representative StoragePool Creation Example

Figure 17: "StoragePool Creation - Initial State" shows the initial state of the block storage system, a single primordial StoragePool that advertises its capabilities. The GetSupportedSizes() and GetSupportedSizeRange() methods determine what sizes of StoragePools can be created from the primordial StoragePool, given a goal StorageSetting. Alternatively, if the StoragePool is to be created from StorageExtents, GetAvailableExtents() obtains a list of available ComponentExtents of the StoragePool that also match the Goal.

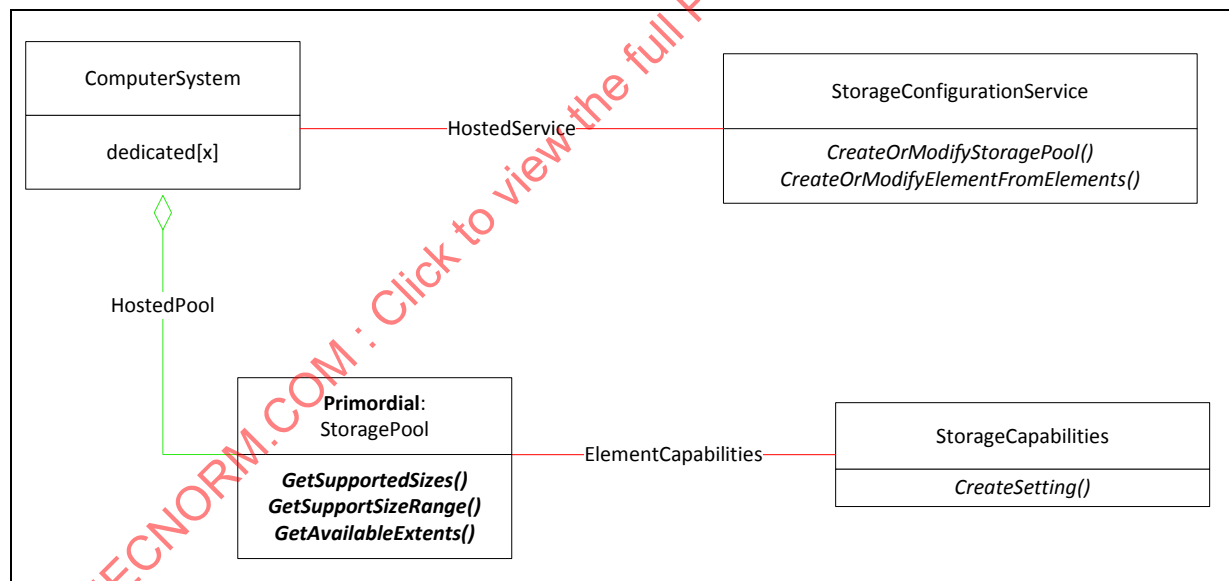


Figure 17 - StoragePool Creation - Initial State

Next, (Figure 18: "StoragePool Creation - Step 1") 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 block storage system supports **StorageSettingWithHints**, an instance of this subclass is created rather than the **StorageSetting** superclass. Alternatively, the client can use one of the predefined **StorageSetting** instances. Pre-existing **Settings** can be located by using the **StorageSettingsAssociatedToCapabilities** association for factory or pre-defined settings or by using the **StorageSettingsGeneratedFromCapabilities** class, where the **StorageSetting.ChangeableType** = "2" ("Changeable - Persistent"). These **Settings** have been generated but were modified to persist.

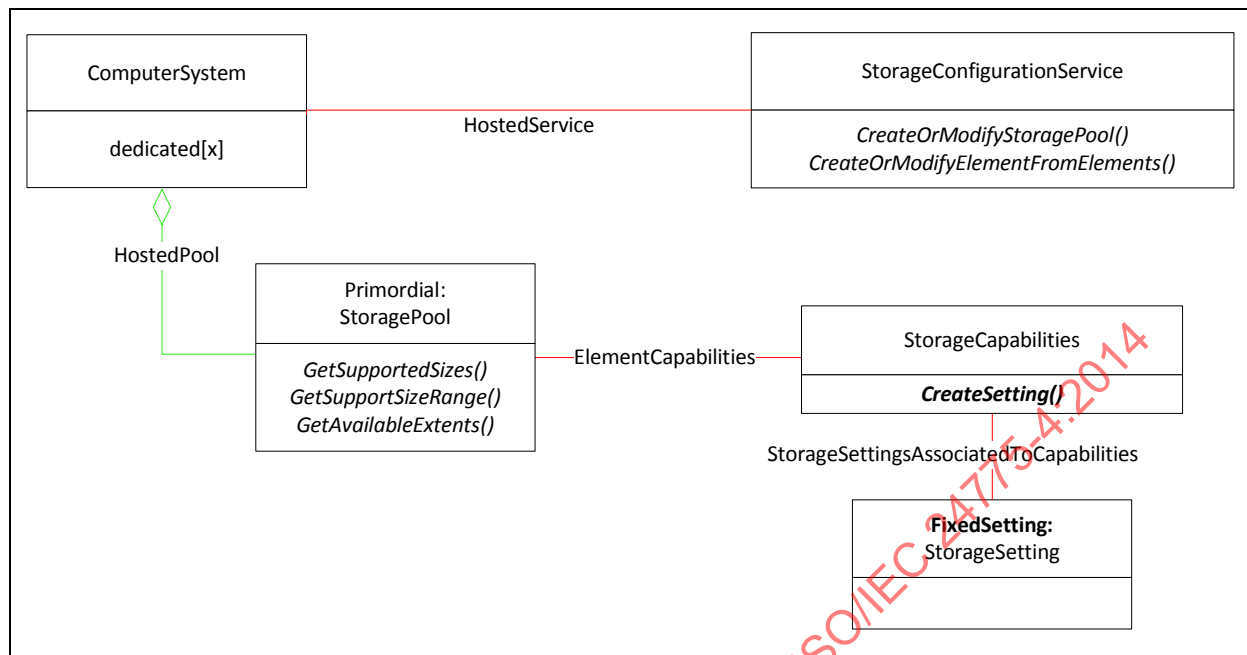


Figure 18 - StoragePool Creation - Step 1

Once this generated Setting has been altered as required or, alternatively, a pre-defined Setting used, the Goal Setting is passed as an argument to the CreateOrModifyStoragePool method in the StorageConfigurationService. (Shown in Figure 19: "StoragePool Creation - Step 2").

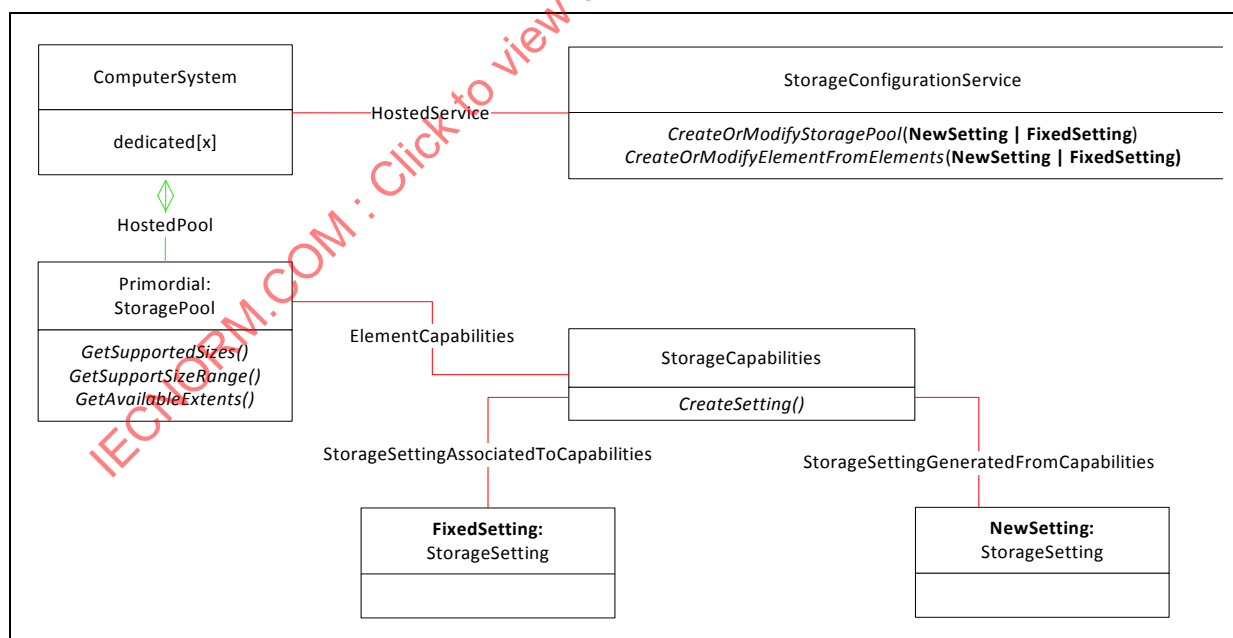


Figure 19 - StoragePool Creation - Step 2

Alternatively, the client can create the StoragePool by passing the Goal, the desired ComponentExtents, and a "Pool" ElementType to CreateOrModifyElementFromElement. If a Size is passed as well, the size shall be equal to or less than the consumable size (in blocks) of the desired ComponentExtents. The list of available StorageExtents is best retrieved using the GetAvailableExtents() method. If the Size is less than the desired StorageExtents by less

than the smallest StorageExtent passed, then one of the StorageExtents is partitioned into used and free parts. See 6.1.15.

The StoragePool is then created, as shown in Figure 20: "StoragePool Creation - Step 3". If the generated Setting was used as the Goal, then this temporary StorageSetting is replaced with an equivalent object linked to the new StoragePool with ElementCapabilities.

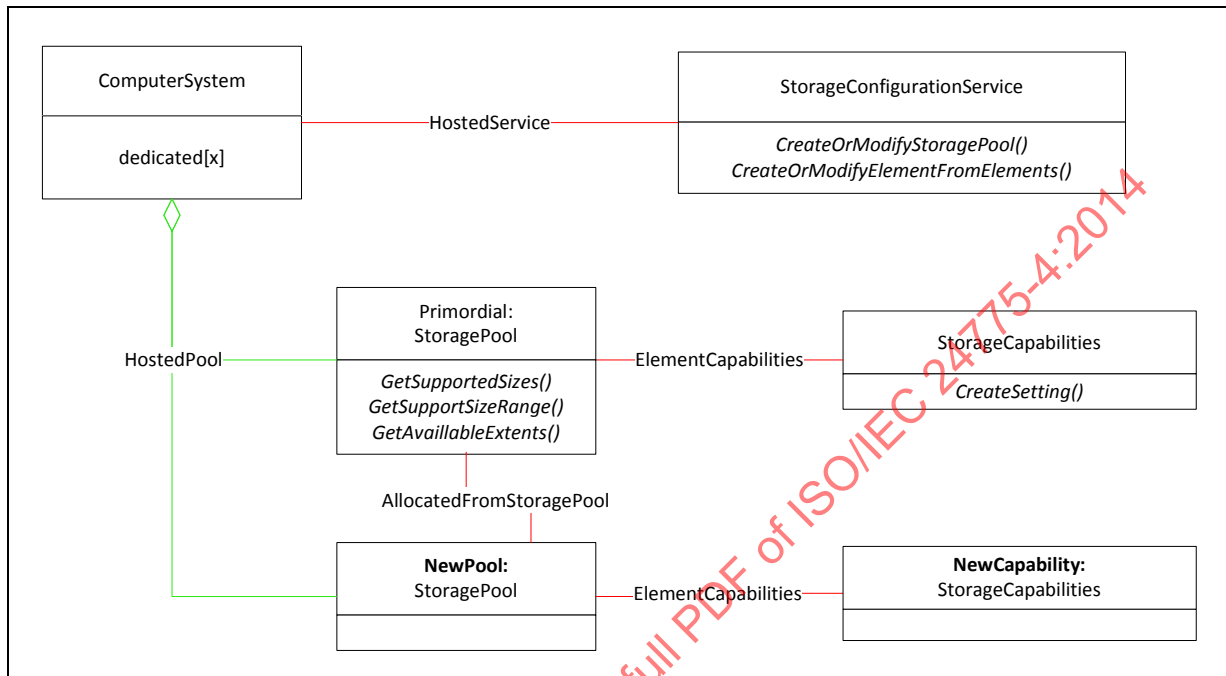


Figure 20 - StoragePool Creation - Step 3

6.6.4 Representative example of StorageVolume or LogicalDisk Creation

Similarly to StoragePools, a client chooses a suitable source StoragePool by referencing the StorageCapabilities objects and using the GetSupportedSizes() and GetSupportSizeRange() methods, given a goal Setting. Alternatively, a client can retrieve the available ComponentExtents of the StoragePool, given a goal StorageSetting, with the GetAvailableExtents() methods. The client may create a StorageVolume or LogicalDisk by specifying a size, source StorageExtents, or a combination, as shown in Figure 21: "StorageVolume Creation - Initial State".

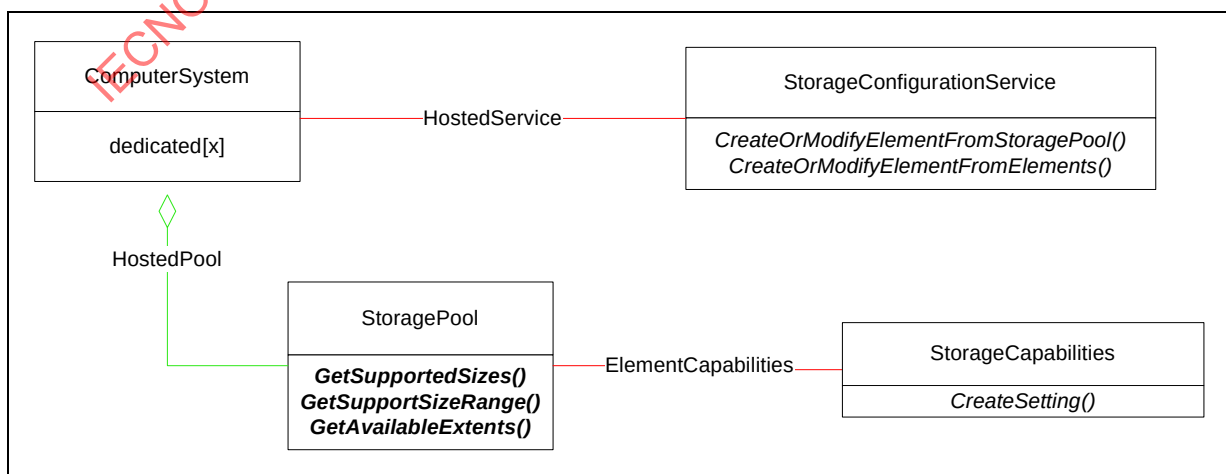


Figure 21 - StorageVolume Creation - Initial State

Once a suitable StoragePool is found, a StorageSetting instance can be created using the CreateSetting method on the StorageCapabilities object. See Figure 22: "StorageVolume Creation - Step 1". If a suitable StorageSetting already exists, it can be used instead. Pre-existing Settings can be located by using the StorageSettingsAssociatedToCapabilities association, for factory or pre-defined settings, or by using the StorageSettingsGeneratedFromCapabilities where the StorageSetting.ChageableType = "2" ("Changeable - Persistent"). These Settings have been generated but were modified to persist, as illustrated in Figure 22: "StorageVolume Creation - Step 1". Another Setting already associated to a storage element can be used as a goal, but it shall not be modifiable.

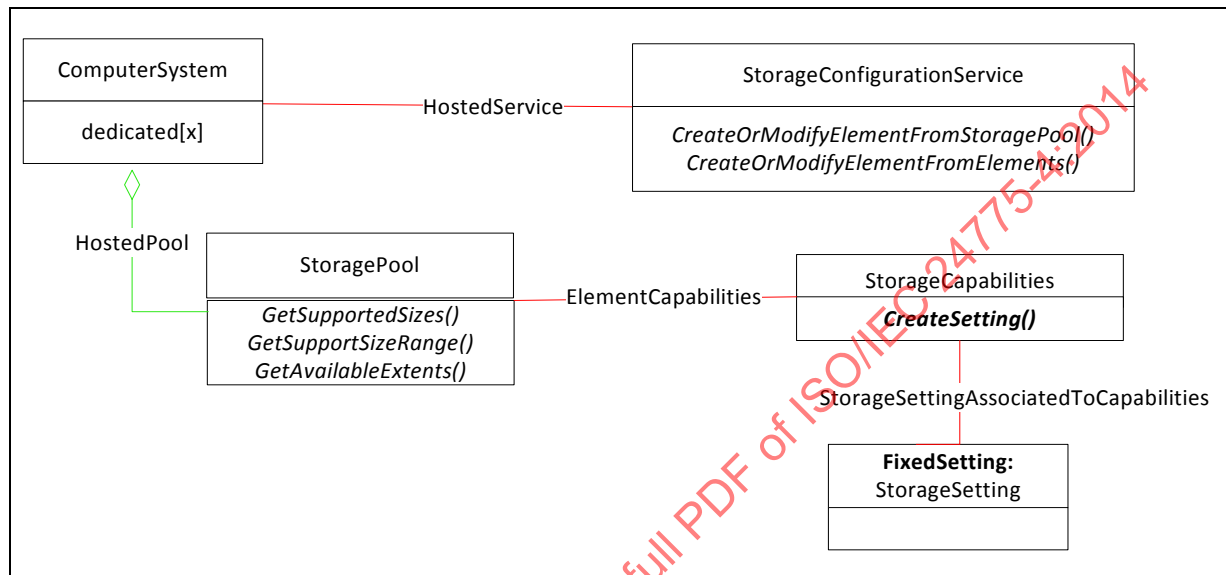


Figure 22 - StorageVolume Creation - Step 1

If a new Setting is created, it is linked back to the originating StorageCapabilities object until it is used as an argument in a StorageConfiguration method. See Figure 23: "StorageVolume Creation - Step 2". Alternatively, the client can create the StorageVolume or LogicalDisk, for example, by passing the Goal, the desired ComponentExtents, and a ElementType to CreateOrModifyElementFromElement. If a Size is passed as well, the size shall be equal to or less than the consumable size (in blocks) of the desired ComponentExtents. The list of available StorageExtents is best retrieved using the GetAvailableExtents() method. If the Size

is less than the desired StorageExtents by a size less than smallest StorageExtent passed, then one of the StorageExtents is partitioned into used and free parts. See 6.1.15.

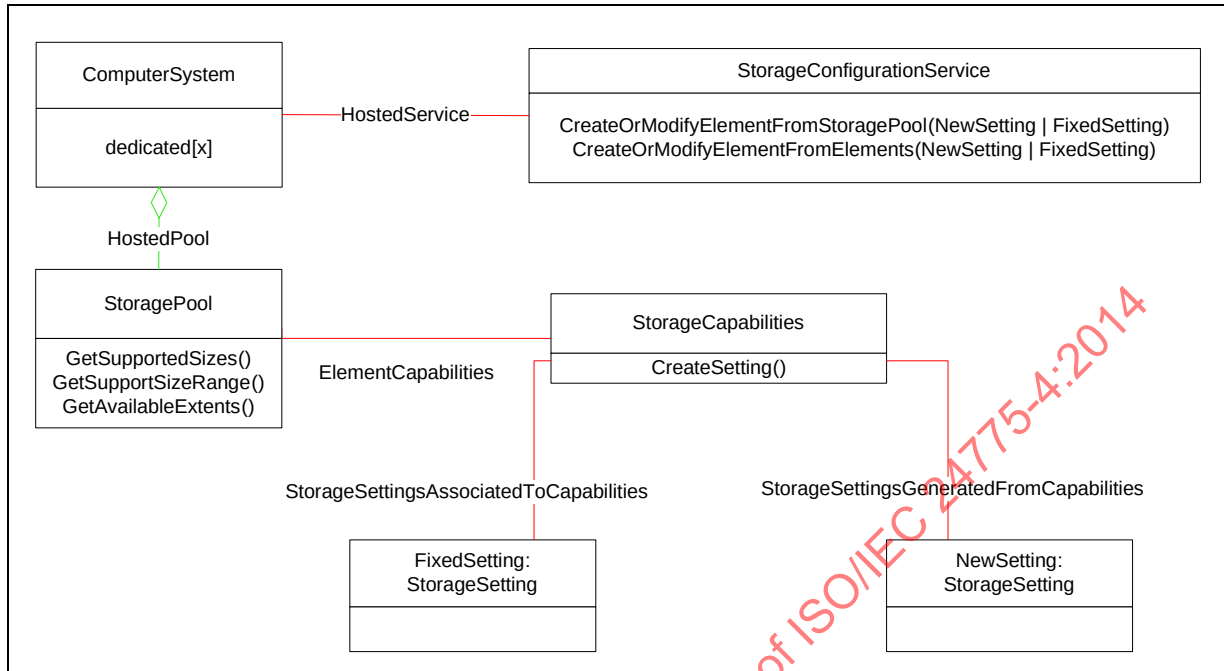


Figure 23 - StorageVolume Creation - Step 2

Once the StorageVolume has been created, the new or existing Setting is associated to the new storage element using the ElementSettingData association. The new Setting and the Goal setting may not be the very same instance. The client cannot assume that the instances are the same instance. See Figure 24: "StorageVolume Creation - Step 3".

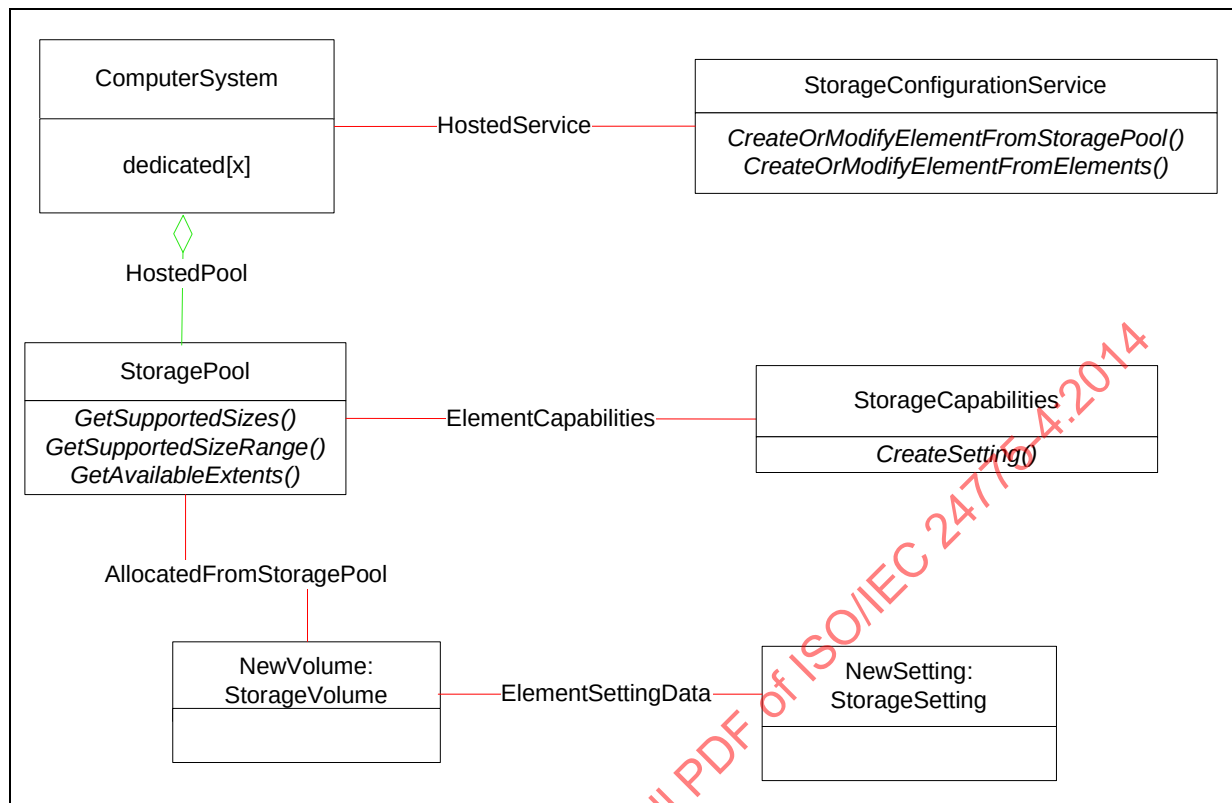


Figure 24 - StorageVolume Creation - Step 3

6.6.5 Summarize the StoragePools in a block storage system and verify the capacity reported

```

// DESCRIPTION
// This recipe retrieves and validates the total, remaining and consumed
// storage
// pool space on a block server.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTIONS
// 1. The object name for the device, CIM_ComputerSystem, of interested
// has
// previously been identified and defined in the $BlockServer->
// variable.

// Step 1. Retrieve the storage pools on the device.
$Pools[] = Associators($BlockServer->,
    "CIM_HostedStoragePool",
    "CIM_StoragePool",
    "GroupComponent",
    "PartComponent",
    false,
    false,
    {"TotalManagedSpace", "RemainingManagedSpace"})

// Step 2. Summarize the space consumed and available in each storage
// pool.
for (#i in $Pools[]) {

```

```
#totalSpace = $Pools[#i].TotalManagedSpace
#remainingSpace = $Pools[#i].RemainingManagedSpace
$Pool-> = $Pools[#i].getObjectPath()

// Step 3. Retrieve the space consumed by each element allocated from
// the
// storage pool.
$Allocs[] = References($Pool->,
    "CIM_AllocatedFromStoragePool",
    "Antecedent",
    false,
    false,
    {"SpaceConsumed"})

#allocSpace = 0
for (#j in $Allocs[]) {
    #allocSpace = #allocSpace + $Allocs[#j].SpaceConsumed
}
if (#totalSpace != #allocSpace + #remainingSpace) {
    <ERROR! Device does not correctly represent capacity>
}
}
```

6.6.6 Conditional: Create StoragePool and Storage Element on Block Server (e.g., Array or Volume Manager)

```
// DESCRIPTION
// The goal of this recipe is to create a storage element with the
// maximum capabilities of the block server.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
// 1. The storage configuration service is supported indicating the
// storage configuration is permitted. This is the condition for the
// recipe.
// 2. A reference to a CIM_ComputerSystem storage array is previously
// defined in the $BlockServer-> variable
// 3. The settings for the new Storage Pool and Storage Volume or Logical
// Disk have
// following size:
// #RequestedSize = 10 * 1024 * 1024 * 1024 // 10 GB
// 4. #StorageElementClass is set to the class name of the element being
// created
// like CIM_StorageVolume or CIM_LogicalDisk.
// 5. #ElementType is set to the element to created
// See CreateOrModifyElementFromStoragePool.ElementType

// Function GetMostCapable
// Get the capabilities that have the maximum DataRedundancy and
// PackageRedundancy
// Input:
```

```

// An array of StorageCapabilities instances associated to the
// StoragePool.
sub REF GetMostCapable($CapabilitiesOffered[])
{
    <Sort the $CapabilitiesOffered[] so that the capability with the
    greatest DataRedundantMax, PackageRedundancyMax, and
    NoSinglePointOfFailure in the last element in the array.
    NoSinglePointOfFailure == true is greater than
    NoSinglePointOfFailure == false
    >

    return $CapabilitiesOffered[$CapabilitiesOffered.length-1]
}

// Function PoolSizeAvailable
// A return value of 0 means that no size is available
sub unit64 PoolSizeAvailable($PoolToDrawFrom->,
    $StorageSetting->, #RequestedSize, #RequestedElementType)

    #ResultSize = 0
    %InArguments["ElementType"] = #RequestedElementType
    %InArguments["Goal"] = $StorageSetting->
    #MethodReturn = InvokeMethod(
        $PoolToDrawFrom->,
        "GetSupportedSizes",
        %InArguments,
        %OutArguments)
    if(#MethodReturn == 0)
    {
        // this method is supported
        #SupportedSizes[] = %OutArguments["Sizes"]
        < Amend to the #SupportedSizes[] all possible combinations of
        summations of the values provided in the array >
        #i = 0
        #max = #SupportedSizes[].length
        while(#i < #max && #RequestedSize > #ResultSize)
        {
            #ResultSize = #SupportedSizes[#i++]
        }
        if(#RequestedSize > #ResultSize)
        {
            // we did not find a size
            #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"]
    #ResultSize = 0 // Set default case
    if(#RequestedSize >= #MinimumVolumeSize &&
        #RequestedSize <= #MaximumVolumeSize)
    {
        // Rounding up to next Size, which is dividable by Divisor
        #ResultSize = (#RequestedSize + (#VolumeSizeDivisor -
            (#RequestedSize MOD #VolumeSizeDivisor)))
    }
}
return #ResultSize
}

// MAIN
// Step 1. Get the configuration services and determine the service
// capabilities. Note that the device may not support storage
// configuration so it is possible that the service is not present and
// the desired management cannot be performed.
try {
    $Services->[] = AssociatorNames($BlockServer->,
        "CIM_HostedService",
        "CIM_StorageConfigurationService",
        null,
        null)
    // StorageConfigurationService and HostedService may not be
    // implemented
    // in the SMI Agent.
    if ($Services->[] == null) {
        <ERROR! Storage Configuration is not supported.>
    }
} catch (CIMException $Exception) {
    // StorageConfigurationService and/or HostedService may not be
    // included in
    // the model implemented at all if Storage Configuration is not
    // supported.
    if ($Exception.CIMStatusCode == CIM_ERR_INVALID_PARAMETER) {
        <ERROR! Storage Configuration is not supported.>
    }
}

// There should be only one storage configuration service
// Associated with the system
$StorageConfigurationService-> = $Services->[0]
$ServiceCapabilities[] = Associators(
```



```

$StorageConfigurationService->,
"CIM_ElementCapabilities",
"CIM_StorageConfigurationCapabilities",
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].SupportedAsynchronousActions[])
#SupportsElementCreation1 = contains(
    5, // Storage Element Creation
    $ServiceCapabilities[0].SupportedSynchronousActions[])
#SupportsElementCreation2 = contains(
    3, // StorageElementCreation
    $ServiceCapabilities[0].SupportedStorageElementFeatures[])
#ElementCreationProducesJob = contains(
    5, // Storage Element Creation
    $ServiceCapabilities[0].SupportedAsynchronousActions[])
// If a storage element can not be created and that storage element is
// neither created synchronously or asynchronously, then fail the test
if (!#SupportedElementCreation2 &&
    !(#SupportedElementCreation1 || #ElementCreationProducesJob))
{
    <ERROR! The StoragePool can be created, but the
    StorageElement creation is not supported.>
}

// Step 2. Enumerate over the CIM_HostedStoragePool associations to find
// all the StoragePools from which storage elements might be created.
$StoragePools[] = Associators(
    $BlockServer->,
    "CIM_HostedStoragePool",
    "CIM_StoragePool",
    null,
    null,
    false,
    false,
    {"InstanceID", "Primordial"})

// Step 3. For each StoragePool, follow the CIM_ElementCapabilities
// association to the StorageCapabilities of that pool. Compare the

```

```
// StorageCapabilities to the desired StorageSetting and find the
// best match.
$PoolToDrawFrom-> = null
for #i in $StoragePools[]
{
    // See if this pool has its own StorageConfigurationCapabilities.
    $PoolServiceCapabilities[] = Associators(
        $StoragePools[#i]->,
        `CIM_ElementCapabilities`,
        `CIM_StorageConfigurationCapabilities`,
        null,
        null,
        false,
        false,
        null)
    if( $PoolServiceCapabilities[]-> != null ) {
        #SupportsPoolCreation = contains(
            2, // Storage Pool Creation

            $PoolServiceCapabilities[0].SupportedSynchronousActions[] ) ||
        contains(
            2, // Storage Pool Creation

            $PoolServiceCapabilities[0].SupportedAsynchronousActions[])
        #PoolCreationProducesJob = contains(
            2, // Storage Pool Creation
            $ServiceCapabilities[0].SupportedAsynchronousActions[])
        #SupportsElementCreation1 = contains(
            5, // Storage Element Creation
            $ServiceCapabilities[0].SupportedSynchronousActions[])
        #SupportsElementCreation2 = contains(
            3, // StorageElementCreation

            $ServiceCapabilities[0].SupportedStorageElementFeatures[])
        #ElementCreationProducesJob = contains(
            5, // Storage Element Creation

            $ServiceCapabilities[0].SupportedAsynchronousActions[])

        if (!#SupportsPoolCreation &&
            !#SupportsElementCreation2 &&
            !(#SupportedElementCreation1 ||
                #ElementCreationProducesJob) )
        {
            <ERROR! The StoragePool can be created, but the
            StorageElement creation is not supported.>
        }
    } // end of if( $PoolServiceCapabilities[]-> != null )
    else {
```

```

        // Continue with global instance of
        StorageConfigurationCapabilities --
        // This Pool does not have StoragePool specific
        capabilities
    }

    // If we can not create Storage Pool, then find a 'concrete'
    // Storage Pool from which to create a Storage Element
    #UsePrimordial = false
    if(#SupportsPoolCreation)
    {
        #UsePrimordial = true
        #RequestedElementType = 2 // StoragePool
    }
    else
    {
        #RequestedElementType = #ElementType
    }
    if ($StoragePools[#i].Primordial == #UsePrimordial)
    {
        $CapabilitiesOffered[] = Associators(
            $StoragePools[#i].getObjectPath(),
            "CIM_ElementCapabilities",
            "CIM_StorageCapabilities",
            null,
            null,
            false,
            false,
            null)
        $StorageCapabilitiesOffered =
            &GetMostCapable($CapabilitiesOffered[])
        $PoolToDrawFrom-> = $StoragePool[#i].getObjectPath()

        // Step 4. Determine if the selected pool has enough space for
        // another pool.
        // If the block server supports hints, then the Storage Setting
        // returned
        // will contain default hints

        // Create a setting
        %InArguments["SettingType"] = 3 // Goal
        #ReturnValue = InvokeMethod(
            $StorageCapabilitiesOffered.getObjectPath(),
            "CreateSetting",
            %InArguments,
            %OutArguments)
        if (#ReturnValue != 0 || null)
        {
            <ERROR! Unable to create storage setting >
        }
        $GeneratedStorageSetting-> = %OutArguments["NewSetting"]
    }

```

```
// Determine the possible size, closest to the requested size
#PossibleSize = &PoolSizeAvailable(
    $PoolToDrawFrom->,
    $GeneratedStorageSetting->,
    #RequestedSize
    #RequestedElementType)
if(0 != #PossibleSize) // we found a size close to #RequestedSize
{
    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
        storage element>
}

// Step 5. Register for indications on configuration jobs
If(#PoolCreationProducesJob || #ElementCreationProducesJob)
{
    // '17' ("Completed") '2' ("OK")
    #Filter1 = "SELECT * FROM CIM_InstModification
        WHERE SourceInstance ISA CIM_ConcreteJob
            AND ANY SourceInstance.OperationalStatus[*] = 17
            AND ANY SourceInstance.OperationalStatus[*] = 2 "
    @{Determine if Indications already exist or have to be
        created}&createIndication(#Filter1)

    // '17' ("Completed") '6' ("Error")
    #Filter2 = "SELECT * FROM CIM_InstModification
        WHERE SourceInstance ISA CIM_ConcreteJob
            AND ANY SourceInstance.OperationalStatus[*] = 17
            AND ANY SourceInstance.OperationalStatus[*] = 6 "
    @{Determine if Indications already exist or have to be
        created}&createIndication(#Filter2)
}

// Step 6. Create the Storage Pool
if(#SupportsPoolCreation)
{
    %InArguments["ElementName"] = NULL// we do not care what
        // the name is
    %InArguments["Goal"] = $GeneratedStorageSetting->
    %InArguments["Size"] = #PossibleSize
    %InArguments["InExtents"] = null
    %InArguments["Pool"] = null
}
```

```

%InArguments["InPools"] = $PoolToDrawFrom->
#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>

    <Wait for indication from either filters defined in step 5
        If the indication states the Job is 'Complete' and 'Error'
        then exit with error
        ERROR! Job did not complete successfully
    >
}
$CapabilitiesOffered[] = Associators(
    $PoolToDrawFrom->,
    "CIM_ElementCapabilities",
    "CIM_StorageCapabilities",
    null,
    null,
    false,
    false,
    null).
$StorageCapabilitiesOffered = $CapabilitiesOffered[0]
}

// Step 7. Create Storage Element.
%InArguments["SettingType"] = 3 // "Goal"
#ReturnValue = InvokeMethod(
    $StorageCapabilitiesOffered.getObjectPath(),
    "CreateSetting",
    %InArguments,
    %OutArguments)
if (#ReturnValue != 0)
{
    <ERROR! Unable to create storage setting >
}
$GeneratedStorageSetting-> = %OutArguments["NewSetting"]

%InArguments["ElementName"] = NULL
%InArguments["ElementType"] = #ElementType
%InArguments["Goal"] = $GeneratedStorageSetting->

```

```
%InArguments["Size"] = #PossibleSize
$InArguments["InPool"] = $PoolToDrawFrom->
%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))
{
    $CreatedElement-> = %OutArguments["TheElement"]
}
else // a Job was created and TheElement is null
{
    <Wait for indication from either filters defined in step 5
    If the indication states the Job is 'Complete' and 'Error'
    then exit with error
    ERROR! Job did not complete successfully
    >

    <Once the 'Job' has completed successfully, see step 5, then
    follow the AffectedJobElement association from the 'Job' to
    retrieve the storage element that was created.>
    $CreateElements[] = Associators(
        $Job->, // Object Name coarsed from %OutArguments["Job"]
        "CIM_AffectedJobElement",
        #StorageElementClass,
        null,
        null,
        false,
        false,
        null)
    // Only one storage element will be created,
    $CreatedElement-> = $CreateElements[0].getObjectPath()
}
```

6.6.7 Conditional: Expand Storage Element on Block Server

```
// DESCRIPTION
// In this recipe, we attempt to expand a LUN on an array by 50%.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
// 1. The storage configuration service is supported indicating the
// storage configuration is permitted. This is the condition for the
// recipe.
// 2. A reference to the CIM_ComputerSystem that represents the array
// $BlockServer->
```

```

// 3.A reference to the particular storage element we wish to expand.
// $ElementToExpand->
// 4. It is assumed that to expand a storage element there needs to be
// enough space available in the parent StoragePool to contain
// another copy of the storage element 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.
// 5. #ElementClassName is set to the class name of the storage element
// be
modified.
// (e.g. CIM_StorageVolume or CIM_LogicalDisk)
// 6. #ElementType is set to the storage element to modified
// See CreateOrModifyElementFromStoragePool.ElementType

// Step 1. Get the configuration services and determine the service
// capabilities. Note that the device may not support storage
// configuration so it is possible that the service is not present and
// the desired management cannot be performed
try {
    $Services->[] = AssociatorNames($BlockServer->,
        "CIM_HostedService",
        "CIM_StorageConfigurationService",
        null,
        null)
    // StorageConfigurationService and HostedService may not be
    // implemented
    // in the SMI Agent.
    if ($Services->[] == null) {
        <ERROR! Storage Configuration is not supported.>
    }
} catch (CIMException $Exception) {
    // StorageConfigurationService and/or HostedService may not be
    // included in
    // the model implemented at all if Storage Configuration is not
    // supported.
    if ($Exception.CIMStatusCode == CIM_ERR_INVALID_PARAMETER) {
        <ERROR! Storage Configuration is not supported.>
    }
}

// There should be only one storage configuration service
// Associated with the system
$StorageConfigurationService-> = $Services->[0]
$ServiceCapabilities[] = Associators(
    $BlockServer->,
    "CIM_ElementCapabilities",
    "CIM_StorageConfigurationCapabilities",
    null,
    null,
    false,
    false,
    null)

```



```
// There should be only one StorageConfigurationCapabilities instance
#SupportsElementModification1 = contains(
    7, // Storage Element Modification
    $ServiceCapabilities[0].SupportedSynchronousActions[] ||
    contains(
        7, // Storage Element Modification
        $ServiceCapabilities[0].SupportedAsynchronousActions[]
    )
#SupportsElementModification2 = contains(
    5, // Storage Element Modification
    $ServiceCapabilities[0].SupportedStorageElementFeatures[]
#ElementModificationProducesJob = contains(
    7, // Storage Element Modification
    $ServiceCapabilities[0].SupportedAsynchronousActions[]
if(!#SupportedElementModification1 || !#SupportedElementModification2)
{
    <EXIT: The ability to modify an existing Storage Element must be
        supported
    to continue.>
}

// Step 2. Read the current size of the Storage Element.
$StorageElement = GetInstance(
    $ElementToExpand->,
    false,
    false,
    false,
    {"BlockSize", "NumberOfBlocks"})
#PreviousSize = $StorageElement.BlockSize *
    $StorageElement.NumberOfBlocks

// Step 3. Follow the AllocatedFromStoragePool association from the
// storage element to find the pool from whence it came.
$Pools->[] = AssociatorNames(
    $ElementToExpand->,
    "CIM_AllocatedFromStoragePool",
    "CIM_StoragePool",
    null,
    null)

// A Storage Element has only one Pool parent
$ParentPool-> = $Pools->[0]

// Step 4. Determine whether the desired space for which to expand the
// storage element exists within the pool.
$StorageSetting->[] = AssociatorNames(
    $ElementToExpand->,
    "CIM_ElementSettingData",
    "CIM_StorageSetting",
    null,
    null)
```

```

$CurrentElementSetting-> = $StorageSetting->[0]
// Calculate the additional space needed
#SizeToExpand = 0.5 * #PreviousSize
// Calculate 150% of previous storage element size
#SizeToExpandTo = #PreviousSize + (0.5 * #PreviousSize)
#NewSizeAvailable =
    @<Create Storage Pool and Storage Element on Block Server>
        &PoolSizeAvailable(
            $ParentPool->,
            $CurrentElementSetting->,
            #SizeToExpand,
            #ElementType)
if (0 == #NewSizeAvailable)
{
    <ERROR! Unable to proceed because the requested size is unavailable
    >
}

// Step 5. Register for indications on configuration jobs
If(#ElementModificationProducesJob)
{
    // '17' ("Completed") '2' ("OK")
    #Filter1 = "SELECT * FROM CIM_InstModification
        WHERE SourceInstance ISA CIM_ConcreteJob
            AND ANY SourceInstance.OperationalStatus[*] = 17
            AND ANY SourceInstance.OperationalStatus[*] = 2 "
    @{Determine if Indications already exist or have to be
        created}&createIndication(#Filter1)

    // '17' ("Completed") '6' ("Error")
    #Filter2 = "SELECT * FROM CIM_InstModification
        WHERE SourceInstance ISA CIM_ConcreteJob
            AND ANY SourceInstance.OperationalStatus[*] = 17
            AND ANY SourceInstance.OperationalStatus[*] = 6 "
    @{Determine if Indications already exist or have to be
        created}&createIndication(#Filter2)
}

// Step 6. Modify the Storage Element
// If there is a Job produced, wait for Job completion
%InArguments["ElementName"] = null// we do not care what the name is
%InArguments["ElementType"] = #ElementType
%InArguments["Goal"] = $CurrentElementSetting
%InArguments["Size"] = #SizeToExpandTo
%InArguments["InPool"] = $ParentPool->
%InArguments["TheElement"] = $ElementToExpand->
#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)
{
    $CreatedElement-> = %OutArguments["TheElement"]
}
else // a Job was created and TheElement is null
{
    <Wait for indication from either filters defined in step 5
    If the indication states the Job is 'Complete' and 'Error'
    then exit with error
    ERROR! Job did not complete successfully
    >

    <Once the 'Job' has stopped, see step 4, then follow the
    AffectedJobElement association from the 'Job' to retrieve
    the storage element that was created.>
    $CreateElements[] = Associators(
        $Job->, // Object Name coerced from %OutArguments["Job"]
        "CIM_AffectedJobElement",
        #ElementClassName,
        null,
        null,
        false,
        false,
        null)
    // Only one Storage Element will be created,
    $CreatedElement-> = $CreatedElement[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 storage element was successfully expanded >
}
else
{
    if (#SizeExpandedTo <= #PreviousSize)
    {
        < indicate the storage element was not expanded >
    }
    else
    {
        < indicate the storage element was only partially expanded to
        #SizeExpandedTo >
    }
}
```

```
}
```

6.6.8 Conditional: Create Storage Element from Elements on Block Server

```
// DESCRIPTION
//
// This recipe demonstrates a use of "CreateOrModifyElementFromElements";
// However the recipe is known to fail when an implementation also
// implements the PoolsFromVolumes component
// profile.
//
// The goal of this recipe is to create a storage element with the
// maximum
// capabilities of the block server. If supported, the pool creation
// specifies
// the disk(s) to use as input rather than the size.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
// 1. The storage configuration service is supported indicating the
// storage configuration is permitted. This is the condition for the
// recipe.
// 2. A reference to a CIM_ComputerSystem Host is previously
// defined in the $Host-> variable
// 3. The references for input disks that are to be used for creating the
// pool
// are in $DisksForPool->[] array. All these must be associated to the
// primordial pool with CIM_ConcreteComponent association.
// On being transferred to a Concrete pool they will be disassociated
// from
// the primordial pool.
// 4. The storage element will be created using available disks in the
// concrete returned by GetAvailableExtents.
// 5. The settings for the new Storage Pool and Logical Disk are defined
// in
// the following variables:
// #RequestedSize = 10 * 1024 * 1024 * 1024 // 10 GB
// 6. #StorageElementClass is set to the class name of the element being
// created like CIM_StorageVolume or CIM_LogicalDisk.
// 7. #ElementType is set to the element to created
//      2 - StorageVolume
//      4 - LogicalDisk
// See CreateOrModifyElementFromStoragePool.ElementType

// MAIN
// Step 1. Get the configuration services and determine the service
// capabilities. Note that the device may not support storage
// configuration so it is possible that the service is not present and
// the desired management cannot be performed.
try {
    $Services->[] = AssociatorNames($Host->,
        "CIM_HostedService",
        "CIM_StorageConfigurationService",
        null,
```

```
        null)
    // StorageConfigurationService and HostedService may not be
        implemented
    // in the SMI Agent.
    if ($Services->[] == null) {
        <ERROR! Storage Configuration is not supported.>
    }
} catch (CIMException $Exception) {
    // StorageConfigurationService and/or HostedService may not be
        included in
    // the model implemented at all if Storage Configuration is not
        supported.
    if ($Exception.CIMStatusCode == CIM_ERR_INVALID_PARAMETER) {
        <ERROR! Storage Configuration is not supported.>
    }
}

// There should be only one storage configuration service
// Associated with the system
$StorageConfigurationService-> = $Services->[0]
$ServiceCapabilities[] = Associators($StorageConfigurationService->,
    "CIM_ElementCapabilities",
    "CIM_StorageConfigurationCapabilities",
    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].SupportedAsynchronousActions[])
#SupportsElementCreation1 = contains(12, // Storage Element from Element
    Creation
    $ServiceCapabilities[0].SupportedSynchronousActions[])
#SupportsElementCreation2 = contains(3, // LogicalDiskCreation
    $ServiceCapabilities[0].SupportedStorageElementFeatures[])
#ElementCreationProducesJob = contains(12, // Storage Element from
    Element Creation
    $ServiceCapabilities[0].SupportedAsynchronousActions[])
#SupportsInExtents = contains(2, // InExtents
    $ServiceCapabilities[0].SupportedStoragePoolFeatures[])

// If StorageExtent creation is not supported, the set of specific disks
    from
// which to allocate the StoragePool is not supported by the device.
if (!#SupportsInExtents) {
```

```

        <EXIT: The StoragePool cannot be created from a specific set of
            disks.>
    }

    // If a storage element can not be created and that storage element is
    // neither created synchronously or asynchronously, then fail the test
    if (!#SupportedElementCreation2 &&
        !(#SupportedElementCreation1 || #ElementCreationProducesJob)) {
        <EXIT: The StoragePool can be created, but the
            storage element from element creation is not supported.>
    }

    // Step 2. Enumerate over the CIM_HostedStoragePool associations to find
    // all the StoragePools from which storage elements might be created.
    $StoragePools[] = Associators($Host->,
        "CIM_HostedStoragePool",
        "CIM_StoragePool",
        null,
        null,
        false,
        false,
        {"InstanceID", "Primordial"})

    // Step 3. For each StoragePool, follow the CIM_ElementCapabilities
    // association to the StorageCapabilities of that pool. Compare the
    // StorageCapabilities to the desired StorageSetting and find the
    // best match.
    $PoolToDrawFrom-> = null
    for (#i in $StoragePools[]) {
        // If we can not create Storage Pool, then find a 'concrete'
        // Storage Pool from which to create a Storage Element
        #UsePrimordial = false
        if (#SupportsPoolCreation) {
            #UsePrimordial = true
            #RequestedElementType = 2 // StoragePool
        } else {
            #RequestedElementType = #ElementType
        }
        if ($StoragePools[#i].Primordial == #UsePrimordial) {
            $CapabilitiesOffered[] = Associators(
                $StoragePools[#i].getObjectPath(),
                "CIM_ElementCapabilities",
                "CIM_StorageCapabilities",
                null,
                null,
                false,
                false,
                null)
            $StorageCapabilitiesOffered =
                &GetMostCapable($CapabilitiesOffered[])
            $PoolToDrawFrom-> = $StoragePool[#i].getObjectPath()
        }
    }

```

```
// Step 4. Determine if the selected pool has enough space for
// another pool. If the block server supports hints, then
// the StorageSetting returned will contain default hints
// Create a setting
%InArguments["SettingType"] = 3 // Goal
#ReturnValue = InvokeMethod(
    $StorageCapabilitiesOffered.GetObjectPath(),
    "CreateSetting",
    %InArguments,
    %OutArguments)
if (#ReturnValue != 0 || null) {
    <ERROR! Unable to create storage setting >
}
$GeneratedStorageSetting-> = %OutArguments["NewSetting"]

// Determine the possible size, closest to the requested size
#PossibleSize = &PoolSizeAvailable(
    $PoolToDrawFrom->,
    $GeneratedStorageSetting->,
    #RequestedSize,
    #RequestedElementType)
if (0 != #PossibleSize) {
    // Located a size close to #RequestedSize
    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
    storage element>
}

// Step 5. Register for indications on configuration jobs
if (#PoolCreationProducesJob || #ElementCreationProducesJob) {
    // '17' ("Completed") '2' ("OK")
    #Filter1 = "SELECT * FROM CIM_InstModification
        WHERE SourceInstance ISA CIM_ConcreteJob
        AND ANY SourceInstance.OperationalStatus[*] = 17
        AND ANY SourceInstance.OperationalStatus[*] = 2 "
    @{Determine if Indications already exist or have to be
        created}&createIndication(#Filter1)

    // '17' ("Completed") '6' ("Error")
    #Filter2 = "SELECT * FROM CIM_InstModification
        WHERE SourceInstance ISA CIM_ConcreteJob
        AND ANY SourceInstance.OperationalStatus[*] = 17
        AND ANY SourceInstance.OperationalStatus[*] = 6 "
```



```

        @{Determine if Indications already exist or have to be
            created}&createIndication(#Filter2)
    }

// Step 6. Create the Storage Pool
if (#SupportsPoolCreation) {
    %InArguments["ElementName"] = NULL// leave up to the device
    %InArguments["Goal"] = $GeneratedStorageSetting->
    %InArguments["Size"] = null
    %InArguments["InExtents"] = $DisksForPool->[]
    %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>

        <Wait for indication from either filters defined in step 5
            If the indication states the Job is 'Complete' and 'Error'
            then exit with error
            ERROR! Job did not complete successfully>
    }
    $CapabilitiesOffered[] = Associators($PoolToDrawFrom->,
        "CIM_ElementCapabilities",
        "CIM_StorageCapabilities",
        null,
        null,
        false,
        false,
        null)
    $StorageCapabilitiesOffered = $CapabilitiesOffered[0]
}

// Step 7. Call GetAvailableExtents to find available extents for
            creating
// the storage element.
%InArguments["Goal"] = $GeneratedStorageSetting->
#ReturnValue = InvokeMethod($PoolToDrawFrom->,
    "GetAvailableExtents",
    %InArguments, %OutArguments)
if (#ReturnValue != 1) {

```

```
// Not supported
<EXIT! Method not supported, can not finish this recipe>
} else if (#ReturnValue != 0) {
    // Method did not succeeded or succeeded but did not create a job
    <ERROR! Failed>
}
$DisksForElement->[] = %OutArguments["AvailableExtents"]

// Step 8. Create Storage Element
%InArguments["SettingType"] = 3 // "Goal"
InvokeMethod($StorageCapabilitiesOffered.getObjectPath(),
    "CreateSetting",
    %InArguments,
    %OutArguments)
if (#ReturnValue != 0) {
    <ERROR! Unable to create storage setting >
}
$GeneratedStorageSetting-> = %OutArguments["NewSetting"]

%InArguments["ElementName"] = NULL
%InArguments["ElementType"] = #ElementType
%InArguments["Goal"] = $GeneratedStorageSetting->
%InArguments["Size"] = #PossibleSize
$InPools->[0] = $PoolToDrawFrom->
%InArguments["InPool"] = $InPools->
%InArguments["InElements"] = $DisksForElement->[]
%InArguments["TheElement"] = null // Create new element
#ReturnValue = InvokeMethod($StorageConfigurationService->,
    "CreateOrModifyElementFromElements",
    %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))) {
    $CreatedElement-> = %OutArguments["TheElement"]
} else // a Job was created and TheElement is null {
    <Wait for indication from either filters defined in step 5
    If the indication states the Job is 'Complete' and 'Error'
    then exit with error
    ERROR! Job did not complete successfully>

    <Once the 'Job' has completed, see step 5, then follow the
    AffectedJobElement association from the 'Job' to retrieve
    the storage element that was created.>
    $CreateElements[] = Associators(
        $Job->, // Object Name coersed from %OutArguments["Job"]
        "CIM_AffectedJobElement",
        #StorageElementClass,
        null,
        null,
```

```

        false,
        false,
        null)
    // Only one LogicalDisk will be created,
    $CreatedElement-> = $CreatedElements[0].getObjectPath()
}

```

6.6.9 Optional: Intentionally General a CIM Error

```

// DESCRIPTION
// Validate reporting an error/exception
// when InvokeMethod is called with an invalid parameter.
//
// This recipe intentionally supplies an invalid "ElementType".
//
// This recipe attempts to optionally utilize properties of CIM_Error
// if CIM_Error is implemented.

// 1. Insert an error
// 2. Catch the exception
// 3. Report the error

// PRE-EXISTING CONDITIONS AND ASSUMPTION
// 1.A reference to a storage setting is previously defined
//     in the $StorageSetting-> variable.
// 2.A size that is possible for the creation of a storage element
//     is provided in the #PossibleSize,
// 3.A reference to Pool is previous defined in the $PoolToDrawFrom->
//     variable
// 4.A object paths for source input Pools is previous defined in the
//     $InPools variable
// 5. A reference to the StorageConfigurationService is already defined
//     in the StorageConfiguratonservice-> variable
//
%InArguments["ElementType"] = 1000 // Invalid ElementType
%InArguments["Goal"] = $StorageSetting->
%InArguments["Size"] = #PossibleSize
%InPools->[0] = $PoolToDrawFrom->
%InArguments["InPool"] = $InPools->
%InArguments["TheElement"] = null
try
{
    #ReturnValue = InvokeMethod(
        $StorageConfigurationService->,
        "CreateOrModifyElementFromStoragePool",
        %InArguments, %OutArguments)
}
catch (CIM Exception $Exception) {
    // For SMI-S 1.1, optionally allow for implementation of CIM_Error.
    if($Exception.MessageID <> null) { // CIM_Error is implemented
        // For example

```

```

if($Exception.MessageArguments[2] ==
    "CreateOrModifyElementFromStoragePool") &&
    $Exception.MessageArguments[0] == "1" && // Second method
        parameter
    $Exception.MessageID = "MP5")
{
    <EXIT: Success -- CIM_Error is constructed properly>
}
else {
    <ERROR! Improperly constructed CIM_Error>
}
}
else {
    <display, optional CIM_Error is not implemented>
    if($Exception.CIMStatusCode != CIM_ERR_INVALID_PARAMETER) {
        <ERROR! Improper CIM status code returned>
    }
    else {
        <EXIT: Success -- correct CIM status code reported>
    }
}
}

if (#ReturnValue != CIM_ERR_INVALID_PARAMETER) { // 5 = Invalid parameter
    <ERROR! Invalid return value >
}

```

6.7 Registered Name and Version

Block Services version 1.5.0 (Component Profile).

6.8 CIM Elements

6.8.1 General

Table 24 describes the CIM elements for Block Services.

Table 24 - CIM Elements for Block Services

Element Name	Requirement	Description
6.8.2 CIM_AllocatedFromStoragePool (Pool from Pool)	Mandatory	AllocatedFromStoragePool.
6.8.3 CIM_AllocatedFromStoragePool (Volume or LogicalDisk from Pool)	Conditional	Conditional requirement: Referenced from Array - StorageVolume is mandatory or Referenced from Storage Virtualizer - StorageVolume is mandatory or Referenced from Host Hardware RAID Controller - StorageVolume is mandatory. AllocatedFromStoragePool.
6.8.4 CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StorageVolume or LogicalDisk)	Optional	Expressed the ability for the element to be named or have its state changed.

Table 24 - CIM Elements for Block Services (continued)

Element Name	Requirement	Description
6.8.5 CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StoragePool)	Optional	Expressed the ability for the element to be named or have its state changed.
6.8.6 CIM_ElementCapabilities (StorageCapabilities to StorageConfigurationService)	Optional	Associates StorageCapabilities with StorageConfigurationService. This StorageCapabilities shall represent the capabilities of the entire implementation.
6.8.7 CIM_ElementCapabilities (StorageCapabilities to StoragePool)	Mandatory	Associates StorageCapabilities with StoragePool. This StorageCapabilities shall represent the capabilities of the StoragePool to which it is associated.
6.8.8 CIM_ElementCapabilities (StorageConfigurationCapabilities to StorageConfigurationService)	Mandatory	Associates StorageConfigurationCapabilities with StorageConfigurationService.
6.8.9 CIM_ElementCapabilities (StorageConfigurationCapabilities to concrete StoragePool)	Optional	Associates StorageConfigurationCapabilities with StoragePool.
6.8.10 CIM_ElementCapabilities (StorageConfigurationCapabilities to primordial StoragePool)	Optional	Associates StorageConfigurationCapabilities with StoragePool.
6.8.11 CIM_ElementCapabilities (Used to declare the naming capabilities of the StoragePool)	Optional	Associates EnabledLogicalElementCapabilities with StorageConfigurationService.
6.8.12 CIM_ElementCapabilities (Used to declare the naming capabilities of the StorageVolume or LogicalDisk)	Optional	Associates EnabledLogicalElementCapabilities with StorageConfigurationService.
6.8.13 CIM_ElementSettingData	Mandatory	
6.8.14 CIM_EnabledLogicalElementCapabilities (For StorageConfigurationService)	Optional	This class is used to express the naming and possible requested state change possibilities for storage elements.
6.8.15 CIM_EnabledLogicalElementCapabilities (For StoragePool)	Optional	This class is used to express the naming and possible requested state change possibilities for storage pools.
6.8.16 CIM_FilterCollection (Block Services Predefined FilterCollection)	Conditional	Experimental. Conditional requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.Supporte dFeatures='5' (Predefined Filter Collections). This is a collection of predefined IndicationFilters to which a client may subscribe.
6.8.17 CIM_HostedCollection (System to predefined IndicationFilters)	Conditional	Experimental. Conditional requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.Supporte dFeatures='5' (Predefined Filter Collections).
6.8.18 CIM_HostedService	Conditional	Conditional requirement: Support for StorageConfigurationService.
6.8.19 CIM_HostedStoragePool	Mandatory	
6.8.20 CIM_IndicationFilter (Logical Disk Creation)	Conditional	Experimental. Conditional requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.Supporte dFeatures='3' (Predefined Filters). This is the 'pre-defined' CIM_IndicationFilter instance for the addition of a new LogicalDisk instance.

Table 24 - CIM Elements for Block Services (continued)

Element Name	Requirement	Description
6.8.21 CIM_IndicationFilter (Logical Disk Deletion)	Conditional	Experimental. Conditional requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='3' (Predefined Filters). This is the 'pre-defined' CIM_IndicationFilter instance for the deletion of a LogicalDisk instance.
6.8.22 CIM_IndicationFilter (Logical Disk OperationalStatus)	Conditional	Experimental. Conditional requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='3' (Predefined Filters). This is the 'pre-defined' CIM_IndicationFilter instance for changes in the OperationalStatus of LogicalDisk instances.
6.8.23 CIM_IndicationFilter (Storage Pool Creation)	Conditional	Experimental. Conditional requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='3' (Predefined Filters). This is the 'pre-defined' CIM_IndicationFilter instance for the addition of a new StoragePool instance.
6.8.24 CIM_IndicationFilter (Storage Pool Deletion)	Conditional	Experimental. Conditional requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='3' (Predefined Filters). This is the 'pre-defined' CIM_IndicationFilter instance for the deletion of a StoragePool instance.
6.8.25 CIM_IndicationFilter (Storage Pool TotalManagedSpace)	Conditional	Experimental. Conditional requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='3' (Predefined Filters). This is the 'pre-defined' CIM_IndicationFilter instance for changes in TotalManagedSpace for StoragePool instances.
6.8.26 CIM_IndicationFilter (Storage Volume Creation)	Conditional	Experimental. Conditional requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='3' (Predefined Filters). This is the 'pre-defined' CIM_IndicationFilter instance for the addition of a new StorageVolume instance.
6.8.27 CIM_IndicationFilter (Storage Volume Deletion)	Conditional	Experimental. Conditional requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='3' (Predefined Filters). This is the 'pre-defined' CIM_IndicationFilter instance for the deletion of a StorageVolume instance.
6.8.28 CIM_IndicationFilter (Storage Volume OperationalStatus)	Conditional	Experimental. Conditional requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='3' (Predefined Filters). This is the 'pre-defined' CIM_IndicationFilter instance for changes in the OperationalStatus of StorageVolume instances.

Table 24 - CIM Elements for Block Services (continued)

Element Name	Requirement	Description
6.8.29 CIM_IndicationFilter (WQL Logical Disk OperationalStatus)	Conditional	Deprecated. Conditional requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='3' (Predefined Filters). This is the 'pre-defined' WQL version of the CIM_IndicationFilter instance for changes in the OperationalStatus of LogicalDisk instances.
6.8.30 CIM_IndicationFilter (WQL Storage Volume OperationalStatus)	Conditional	Deprecated. Conditional requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='3' (Predefined Filters). This is the 'pre-defined' WQL version of the CIM_IndicationFilter instance for changes in the OperationalStatus of StorageVolume instances.
6.8.31 CIM_LogicalDisk	Conditional	Conditional requirement: Referenced from Volume Management - LogicalDisk is mandatory. A LogicalDisk is allocated from a concrete StoragePool.
6.8.32 CIM_MemberOfCollection (Block Services Filter Collection to FilterCollection)	Conditional	Experimental. Conditional requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='5' (Predefined Filter Collections). This associates the Block Services predefined FilterCollection to the FilterCollection for the autonomous profile (e.g., the Array FilterCollection).
6.8.33 CIM_MemberOfCollection (Predefined Filter Collection to Block Services Filters)	Conditional	Experimental. Conditional requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='5' (Predefined Filter Collections). This associates the Block Services predefined FilterCollection to the predefined Filters supported by the implementation.
6.8.34 CIM_OwningJobElement	Conditional	Conditional requirement: Support for Job Control profile.
6.8.35 CIM_StorageCapabilities	Mandatory	
6.8.36 CIM_StorageConfigurationCapabilities (Concrete)	Optional	
6.8.37 CIM_StorageConfigurationCapabilities (Global)	Conditional	Conditional requirement: Support for StorageConfigurationService.
6.8.38 CIM_StorageConfigurationCapabilities (Primordial)	Optional	
6.8.39 CIM_StorageConfigurationService	Optional	
6.8.40 CIM_StoragePool (Concrete)	Mandatory	The concrete StoragePool. A concrete StoragePool shall be allocated from another StoragePool. It shall be used for allocating StorageVolumes and LogicalDisks as well as other concrete StoragePools.
6.8.41 CIM_StoragePool (Empty)	Optional	An empty StoragePool is a special case of a StoragePool (Concrete or Primordial) where the StoragePool contains no capacity.

Table 24 - CIM Elements for Block Services (continued)

Element Name	Requirement	Description
6.8.42 CIM_StoragePool (Primordial)	Mandatory	The primordial StoragePool. It is created by the provider and cannot be deleted or modified. It cannot be used to allocate any storage element other than concrete StoragePools.
6.8.43 CIM_StorageSetting	Mandatory	
6.8.44 CIM_StorageSettingWithHints	Optional	
6.8.45 CIM_StorageSettingsAssociatedToCapabilities	Optional	This class associates the StorageCapabilities with the preset setting. Any StorageSetting instance associated with this association shall work, unmodified, to create a storage element. The preset settings should not change overtime and represent possible settings for storage elements are set of design time rather than runtime. All StorageSetting instances linked with this association shall have a ChangeableType of "0" ("Fixed - Not Changeable").
6.8.46 CIM_StorageSettingsGeneratedFromCapabilities	Conditional	Conditional requirement: Support for StorageConfigurationService. This class associates the StorageCapabilities with the StorageSetting generated from it via the CreateSetting method. StorageSettings instances generated in this manner, as identified with this association, may be removed from the model at any time by the implementation if the ChangeableType of the associated setting is set to "2" ("Changeable - Transient"). All StorageSettings associated with this class shall be changeable, ChangeableType is "2" or "3". Some implementations may permit the modification of the ChangeableType property itself on StorageSetting instances associated via this class. Provided this is allowed, a client may change the ChangeableType to "3" ("Changeable - Persistent") to have this setting retained either after generation of the instance or after its modification by the client. The DefaultSetting property of the StorageSetting instances linked with this association is meaningless.
6.8.47 CIM_StorageVolume	Conditional	Conditional requirement: Referenced from Array - StorageVolume is mandatory or Referenced from Storage Virtualizer - StorageVolume is mandatory or Referenced from Host Hardware RAID Controller - StorageVolume is mandatory. Representation of a virtual disk (for SCSI, a logical unit). A StorageVolume is allocated from a concrete StoragePool. See the "Standard Formats for Logical Unit Names" section in Information Technology – Storage Management, Part 1 Common Architecture for details on how to set Name, NameFormat, and NameNamespace properties.
6.8.48 CIM_SystemDevice (System to StorageVolume or LogicalDisk)	Mandatory	Associates top level system from Array, Virtualizer, ... to StorageVolume or LogicalDisk.
6.8.49 SNIA_StorageVolume	Optional	An optional extension of CIM_StorageVolume.
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_StoragePool	Mandatory	Creation/Deletion of StoragePool. See 6.8.23 CIM_IndicationFilter (Storage Pool Creation).
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_StoragePool	Mandatory	Deletion of StoragePool. See 6.8.24 CIM_IndicationFilter (Storage Pool Deletion).

Table 24 - CIM Elements for Block Services (continued)

Element Name	Requirement	Description
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_StorageVolume	Conditional	Conditional requirement: Referenced from Array - StorageVolume is mandatory or Referenced from Storage Virtualizer - StorageVolume is mandatory or Referenced from Host Hardware RAID Controller - StorageVolume is mandatory. Creation of StorageVolume, if the StorageVolume storage element is implemented. See 6.8.26 CIM_IndicationFilter (Storage Volume Creation).
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_StorageVolume	Conditional	Conditional requirement: Referenced from Array - StorageVolume is mandatory or Referenced from Storage Virtualizer - StorageVolume is mandatory or Referenced from Host Hardware RAID Controller - StorageVolume is mandatory. Deletion of StorageVolume, if the StorageVolume storage element is implemented. See 6.8.27 CIM_IndicationFilter (Storage Volume Deletion).
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_StorageVolume AND SourceInstance.OperationalStatus <> PreviousInstance.OperationalStatus	Conditional	Conditional requirement: Referenced from Array - StorageVolume is mandatory or Referenced from Storage Virtualizer - StorageVolume is mandatory or Referenced from Host Hardware RAID Controller - StorageVolume is mandatory. Deprecated WQL - Change of status of a Storage Volume, if Storage Volume is implemented. See 6.8.30 CIM_IndicationFilter (WQL Storage Volume OperationalStatus).
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_StorageVolume AND SourceInstance.CIM_StorageVolume::OperationalS tatus <> PreviousInstance.CIM_StorageVolume::Operational Status	Conditional	Conditional requirement: Referenced from Array - StorageVolume is mandatory or Referenced from Storage Virtualizer - StorageVolume is mandatory or Referenced from Host Hardware RAID Controller - StorageVolume is mandatory. CQL -Change of status of a Storage Volume, if Storage Volume is implemented. See 6.8.28 CIM_IndicationFilter (Storage Volume OperationalStatus).
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_LogicalDisk	Conditional	Conditional requirement: Referenced from Volume Management - LogicalDisk is mandatory. Creation of LogicalDisk, if the LogicalDisk storage element is implemented. See 6.8.20 CIM_IndicationFilter (Logical Disk Creation).
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_LogicalDisk	Conditional	Conditional requirement: Referenced from Volume Management - LogicalDisk is mandatory. Deletion of LogicalDisk, if the LogicalDisk storage element is implemented. See 6.8.21 CIM_IndicationFilter (Logical Disk Deletion).

Table 24 - CIM Elements for Block Services (continued)

Element Name	Requirement	Description
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_LogicalDisk AND SourceInstance.OperationalStatus <> PreviousInstance.OperationalStatus	Conditional	Conditional requirement: Referenced from Volume Management - LogicalDisk is mandatory. Deprecated WQL -Change of status of LogicalDisk, if LogicalDisk is implemented. See 6.8.29 CIM_IndicationFilter (WQL Logical Disk OperationalStatus).
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_LogicalDisk AND SourceInstance.CIM_LogicalDisk::OperationalStatus <> PreviousInstance.CIM_LogicalDisk::OperationalStatus	Conditional	Conditional requirement: Referenced from Volume Management - LogicalDisk is mandatory. CQL - Change of status of LogicalDisk, if LogicalDisk is implemented. See 6.8.22 CIM_IndicationFilter (Logical Disk OperationalStatus).
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_StoragePool AND SourceInstance.CIM_StoragePool::TotalManagedSpace <> PreviousInstance.CIM_StoragePool::TotalManagedSpace	Mandatory	CQL -Change of TotalManagedSpace. See 6.8.25 CIM_IndicationFilter (Storage Pool TotalManagedSpace).

6.8.2 CIM_AllocatedFromStoragePool (Pool from Pool)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 25 describes class CIM_AllocatedFromStoragePool (Pool from Pool).

Table 25 - SMI Referenced Properties/Methods for CIM_AllocatedFromStoragePool (Pool from Pool)

Properties	Flags	Requirement	Description and Notes
SpaceConsumed		Mandatory	
Antecedent		Mandatory	Antecedent references the parent pool from which the dependent pool is allocated.
Dependent		Mandatory	

6.8.3 CIM_AllocatedFromStoragePool (Volume or LogicalDisk from Pool)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Referenced from Array - StorageVolume is mandatory or Referenced from Storage Virtualizer - StorageVolume is mandatory or Referenced from Host Hardware RAID Controller - StorageVolume is mandatory.

Table 26 describes class CIM_AllocatedFromStoragePool (Volume or LogicalDisk from Pool).

**Table 26 - SMI Referenced Properties/Methods for CIM_AllocatedFromStoragePool
(Volume or LogicalDisk from Pool)**

Properties	Flags	Requirement	Description and Notes
SpaceConsumed		Mandatory	
Antecedent		Mandatory	
Dependent		Mandatory	

6.8.4 CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StorageVolume or LogicalDisk)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 27 describes class CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StorageVolume or LogicalDisk).

**Table 27 - SMI Referenced Properties/Methods for CIM_ElementCapabilities
(EnabledLogicalElementCapabilities to StorageVolume or LogicalDisk)**

Properties	Flags	Requirement	Description and Notes
Capabilities		Mandatory	The capabilities object associated with the element.
ManagedElement		Mandatory	A Storage Volume or Logical Disk.

6.8.5 CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StoragePool)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 28 describes class CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StoragePool).

**Table 28 - SMI Referenced Properties/Methods for CIM_ElementCapabilities
(EnabledLogicalElementCapabilities to StoragePool)**

Properties	Flags	Requirement	Description and Notes
Capabilities		Mandatory	The capabilities object (CIM_EnabledLogicalElementCapabilities) with an ElementName of "StoragePool Enabled Capabilities" that is associated with a storage pool.
ManagedElement		Mandatory	A reference to an instance of a StoragePool.

6.8.6 CIM_ElementCapabilities (StorageCapabilities to StorageConfigurationService)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 29 describes class CIM_ElementCapabilities (StorageCapabilities to StorageConfigurationService).

Table 29 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (StorageCapabilities to StorageConfigurationService)

Properties	Flags	Requirement	Description and Notes
Capabilities		Mandatory	The capabilities object associated with the element.
ManagedElement		Mandatory	The managed element.

6.8.7 CIM_ElementCapabilities (StorageCapabilities to StoragePool)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 30 describes class CIM_ElementCapabilities (StorageCapabilities to StoragePool).

Table 30 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (StorageCapabilities to StoragePool)

Properties	Flags	Requirement	Description and Notes
Capabilities		Mandatory	The capabilities object associated with the element.
ManagedElement		Mandatory	The managed element.

6.8.8 CIM_ElementCapabilities (StorageConfigurationCapabilities to StorageConfigurationService)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 31 describes class CIM_ElementCapabilities (StorageConfigurationCapabilities to StorageConfigurationService).

Table 31 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (StorageConfigurationCapabilities to StorageConfigurationService)

Properties	Flags	Requirement	Description and Notes
Capabilities		Mandatory	The capabilities object associated with the element.
ManagedElement		Mandatory	The managed element.

6.8.9 CIM_ElementCapabilities (StorageConfigurationCapabilities to concrete Storage-Pool)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 32 describes class CIM_ElementCapabilities (StorageConfigurationCapabilities to concrete StoragePool).

**Table 32 - SMI Referenced Properties/Methods for CIM_ElementCapabilities
(StorageConfigurationCapabilities to concrete StoragePool)**

Properties	Flags	Requirement	Description and Notes
Capabilities		Mandatory	The capabilities object associated with the element.
ManagedElement		Mandatory	The managed element.

6.8.10 CIM_ElementCapabilities (StorageConfigurationCapabilities to primordial StoragePool)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 33 describes class CIM_ElementCapabilities (StorageConfigurationCapabilities to primordial StoragePool).

**Table 33 - SMI Referenced Properties/Methods for CIM_ElementCapabilities
(StorageConfigurationCapabilities to primordial StoragePool)**

Properties	Flags	Requirement	Description and Notes
Capabilities		Mandatory	The capabilities object associated with the element.
ManagedElement		Mandatory	The managed element.

6.8.11 CIM_ElementCapabilities (Used to declare the naming capabilities of the StoragePool)

Associates EnabledLogicalElementCapabilities with StorageConfigurationService. This is for identifying the capability to provide an element name for storage pools.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 34 describes class CIM_ElementCapabilities (Used to declare the naming capabilities of the StoragePool).

**Table 34 - SMI Referenced Properties/Methods for CIM_ElementCapabilities
(Used to declare the naming capabilities of the StoragePool)**

Properties	Flags	Requirement	Description and Notes
Capabilities		Mandatory	The capabilities object (CIM_EnabledLogicalElementCapabilities) with an ElementName of "StoragePool Enabled Capabilities" that is associated with an instance of StorageConfigurationService.
ManagedElement		Mandatory	A reference to an instance of CIM_StorageConfigurationService.

6.8.12 CIM_ElementCapabilities (Used to declare the naming capabilities of the StorageVolume or LogicalDisk)

Associates EnabledLogicalElementCapabilities with StorageConfigurationService. This is for identifying the capability to provide an element name for storage volumes or logical disks.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 35 describes class CIM_ElementCapabilities (Used to declare the naming capabilities of the StorageVolume or LogicalDisk).

**Table 35 - SMI Referenced Properties/Methods for CIM_ElementCapabilities
(Used to declare the naming capabilities of the StorageVolume or LogicalDisk)**

Properties	Flags	Requirement	Description and Notes
Capabilities		Mandatory	The capabilities object (CIM_EnabledLogicalElementCapabilities) with an ElementName of "StorageVolume Enabled Capabilities" or "LogicalDisk Enabled Capabilities" that is associated with an instance of StorageConfigurationService.
ManagedElement		Mandatory	A reference to an instance of CIM_StorageConfigurationService.

6.8.13 CIM_ElementSettingData

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 36 describes class CIM_ElementSettingData.

Table 36 - SMI Referenced Properties/Methods for CIM_ElementSettingData

Properties	Flags	Requirement	Description and Notes
IsDefault		Mandatory	An enumerated integer indicating that the referenced setting is a default setting for the element, or that this information is unknown. Value shall be 0,1 or 2 (Unknown or Is Default or Is Not Default).
IsCurrent		Mandatory	An enumerated integer indicating that the referenced setting is currently being used in the operation of the element, or that this information is unknown. Value shall be 0,1 or 2 (Unknown or Is Default or Is Not Default).
ManagedElement		Mandatory	StorageVolume or LogicalDisk.
SettingData		Mandatory	The StorageSetting or StorageSettingWithHints that is associated with the Storage Volume or Logical Disk.

6.8.14 CIM_EnabledLogicalElementCapabilities (For StorageConfigurationService)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 37 describes class CIM_EnabledLogicalElementCapabilities (For StorageConfigurationService).

Table 37 - SMI Referenced Properties/Methods for CIM_EnabledLogicalElementCapabilities (For StorageConfigurationService)

Properties	Flags	Requirement	Description and Notes
InstanceID		Mandatory	
ElementName		Mandatory	For this usage of the capabilities this should include one of the following three values: StoragePool Enabled Capabilities StorageVolume Enabled Capabilities LogicalDisk Enabled Capabilities.
ElementNameEditSupported		Mandatory	Denotes whether a storage element can be named.
MaxElementNameLen		Mandatory	Specifies the maximum length in glyphs (letters) for the name. See MOF for details.
ElementNameMask		Mandatory	The regular expression that specifies the possible content and format for the element name. See MOF for details.
RequestedStatesSupported		Optional	Expresses the states to which this element may be changed using the RequestStateChange method. If this property, it may be assumed that the state may not be changed.

6.8.15 CIM_EnabledLogicalElementCapabilities (For StoragePool)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 38 describes class CIM_EnabledLogicalElementCapabilities (For StoragePool).

Table 38 - SMI Referenced Properties/Methods for CIM_EnabledLogicalElementCapabilities (For StoragePool)

Properties	Flags	Requirement	Description and Notes
InstanceID		Mandatory	
ElementName		Mandatory	For this usage of the capabilities this should be 'StoragePool Enabled Capabilities'.
ElementNameEditSupported		Mandatory	Denotes whether a storage element can be named.

Table 38 - SMI Referenced Properties/Methods for CIM_EnabledLogicalElementCapabilities (For StoragePool)

Properties	Flags	Requirement	Description and Notes
MaxElementNameLen		Mandatory	Specifies the maximum length in glyphs (letters) for the name. See MOF for details.
ElementNameMask		Mandatory	The regular expression that specifies the possible content and format for the element name. See Annex A (MOF) for details.
RequestedStatesSupported		Optional	Expresses the states to which this element may be changed using the RequestStateChange method. If this property, it may be assumed that the state may not be changed.

6.8.16 CIM_FilterCollection (Block Services Predefined FilterCollection)

Experimental. This is a collection of predefined IndicationFilters to which a client may subscribe. A Block Services implementation shall indicate support for predefined FilterCollections by the SNIA_IndicationConfigurationCapabilities.FeaturesSupported = '5' (Predefined Filter Collections).

Requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='5' (Predefined Filter Collections).

Table 39 describes class CIM_FilterCollection (Block Services Predefined FilterCollection).

Table 39 - SMI Referenced Properties/Methods for CIM_FilterCollection (Block Services Predefined FilterCollection)

Properties	Flags	Requirement	Description and Notes
InstanceID		Mandatory	Shall specify the unique identifier for an instance of this class within the Implementation namespace.
CollectionName		Mandatory	The value of CollectionName shall be 'SNIA:Block Services'.

6.8.17 CIM_HostedCollection (System to predefined IndicationFilters)

Experimental.

Requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='5' (Predefined Filter Collections).

Table 40 describes class CIM_HostedCollection (System to predefined IndicationFilters).

Table 40 - SMI Referenced Properties/Methods for CIM_HostedCollection (System to predefined IndicationFilters)

Properties	Flags	Requirement	Description and Notes
Dependent		Mandatory	Reference to the predefined FilterCollection for Block Services.
Antecedent		Mandatory	Reference to the System of the referencing profile.

6.8.18 CIM_HostedService

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Support for StorageConfigurationService.

Table 41 describes class CIM_HostedService.

Table 41 - SMI Referenced Properties/Methods for CIM_HostedService

Properties	Flags	Requirement	Description and Notes
Antecedent		Mandatory	The hosting computer system.
Dependent		Mandatory	The storage configuration service hosted on the computer system.

6.8.19 CIM_HostedStoragePool

Requirement: Mandatory.

Table 42 describes class CIM_HostedStoragePool.

Table 42 - SMI Referenced Properties/Methods for CIM_HostedStoragePool

Properties	Flags	Requirement	Description and Notes
GroupComponent		Mandatory	The reference to the hosting computer system.
PartComponent		Mandatory	The reference to the hosted storage pool.

6.8.20 CIM_IndicationFilter (Logical Disk Creation)

Experimental. This is the 'pre-defined' CIM_IndicationFilter instance for the addition of a new LogicalDisk instance. This would typically occur as a result of an invocation of CreateOrModifyElementFromStoragePool method.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='3' (Predefined Filters).

Table 43 describes class CIM_IndicationFilter (Logical Disk Creation).

Table 43 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Logical Disk Creation)

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
CreationClassName		Mandatory	See the CreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
SystemName		Mandatory	See the SystemName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
Name		Mandatory	This shall be 'SNIA:Block Services:LogicalDiskCreation'.

Table 43 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Logical Disk Creation)

Properties	Flags	Requirement	Description and Notes
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
Query		Mandatory	SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_LogicalDisk.
QueryLanguage		Mandatory	This should be 'DMTF:CQL' for CQL queries, but may be 'WQL' or 'SMI-S V1.0'. WQL and SMI-S V1.0 are deprecated in favor of 'DMTF:CQL'.
ElementName	N	Optional	See the ElementName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).

6.8.21 CIM_IndicationFilter (Logical Disk Deletion)

Experimental. This is the 'pre-defined' CIM_IndicationFilter instance for the deletion of a LogicalDisk instance. This would typically occur as a result of an invocation of ReturnToStoragePool method.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='3' (Predefined Filters).

Table 44 describes class CIM_IndicationFilter (Logical Disk Deletion).

Table 44 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Logical Disk Deletion)

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
CreationClassName		Mandatory	See the CreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
SystemName		Mandatory	See the SystemName definition in section <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
Name		Mandatory	This shall be 'SNIA:Block Services:LogicalDiskDeletion'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in section <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in section <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).

Table 44 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Logical Disk Deletion)

Properties	Flags	Requirement	Description and Notes
Query		Mandatory	SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_LogicalDisk.
QueryLanguage		Mandatory	This should be 'DMTF:CQL' for CQL queries, but may be 'WQL' or 'SMI-S V1.0'. WQL and SMI-S V1.0 are deprecated in favor of 'DMTF:CQL'.
ElementName	N	Optional	See the ElementName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).

6.8.22 CIM_IndicationFilter (Logical Disk OperationalStatus)

Experimental. This is the 'pre-defined' CIM_IndicationFilter instance for changes in the OperationalStatus of LogicalDisk instances.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures=3 (Predefined Filters).

Table 45 describes class CIM_IndicationFilter (Logical Disk OperationalStatus).

Table 45 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Logical Disk OperationalStatus)

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
CreationClassName		Mandatory	See the CreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
SystemName		Mandatory	See the SystemName definition in section <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
Name		Mandatory	This shall be 'SNIA:Block Services:LogicalDiskOperationalStatus'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
Query		Mandatory	SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_LogicalDisk AND SourceInstance.CIM_LogicalDisk::OperationalStatus <> PreviousInstance.CIM_LogicalDisk::OperationalStatus.
QueryLanguage		Mandatory	This shall be 'DMTF:CQL'.
ElementName	N	Optional	See the ElementName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).

6.8.23 CIM_IndicationFilter (Storage Pool Creation)

Experimental. This is the 'pre-defined' CIM_IndicationFilter instance for the addition of a new StoragePool instance. This would typically occur as a result of an invocation of CreateOrModifyStoragePool method.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='3' (Predefined Filters).

Table 46 describes class CIM_IndicationFilter (Storage Pool Creation).

Table 46 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Storage Pool Creation)

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
CreationClassName		Mandatory	See the CreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
SystemName		Mandatory	See the SystemName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
Name		Mandatory	This shall be 'SNIA:Block Services:StoragePoolCreation'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
Query		Mandatory	SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_StoragePool.
QueryLanguage		Mandatory	This should be 'DMTF:CQL' for CQL queries, but may be 'WQL' or 'SMI-S V1.0'. WQL and SMI-S V1.0 are deprecated in favor of 'DMTF:CQL'.
ElementName	N	Optional	See the ElementName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).

6.8.24 CIM_IndicationFilter (Storage Pool Deletion)

Experimental. This is the 'pre-defined' CIM_IndicationFilter instance for the deletion of a StoragePool instance. This would typically occur as a result of an invocation of DeleteStoragePool method.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='3' (Predefined Filters).

Table 47 describes class CIM_IndicationFilter (Storage Pool Deletion).

Table 47 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Storage Pool Deletion)

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
CreationClassName		Mandatory	See the CreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
SystemName		Mandatory	See the SystemName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
Name		Mandatory	This shall be 'SNIA:Block Services:StoragePoolDeletion'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
Query		Mandatory	SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_StoragePool.
QueryLanguage		Mandatory	This should be 'DMTF:CQL' for CQL queries, but may be 'WQL' or 'SMI-S V1.0'. WQL and SMI-S V1.0 are deprecated in favor of 'DMTF:CQL'.
ElementName	N	Optional	See the ElementName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).

6.8.25 CIM_IndicationFilter (Storage Pool TotalManagedSpace)

Experimental. This is the 'pre-defined' CIM_IndicationFilter instance for changes in TotalManagedSpace for StoragePool instances. This would typically occur as a result of an invocation of CreateOrModifyStoragePool that expands a StoragePool.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='3' (Predefined Filters).

Table 48 describes class CIM_IndicationFilter (Storage Pool TotalManagedSpace).

**Table 48 - SMI Referenced Properties/Methods for CIM_IndicationFilter
(Storage Pool TotalManagedSpace)**

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
CreationClassName		Mandatory	See the CreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
SystemName		Mandatory	See the SystemName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
Name		Mandatory	This shall be 'SNIA:Block Services:StoragePoolTotalManagedSpace'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
Query		Mandatory	SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_StoragePool AND SourceInstance.CIM_StoragePool::TotalManagedSpace <> PreviousInstance.CIM_StoragePool::TotalManagedSpace.
QueryLanguage		Mandatory	This shall be 'DMTF:CQL'.
ElementName	N	Optional	See the ElementName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).

6.8.26 CIM_IndicationFilter (Storage Volume Creation)

Experimental. This is the 'pre-defined' CIM_IndicationFilter instance for the addition of a new StorageVolume instance. This would typically occur as a result of an invocation of CreateOrModifyElementFromStoragePool method.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='3' (Predefined Filters).

Table 49 describes class CIM_IndicationFilter (Storage Volume Creation).

**Table 49 - SMI Referenced Properties/Methods for CIM_IndicationFilter
(Storage Volume Creation)**

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
CreationClassName		Mandatory	See the CreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
SystemName		Mandatory	See the SystemName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
Name		Mandatory	This shall be 'SNIA:Block Services:StorageVolumeCreation'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
Query		Mandatory	SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_StorageVolume.
QueryLanguage		Mandatory	This should be 'DMTF:CQL' for CQL queries, but may be 'WQL' or 'SMI-S V1.0'. WQL and SMI-S V1.0 are deprecated in favor of 'DMTF:CQL'.
ElementName	N	Optional	See the ElementName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).

6.8.27 CIM_IndicationFilter (Storage Volume Deletion)

Experimental. This is the 'pre-defined' CIM_IndicationFilter instance for the deletion of a StorageVolume instance. This would typically occur as a result of an invocation of ReturnToStoragePool method.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='3' (Predefined Filters).

Table 50 describes class CIM_IndicationFilter (Storage Volume Deletion).

**Table 50 - SMI Referenced Properties/Methods for CIM_IndicationFilter
(Storage Volume Deletion)**

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
CreationClassName		Mandatory	See the CreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
SystemName		Mandatory	See the SystemName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
Name		Mandatory	This shall be 'SNIA:Block Services:StorageVolumeDeletion'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
Query		Mandatory	SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_StorageVolume.
QueryLanguage		Mandatory	This should be 'DMTF:CQL' for CQL queries, but may be 'WQL' or 'SMI-S V1.0'. WQL and SMI-S V1.0 are deprecated in favor of 'DMTF:CQL'.
ElementName	N	Optional	See the ElementName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).

6.8.28 CIM_IndicationFilter (Storage Volume OperationalStatus)

Experimental. This is the 'pre-defined' CIM_IndicationFilter instance for changes in the OperationalStatus of StorageVolume instances.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='3' (Predefined Filters).

Table 51 describes class CIM_IndicationFilter (Storage Volume OperationalStatus).

**Table 51 - SMI Referenced Properties/Methods for CIM_IndicationFilter
(Storage Volume OperationalStatus)**

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
CreationClassName		Mandatory	See the CreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
SystemName		Mandatory	See the SystemName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
Name		Mandatory	This shall be 'SNIA:Block Services:StorageVolumeOperationalStatus'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
Query		Mandatory	SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_StorageVolume AND SourceInstance.CIM_StorageVolume::OperationalStatus <> PreviousInstance.CIM_StorageVolume::OperationalStatus.
QueryLanguage		Mandatory	This shall be 'DMTF:CQL'.
ElementName	N	Optional	See the ElementName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).

6.8.29 CIM_IndicationFilter (WQL Logical Disk OperationalStatus)

Deprecated. This is the 'pre-defined' WQL version of the CIM_IndicationFilter instance for changes in the OperationalStatus of LogicalDisk instances.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='3' (Predefined Filters).

Table 52 describes class CIM_IndicationFilter (WQL Logical Disk OperationalStatus).

**Table 52 - SMI Referenced Properties/Methods for CIM_IndicationFilter
(WQL Logical Disk OperationalStatus)**

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
CreationClassName		Mandatory	See the CreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
SystemName		Mandatory	See the SystemName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
Name		Mandatory	This shall be 'SNIA:Block Services:LogicalDiskOperationalStatusWQL'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
Query		Mandatory	SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_LogicalDisk AND SourceInstance.OperationalStatus <> PreviousInstance.OperationalStatus.
QueryLanguage		Mandatory	This shall be 'WQL' or 'SMI-S V1.0'.
ElementName	N	Optional	See the ElementName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).

6.8.30 CIM_IndicationFilter (WQL Storage Volume OperationalStatus)

Deprecated. This is the 'pre-defined' WQL version of the CIM_IndicationFilter instance for changes in the OperationalStatus of StorageVolume instances.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='3' (Predefined Filters).

Table 53 describes class CIM_IndicationFilter (WQL Storage Volume OperationalStatus).

**Table 53 - SMI Referenced Properties/Methods for CIM_IndicationFilter
(WQL Storage Volume OperationalStatus)**

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
CreationClassName		Mandatory	See the CreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
SystemName		Mandatory	See the SystemName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
Name		Mandatory	This shall be 'SNIA:Block Services:StorageVolumeOperationalStatusWQL'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
Query		Mandatory	SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_StorageVolume AND SourceInstance.OperationalStatus <> PreviousInstance.OperationalStatus.
QueryLanguage		Mandatory	This shall be 'WQL' or 'SMI-S V1.0'.
ElementName	N	Optional	See the ElementName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).

6.8.31 CIM_LogicalDisk

Created By: Static

Modified By: Static

Deleted By: Extrinsic: StorageConfigurationService.ReturnToStoragePool

Requirement: Referenced from Volume Management - LogicalDisk is mandatory.

Table 54 describes class CIM_LogicalDisk.

Table 54 - SMI Referenced Properties/Methods for CIM_LogicalDisk

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	Opaque identifier.
ElementName		Optional	User-friendly name.
Name		Mandatory	OS Device Name.

Table 54 - SMI Referenced Properties/Methods for CIM_LogicalDisk

Properties	Flags	Requirement	Description and Notes
NameFormat		Mandatory	This shall be "12" (OS Device Name).
ExtentStatus		Mandatory	
OperationalStatus		Mandatory	Value shall be 2 3 6 8 15 (OK or Degraded or Error or Starting or Dormant).
BlockSize		Mandatory	
NumberOfBlocks		Mandatory	The number of blocks of capacity consumed from the parent StoragePool.
ConsumableBlocks		Mandatory	The number of blocks usable by consumers.
IsBasedOnUnderlyingRedundancy		Mandatory	
NoSinglePointOfFailure		Mandatory	
DataRedundancy		Mandatory	
PackageRedundancy		Mandatory	
DeltaReservation		Mandatory	
Usage		Optional	The specialized usage intended for this element.
OtherUsageDescription		Optional	Set when Usage value is "Other".
ClientSettableUsage		Optional	Lists Usage values that can be set by a client for this element.
Primordial		Mandatory	Shall be false.
ExtentDiscriminator		Mandatory	Experimental. This is an array of values that shall contain 'SNIA:Allocated'.

6.8.32 CIM_MemberOfCollection (Block Services Filter Collection to FilterCollection)

Experimental. This associates the Block Services predefined FilterCollection to the FilterCollection for the autonomous profile (e.g., the Array FilterCollection).

Requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='5' (Predefined Filter Collections).

Table 55 describes class CIM_MemberOfCollection (Block Services Filter Collection to FilterCollection).

Table 55 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Block Services Filter Collection to FilterCollection)

Properties	Flags	Requirement	Description and Notes
Collection		Mandatory	Reference to the Block Services predefined FilterCollection.
Member		Mandatory	Reference to the Block Services predefined FilterCollection.

6.8.33 CIM_MemberOfCollection (Predefined Filter Collection to Block Services Filters)

Experimental. This associates the Block Services predefined FilterCollection to the predefined Filters supported by the implementation.

Requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='5' (Predefined Filter Collections).

Table 56 describes class CIM_MemberOfCollection (Predefined Filter Collection to Block Services Filters).

**Table 56 - SMI Referenced Properties/Methods for CIM_MemberOfCollection
(Predefined Filter Collection to Block Services Filters)**

Properties	Flags	Requirement	Description and Notes
Collection		Mandatory	Reference to the Block Services predefined FilterCollection.
Member		Mandatory	Reference to the predefined IndicationFilters of the Block Services implementation.

6.8.34 CIM_OwningJobElement

Conditional on support for Job Control profile.

Requirement: Support for Job Control profile.

Table 57 describes class CIM_OwningJobElement.

Table 57 - SMI Referenced Properties/Methods for CIM_OwningJobElement

Properties	Flags	Requirement	Description and Notes
OwnedElement		Mandatory	
OwningElement		Mandatory	

6.8.35 CIM_StorageCapabilities

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 58 describes class CIM_StorageCapabilities.

Table 58 - SMI Referenced Properties/Methods for CIM_StorageCapabilities

Properties	Flags	Requirement	Description and Notes
InstanceID		Mandatory	
ElementName		Mandatory	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 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		Mandatory	Enumeration indicating the type of instance to which this StorageCapabilities applies. Shall be either 5 or 6 (StoragePool or StorageConfigurationService).

Table 58 - SMI Referenced Properties/Methods for CIM_StorageCapabilities

Properties	Flags	Requirement	Description and Notes
NoSinglePointOfFailure		Mandatory	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		Mandatory	Indicates the default value for the NoSinglePointOfFailure property.
DataRedundancyMin		Mandatory	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 .
DataRedundancyMax		Mandatory	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 .
DataRedundancyDefault		Mandatory	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 .
PackageRedundancyMin		Mandatory	PackageRedundancyMin describes the minimum number of spindles or logical devices that can be used. Package redundancy describes how many disk spindles or logical devices 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 .
PackageRedundancyMax		Mandatory	PackageRedundancyMax describes the maximum number of spindles or logical devices that can be used. Package redundancy describes how many disk spindles or logical devices 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		Mandatory	PackageRedundancyDefault describes the default number of spindles or logical devices that can be used. Package redundancy describes how many disk spindles or logical devices 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 .
ExtentStripeLengthDefault		Optional	Describes what the default stripe length, the number of members or columns, a storage element will have when created or modified using this capability. A NULL means that the setting of stripe length is not supported at all or not supported at this level of storage element allocation or assignment.
ParityLayoutDefault		Optional	ParityLayoutDefault describes what the default parity a storage element will have when created or modified using this capability. A NULL means that the setting of the parity is not supported at all or is not supported at this level of storage element allocation or assignment.
UserDataStripeDepthDefault		Optional	UserDataStripeDepthDefault describes what the number of bytes forming a stripe that a storage element will have when created or modified using this capability. A NULL means that the setting of stripe depth is not supported at all or not supported at this level of storage element allocation or assignment.

Table 58 - SMI Referenced Properties/Methods for CIM_StorageCapabilities

Properties	Flags	Requirement	Description and Notes
CreateSetting()		Conditional	Conditional requirement: Support for StorageConfigurationService. Generate a setting to use as a goal for creating or modifying storage elements.
GetSupportedStripeLengths()		Optional	List the possible discrete stripe lengths supported at this time of this method's execution.
GetSupportedStripeLengthRange()		Optional	List the possible stripe length ranges supported at the time of this method's execution.
GetSupportedParityLayouts()		Optional	List the possible parity layouts supported at the time of this method's execution.
GetSupportedStripeDepths()		Optional	List the possible stripe depths supported at the time of this method's execution.
GetSupportedStripeDepthRange()		Optional	List the possible stripe depth ranges supported at the time of this method's execution.

6.8.36 CIM_StorageConfigurationCapabilities (Concrete)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 59 describes class CIM_StorageConfigurationCapabilities (Concrete).

Table 59 - SMI Referenced Properties/Methods for CIM_StorageConfigurationCapabilities (Concrete)

Properties	Flags	Requirement	Description and Notes
InstanceID		Mandatory	
ElementName		Mandatory	
SupportedStoragePoolFeatures		Optional	Lists what StorageConfigurationService functionalities are implemented. Matches 2 3 5 6 7 (InExtents or Single InPool or Storage Pool QoS Change or Storage Pool Capacity Expansion or Storage Pool Capacity Reduction).
SupportedSynchronousActions		Optional	Lists what actions, invoked through StorageConfigurationService methods, shall not produce Concrete jobs. This International Standard recognizes "2" (Storage Pool Creation), "4" (Storage Pool Modification), "5" (Storage Element Creation), "12" (Storage Element from Element Creation), "13" (Storage Element from Element Modification) or "15" (StoragePool Usage Modification).
SupportedStorageElementTypes		Mandatory	Lists the type of storage elements that are supported by this implementation. This version of the International Standard recognizes '2' (StorageVolume) or '4' (LogicalDisk).
SupportedAsynchronousActions		Optional	Lists what actions, invoked through StorageConfigurationService methods, may produce Concrete jobs. This version of the International Standard recognizes "2" (Storage Pool Creation), "4" (Storage Pool Modification), "5" (Storage Element Creation), "12" (Storage Element from Element Creation), "13" (Storage Element from Element Modification) or "15" (StoragePool Usage Modification).

Table 59 - SMI Referenced Properties/Methods for CIM_StorageConfigurationCapabilities (Concrete)

Properties	Flags	Requirement	Description and Notes
SupportedStorageElementFeatures		Optional	Lists actions supported through the invocation of StorageServiceService.CreateOrModifyElementFromStoragePool(). Matches 3 8 (StorageVolume Creation or LogicalDisk Creation).
SupportedStorageElementUsage		Optional	Indicates the intended usage or any restrictions that may have been imposed on supported storage elements.
ClientSettableElementUsage		Optional	Indicates the intended usage or any restrictions that may have been imposed on the usage of client-settable elements.
SupportedStoragePoolUsage		Optional	Indicates the intended usage or any restrictions that may have been imposed on storage pools.
ClientSettablePoolUsage		Optional	Indicates the intended usage or any restrictions that may have been imposed on the usage of a client-settable storage pool.

6.8.37 CIM_StorageConfigurationCapabilities (Global)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Support for StorageConfigurationService.

Table 60 describes class CIM_StorageConfigurationCapabilities (Global).

Table 60 - SMI Referenced Properties/Methods for CIM_StorageConfigurationCapabilities (Global)

Properties	Flags	Requirement	Description and Notes
InstanceID		Mandatory	
ElementName		Mandatory	
SupportedStoragePoolFeatures		Optional	Lists what StorageConfigurationService functionalities are implemented. Matches 2 3 5 6 7 (InExtents or Single InPool or Storage Pool QoS Change or Storage Pool Capacity Expansion or Storage Pool Capacity Reduction).
SupportedSynchronousActions		Optional	Lists what actions, invoked through StorageConfigurationService methods, shall not produce Concrete jobs.
SupportedStorageElementTypes		Mandatory	Lists the type of storage elements that are supported by this implementation.
SupportedAsynchronousActions		Optional	Lists what actions, invoked through StorageConfigurationService methods, may produce Concrete jobs.
SupportedStorageElementFeatures		Optional	Lists actions supported through the invocation of StorageServiceService.CreateOrModifyElementFromStoragePool(). Matches 3 5 8 9 11 12 13 (StorageVolume Creation or StorageVolume Modification or LogicalDisk Creation or LogicalDisk Modification or Storage Element QoS Change or Storage Element Capacity Expansion or Storage Element Capacity Reduction).
SupportedStorageElementUsage		Optional	Indicates the intended usage or any restrictions that may have been imposed on supported storage elements.

Table 60 - SMI Referenced Properties/Methods for CIM_StorageConfigurationCapabilities (Global)

Properties	Flags	Requirement	Description and Notes
ClientSettableElementUsage		Optional	Indicates the intended usage or any restrictions that may have been imposed on the usage of client-settable elements.
SupportedStoragePoolUsage		Optional	Indicates the intended usage or any restrictions that may have been imposed on storage pools.
ClientSettablePoolUsage		Optional	Indicates the intended usage or any restrictions that may have been imposed on the usage of a client-settable storage pool.

6.8.38 CIM_StorageConfigurationCapabilities (Primordial)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 61 describes class CIM_StorageConfigurationCapabilities (Primordial).

Table 61 - SMI Referenced Properties/Methods for CIM_StorageConfigurationCapabilities (Primordial)

Properties	Flags	Requirement	Description and Notes
InstanceID		Mandatory	
ElementName		Mandatory	
SupportedStoragePoolFeatures		Optional	Lists what StorageConfigurationService functionalities are implemented. Matches 2 3 (InExtents or Single InPool).
SupportedSynchronousActions		Optional	Lists what actions, invoked through StorageConfigurationService methods, shall not produce Concrete jobs. This version of the International Standard recognizes "2" (Storage Pool Creation), "12" (Storage Element from Element Creation) or "15" (StoragePool Usage Modification).
SupportedStorageElementTypes		Optional	Lists the type of storage elements that are supported by this implementation. This version of the International Standard does not recognize any values for this property.
SupportedAsynchronousActions		Optional	Lists what actions, invoked through StorageConfigurationService methods, may produce Concrete jobs. This version of the International Standard recognizes "2" (Storage Pool Creation), "12" (Storage Element from Element Creation) or "15" (StoragePool Usage Modification).
SupportedStorageElementFeatures		Optional	Lists actions supported through the invocation of StorageServiceService.CreateOrModifyElementFromStoragePool(). This version of the International Standard does not recognize any values for this property. For Primordial pools, this shall not contain 3 (StorageVolume Creation), 5 (StorageVolume Modification), 8 (LogicalDisk Creation) or 9 (LogicalDisk Modification).
SupportedStorageElementUsage		Optional	For Primordial StorageConfigurationCapabilities, this shall be NULL.

Table 61 - SMI Referenced Properties/Methods for CIM_StorageConfigurationCapabilities (Primordial)

Properties	Flags	Requirement	Description and Notes
ClientSettableElementUsage		Optional	For Primordial StorageConfigurationCapabilities, this shall be NULL.
SupportedStoragePoolUsage		Optional	Indicates the intended usage or any restrictions that may have been imposed on storage pools.
ClientSettablePoolUsage		Optional	Indicates the intended usage or any restrictions that may have been imposed on the usage of a client-settable storage pool.

6.8.39 CIM_StorageConfigurationService

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 62 describes class CIM_StorageConfigurationService.

Table 62 - SMI Referenced Properties/Methods for CIM_StorageConfigurationService

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	
CreationClassName		Mandatory	
SystemName		Mandatory	
Name		Mandatory	
CreateOrModifyStoragePool()		Optional	Create (or modify) a StoragePool. A job may be created as well.
DeleteStoragePool()		Optional	Start a job to delete a StoragePool.
CreateOrModifyElementFromStoragePool()		Mandatory	Create or modify a storage element. A job may be created as well.
CreateOrModifyElementFromElements()		Optional	Create or modify a storage element using component StorageExtents of the Pool. A job may be created as well.
ReturnToStoragePool()		Mandatory	Release the capacity represented by this storage element back to the Pool.
RequestUsageChange()		Optional	Allows a client to change the Usage for the element.
GetElementsBasedOnUsage()		Optional	Allows a client to retrieve elements for a specialized Usage.

6.8.40 CIM_StoragePool (Concrete)

Created By: Extrinsic: StorageConfigurationService.CreateOrModifyStoragePool

Modified By: Extrinsic: StorageConfigurationService.CreateOrModifyStoragePool

Deleted By: Extrinsic: StorageConfigurationService.DeleteStoragePool

Requirement: Mandatory.

Table 63 describes class CIM_StoragePool (Concrete).

Table 63 - SMI Referenced Properties/Methods for CIM_StoragePool (Concrete)

Properties	Flags	Requirement	Description and Notes
Primordial		Mandatory	Shall be false.
InstanceID		Mandatory	
ElementName		Optional	
PoolID		Mandatory	A unique name in the context of this system that identifies this Pool.
TotalManagedSpace		Mandatory	
RemainingManagedSpace		Mandatory	
Usage		Optional	The specialized usage intended for this element.
OtherUsageDescription		Optional	Set when Usage value is "Other".
ClientSettableUsage		Optional	Lists Usage values that can be set by a client for this element.
GetSupportedSizes()		Conditional	Conditional requirement: Support for StorageConfigurationService. List the discrete storage element sizes that can be created or expanded from this Pool.
GetSupportedSizeRange()		Conditional	Conditional requirement: Support for StorageConfigurationService. List the size ranges for storage element that can be created or expanded from this Pool.
GetAvailableExtents()		Optional	List the StorageExtents from this Pool that may be used to create or expand a storage element. The StorageExtents may not already be in use as supporting capacity for existing storage element.

6.8.41 CIM_StoragePool (Empty)

An empty StoragePool is a special case of a StoragePool where the StoragePool contains no capacity. All properties are supported as defined for the StoragePool (Concrete or Primordial), except that the empty StoragePool has TotalManagedSpace=0.

Created By: Extrinsic: StorageConfigurationService.CreateOrModifyStoragePool

Modified By: Extrinsic: StorageConfigurationService.CreateOrModifyStoragePool

Deleted By: Extrinsic: StorageConfigurationService.DeleteStoragePool

Requirement: Optional.

Table 64 describes class CIM_StoragePool (Empty).

Table 64 - SMI Referenced Properties/Methods for CIM_StoragePool (Empty)

Properties	Flags	Requirement	Description and Notes
Primordial		Mandatory	This may be either true or false. That is, both concrete and primordial StoragePools may be empty.
InstanceID		Mandatory	
ElementName		Optional	
PoolID		Mandatory	
TotalManagedSpace		Mandatory	This shall be 0 for an empty StoragePool.
RemainingManagedSpace		Mandatory	

Table 64 - SMI Referenced Properties/Methods for CIM_StoragePool (Empty)

Properties	Flags	Requirement	Description and Notes
Usage		Optional	
OtherUsageDescription		Optional	
ClientSettableUsage		Optional	
GetSupportedSizes()		Conditional	Conditional requirement: Support for StorageConfigurationService.
GetSupportedSizeRange()		Conditional	Conditional requirement: Support for StorageConfigurationService.
GetAvailableExtents()		Optional	

6.8.42 CIM_StoragePool (Primordial)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 65 describes class CIM_StoragePool (Primordial).

Table 65 - SMI Referenced Properties/Methods for CIM_StoragePool (Primordial)

Properties	Flags	Requirement	Description and Notes
Primordial		Mandatory	Shall be true.
InstanceID		Mandatory	
ElementName		Optional	
PoolID		Mandatory	A unique name in the context of this system that identifies this Pool.
TotalManagedSpace		Mandatory	
RemainingManagedSpace		Mandatory	
Usage		Optional	The specialized usage intended for this element.
OtherUsageDescription		Optional	Set when Usage value is "Other".
ClientSettableUsage		Optional	Lists Usage values that can be set by a client for this element.
GetSupportedSizes()		Conditional	Conditional requirement: Support for StorageConfigurationService. List the discrete storage element sizes that can be created or expanded from this Pool.
GetSupportedSizeRange()		Conditional	Conditional requirement: Support for StorageConfigurationService. List the size ranges for storage element that can be created or expanded from this Pool.
GetAvailableExtents()		Optional	List the StorageExtents from this Pool that may be used to create or expand a storage element. The StorageExtents may not already be in use as supporting capacity for existing storage element.

6.8.43 CIM_StorageSetting

Created By: Extrinsic: StorageCapabilities.CreateSetting

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 66 describes class CIM_StorageSetting.

Table 66 - SMI Referenced Properties/Methods for CIM_StorageSetting

Properties	Flags	Requirement	Description and Notes
InstanceID		Mandatory	
ElementName		Mandatory	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 Name does not have to be unique within a namespace).
NoSinglePointOfFailure		Mandatory	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.
DataRedundancyMin		Mandatory	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 .
DataRedundancyMax		Mandatory	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 .
DataRedundancyGoal		Mandatory	
PackageRedundancyMin		Mandatory	PackageRedundancyMin describes the minimum number of spindles or logical devices to be used. Package redundancy describes how many disk spindles or logical devices 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 .
PackageRedundancyMax		Mandatory	PackageRedundancyMax describes the maximum number of spindles or logical devices to be used. Package redundancy describes how many disk spindles or logical devices 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 .
PackageRedundancyGoal		Mandatory	
ExtentStripeLength		Optional	ExtentStripeLength describes the desired stripe length goal.
ExtentStripeLengthMin		Optional	ExtentStripeLengthMin describes the minimum acceptable stripe length.
ExtentStripeLengthMax		Optional	ExtentStripeLengthMax describes the maximum acceptable stripe length.
ParityLayout		Optional	ParityLayout describes the desired parity layout. The value may be 1 or 2 (Non-rotated Parity or Rotated Parity).
UserDataStripeDepth		Optional	UserDataStripeDepth describes the desired stripe depth.
UserDataStripeDepthMin		Optional	UserDataStripeDepthMin describes the minimum acceptable stripe depth.

Table 66 - SMI Referenced Properties/Methods for CIM_StorageSetting

Properties	Flags	Requirement	Description and Notes
UserDataStripeDepthMax		Optional	UserDataStripeDepthMax describes the maximum acceptable stripe depth.
ChangeableType		Mandatory	This property informs a client if the setting can be modified. It also tells the client how long this setting is expected to remain in the model. If the implementation allows it, the client can use the property to request that the setting's existence be not transient.
StorageExtentInitialUsage		Optional	The Usage value to be used when creating a new storage element.
StoragePoolInitialUsage		Optional	The Usage value to be used when creating a new storage pool.

6.8.44 CIM_StorageSettingWithHints

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 67 describes class CIM_StorageSettingWithHints.

Table 67 - SMI Referenced Properties/Methods for CIM_StorageSettingWithHints

Properties	Flags	Requirement	Description and Notes
InstanceID		Mandatory	
ElementName		Mandatory	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 Name does not have to be unique within a namespace.)
NoSinglePointOfFailure		Mandatory	
DataRedundancyMin		Mandatory	
DataRedundancyMax		Mandatory	
DataRedundancyGoal		Mandatory	
PackageRedundancyMin		Mandatory	
PackageRedundancyMax		Mandatory	
PackageRedundancyGoal		Mandatory	
ExtentStripeLength		Optional	
ExtentStripeLengthMin		Optional	
ExtentStripeLengthMax		Optional	
ParityLayout		Optional	
UserDataStripeDepth		Optional	
UserDataStripeDepthMin		Optional	
UserDataStripeDepthMax		Optional	
StorageExtentInitialUsage		Optional	
StoragePoolInitialUsage		Optional	

Table 67 - SMI Referenced Properties/Methods for CIM_StorageSettingWithHints

Properties	Flags	Requirement	Description and Notes
DataAvailabilityHint		Mandatory	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		Mandatory	This hint is an indication from a client of the randomness of accesses. Values are 0=Entirely Sequential to 10=Entirely Random.
AccessDirectionHint		Mandatory	This hint is an indication from a client of the direction of accesses. Values are 0=Entirely Read to 10=Entirely Write.
AccessSizeHint		Mandatory	This hint is an indication from a client of the optimal access sizes. Several sizes can be specified. Units ("Megabytes").
AccessLatencyHint		Mandatory	This hint is an indication from a client how important access latency is. Values are 0=Don't Care to 10=Very Important.
AccessBandwidthWeight		Mandatory	This hint is an indication from a client of bandwidth prioritization. Values are 0=Don't Care to 10=Very Important.
StorageCostHint		Mandatory	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		Mandatory	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.
ChangeableType		Mandatory	

6.8.45 CIM_StorageSettingsAssociatedToCapabilities

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 68 describes class CIM_StorageSettingsAssociatedToCapabilities.

Table 68 - SMI Referenced Properties/Methods for CIM_StorageSettingsAssociatedToCapabilities

Properties	Flags	Requirement	Description and Notes
DefaultSetting		Mandatory	This boolean designates the setting that will be used if the CreateSetting() method is called with providing the NewSetting parameter. However, some implementations may require that the NewSetting parameter be non null. There may be only one default setting per the combination of StorageCapabilities and associated StoragePool as associated through ElementCapabilities.
Dependent		Mandatory	The StorageSetting reference.
Antecedent		Mandatory	The StorageCapabilities reference.

6.8.46 CIM_StorageSettingsGeneratedFromCapabilities

Created By: Extrinsic: CreateSetting

Modified By: Static

Deleted By: Static

Requirement: Support for StorageConfigurationService.

Table 69 describes class CIM_StorageSettingsGeneratedFromCapabilities.

Table 69 - SMI Referenced Properties/Methods for CIM_StorageSettingsGeneratedFromCapabilities

Properties	Flags	Requirement	Description and Notes
Dependent		Mandatory	The StorageSetting reference.
Antecedent		Mandatory	The StorageCapabilities reference.

6.8.47 CIM_StorageVolume

Created By: Static

Modified By: Static

Deleted By: Extrinsic: StorageConfigurationService.ReturnToStoragePool.

Requirement: Referenced from Array - StorageVolume is mandatory or Referenced from Storage Virtualizer - StorageVolume is mandatory or Referenced from Host Hardware RAID Controller - StorageVolume is mandatory.

Table 70 describes class CIM_StorageVolume.

Table 70 - SMI Referenced Properties/Methods for CIM_StorageVolume

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	Opaque identifier.
ElementName		Optional	User-friendly name.
Name	CD	Mandatory	Identifier for this volume; based on datapath standards such as SCSI or ATAPI.
OtherIdentifyingInfo	CD	Optional	Additional correlatable names.
IdentifyingDescriptions		Optional	

Table 70 - SMI Referenced Properties/Methods for CIM_StorageVolume

Properties	Flags	Requirement	Description and Notes
NameFormat		Mandatory	The type of identifier in the Name property. The valid values for StorageVolumes are: 1 (Other) 2 (VPD83NAA6) 3 (VPD83NAA5) 4 (VPD83Type2) 5 (VPD83Type1) 6 (VPD83Type0) 7 (SNVM) 8 (NodeWWN) 9 (NAA) 10 (EUI64) 11 (T10VID).
NameNamespace		Mandatory	The namespace that defines uniqueness for the NameFormat.
ExtentStatus		Mandatory	
OperationalStatus		Mandatory	Value shall be 2 3 6 8 15 (OK or Degraded or Error or Starting or Dormant).
BlockSize		Mandatory	
NumberOfBlocks		Mandatory	The number of blocks of capacity consumed from the parent StoragePool.
ConsumableBlocks		Mandatory	The number of blocks usable by consumers.
IsBasedOnUnderlyingRedundancy		Mandatory	
NoSinglePointOfFailure		Mandatory	
DataRedundancy		Mandatory	
PackageRedundancy		Mandatory	
DeltaReservation		Mandatory	
Usage		Optional	The specialized usage intended for this element.
OtherUsageDescription		Optional	Set when Usage value is "Other".
ClientSettableUsage		Optional	Lists Usage values that can be set by a client for this element.
Primordial		Mandatory	Shall be false.
ExtentDiscriminator		Mandatory	Experimental. This is an array of values that shall contain 'SNIA:Allocated'.

6.8.48 CIM_SystemDevice (System to StorageVolume or LogicalDisk)

Created By: Static

Modified By: Static

Deleted By: Extrinsic: StorageConfigurationService.ReturnToStoragePool

Requirement: Mandatory.

Table 71 describes class CIM_SystemDevice (System to StorageVolume or LogicalDisk).

**Table 71 - SMI Referenced Properties/Methods for CIM_SystemDevice
(System to StorageVolume or LogicalDisk)**

Properties	Flags	Requirement	Description and Notes
PartComponent		Mandatory	
GroupComponent		Mandatory	

6.8.49 SNIA_StorageVolume

This represents the same instance as CIM_StorageVolume, but is extended to support the CanDelete property.

Created By: Static

Modified By: Static

Deleted By: Extrinsic: StorageConfigurationService.ReturnToStoragePool

Requirement: Optional.

IECNORM.COM : Click to view the full PDF of ISO/IEC 24775-4:2014

Table 72 describes class SNIA_StorageVolume.

Table 72 - SMI Referenced Properties/Methods for SNIA_StorageVolume

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	
ElementName		Optional	
Name		Mandatory	
OtherIdentifyingInfo		Optional	
IdentifyingDescriptions		Optional	
NameFormat		Mandatory	
NameNamespace		Mandatory	
ExtentStatus		Mandatory	
OperationalStatus		Mandatory	
BlockSize		Mandatory	
NumberOfBlocks		Mandatory	
ConsumableBlocks		Mandatory	
IsBasedOnUnderlyingRedundancy		Mandatory	
NoSinglePointOfFailure		Mandatory	
DataRedundancy		Mandatory	
PackageRedundancy		Mandatory	
DeltaReservation		Mandatory	
Usage		Optional	
OtherUsageDescription		Optional	
ClientSettableUsage		Optional	
Primordial		Mandatory	
ExtentDiscriminator		Mandatory	Experimental. This is an array of values that shall contain 'SNIA:Allocated'.
CanDelete		Optional	Experimental. Indicates if the volume is able to be deleted by a client application.

STABLE

EXPERIMENTAL

7 Block Storage Views Profile

7.1 Description

7.1.1 Synopsis

Profile Name: Block Storage Views (Component Profile)

Version: 1.5.0

Organization: SNIA

CIM Schema Version: 2.23

Table 73 describes the related profiles for Block Storage Views.

Table 73 - Related Profiles for Block Storage Views

Profile Name	Organization	Version	Requirement	Description
Block Services	SNIA	1.5.0	Conditional	Conditional requirement: Required if the array property SNIA_ViewCapabilities.Support edViews contains the string "VolumeView" (and the Block Service Package is implemented).
Block Server Performance	SNIA	1.5.0	Conditional	
Disk Drive Lite	SNIA	1.5.0	Conditional	Conditional requirement: Required if the array property SNIA_ViewCapabilities.Support edViews contains the string "DiskDriveView" (and the Disk Drive Lite Profile is implemented).
Masking and Mapping	SNIA	1.4.0	Conditional	
Extent Composition	SNIA	1.5.0	Conditional	Conditional requirement: Required if the array property SNIA_ViewCapabilities.Support edViews contains the string "VolumeView" and Extent Composition is implemented.
Copy Services	SNIA	1.5.0	Conditional	Conditional requirement: Required if the array property SNIA_ViewCapabilities.Support edViews contains the string "ReplicaPairView" (and the Copy Services Profile is implemented).

Central Class: SNIA_ViewCapabilities

Scoping Class: CIM_ComputerSystem

7.1.2 Overview

7.1.2.1 General

This Profile specifies SNIA_ View Classes for the Array, Storage Virtualizer and Volume Management Profiles.

In this release of SMI-S, SNIA_ view classes provide an optimization of retrieval of information provided by multiple (associated) instances in a Profile. There is no support for update of SNIA_ view classes instances. Update of a SNIA_ view class instance can only be accomplished by updating the base class instances from which the view is derived.

7.1.2.2 Goals of SNIA_ View Classes

7.1.2.2.1 Goals that SNIA_ View Classes are intended to address are

- Get more data in one call to CIM Server.

The CIM model for arrays and Storage Virtualizers involve a lot of classes and associations. The objective is to allow discovery of the array model using SNIA_ View Classes with a reduction in the number of association traversals required.

- Allow providers to optimize the Request.

In many cases, the data represented by a View Class is actually kept (and returned) by a device as one entity. When the "normalized" CIM model is traversed many calls are made to retrieve that one entity. The provider takes the data from the one entity and carves it up for each CIM request. In many cases this involves retrieving the same entity multiple times. The objective is to allow a Provider to return the single entity in one SMI-S request (for data that is typically kept together by the device).

7.1.2.2.2 Additional Goals

- Do more things in one call to CIM Server.

An example would be retrieval or discovery of model information with fewer calls. However, this goal also extends to updating the CIM model (e.g., configuration actions). The SNIA_ View Classes are NOT intended to help in the latter case. However, SNIA_ View Classes should facilitate access to underlying classes in support of configuration operations.

It is important to note that the SNIA_ View Classes proposal was based directly on experiences relating to the scalability and performance of SMI-S real-world implementations. The focus is on improving performance in large configurations (e.g. thousands of volumes and thousands of disk drives).

7.1.2.3 Specific Requirements and Objectives of View Classes

7.1.2.3.1 Pre-defined View Classes

In order to gain the desired performance advantage, it is felt that view classes would have to be pre-defined (in SMI-S) to allow provider optimization of the requested information.

- Enable Associator Calls to View Class instances.

It should be possible to retrieve a View Class by an associators call to the class.

However, it is desired that the association should be clearly distinguished from existing associations on the base classes.

- Enable Associator Calls from View Class instances.

It should be possible to get related classes (e.g., base classes) from the View Class by using associator calls.

Again, the associations used should be clearly distinguished from existing associations on the base classes.

7.1.2.3.2 Specific Views requested

- Getting asset information
- Mix of StorageVolume with LUN Mapping and Masking

Most of these requests are addressed by this International Standard.

- Allow View Classes to be used where other classes are used.

This certainly includes "read" intrinsic and as parameters of Extrinsic

However, at this time "Write" intrinsic support is deferred and use in Extrinsics (as IN or OUT parameters) is not covered in this release of SMI-S.

7.1.2.3.3 Support Life Cycle Indications on View Classes

This requirement is being deferred for consideration in a future release of SMI-S.

7.1.3 Class Diagram for SNIA View Classes

Figure 25 illustrates the class diagram for SNIA_view classes.

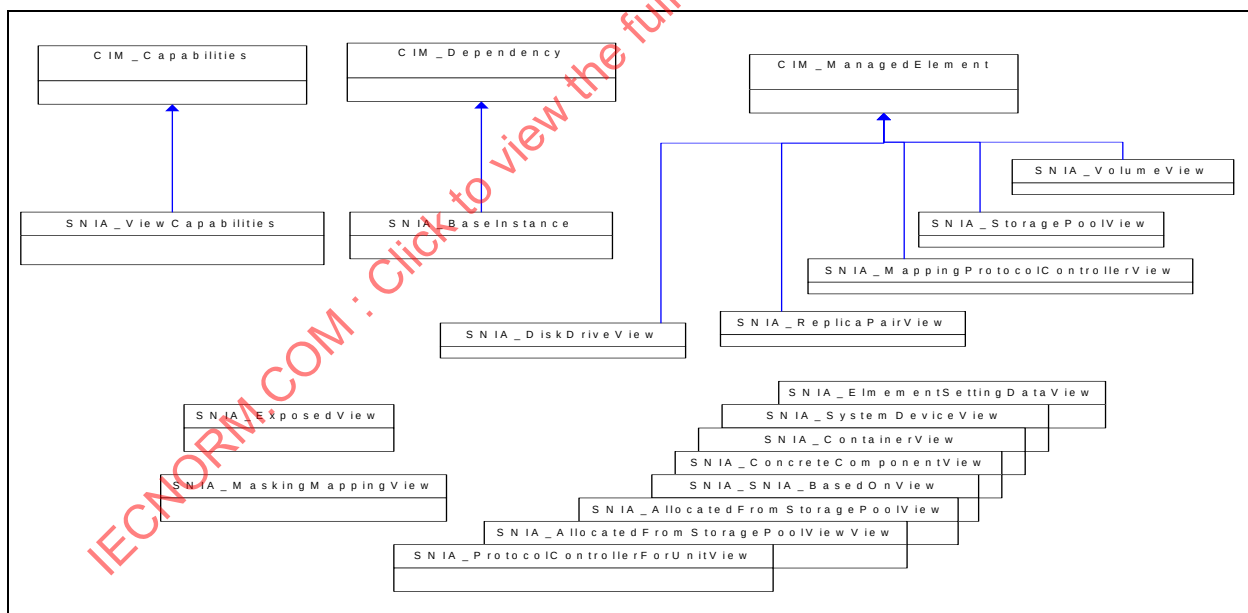


Figure 25 - Class Diagram for SNIA_ View Classes

The `SNIA_ViewCapabilities` inherits from `CIM_Capabilities`. The `SNIA_VolumeView` and `SNIA_DiskDriveView` classes inherit from `CIM_ManagedElement`. The `SNIA_` association views (`SNIA_ExposedView` and `SNIA_MaskingMappingView`) do not inherit from anything.

7.1.4 Implementation

7.1.4.1 View Class Capabilities

The implementation shall identify which view classes are implemented using a set of conditions. The model for determining whether or not the Block Storage Views Profile is supported and which views are supported is illustrated in Figure 26.

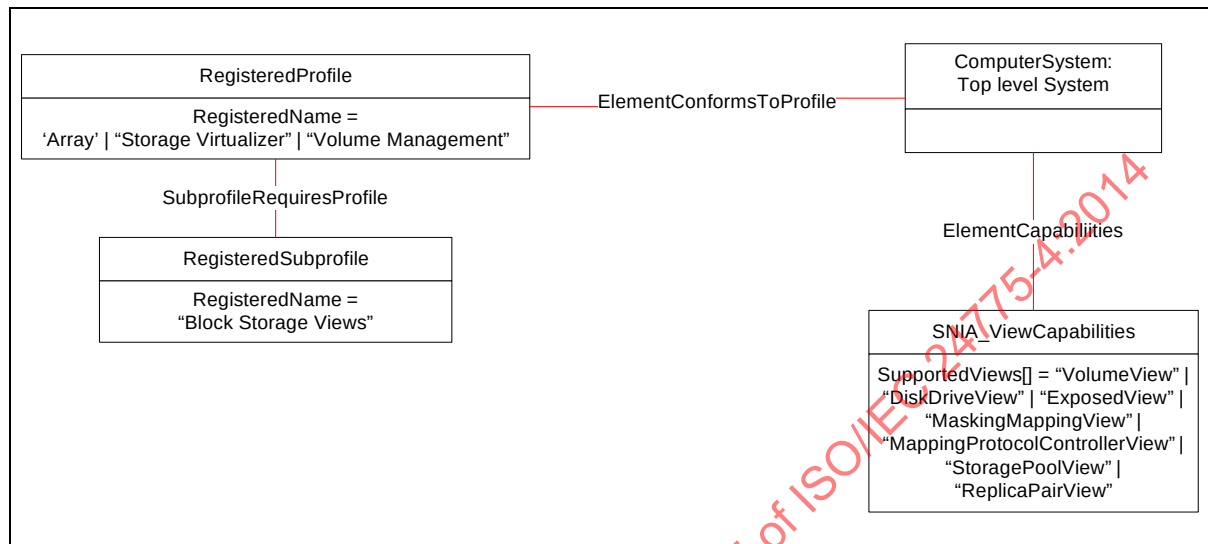


Figure 26 - Block Storage View Class Capabilities

First a client may determine whether or not a profile implementation has implemented any view classes by looking for a **RegisteredSubprofile** with a **RegisteredName** of "Block Storage Views". If this **RegisteredSubprofile** exists then the profile supports some number of view classes.

Next a client would be able to determine which view classes are supported by an implementation by following the **ElementConformsToProfile** to the top level system and then following the **ElementCapabilities** from that system to the **SNIA_ViewCapabilities** instance. There shall be one instance of the **SNIA_ViewCapabilities** class if the profile supports the Block Storage Views Subprofile. The **SNIA_ViewCapabilities** instance shall have an array of strings (**SupportedViews**) that identify the view classes that are supported. For example, if the **SupportedViews** array includes the "VolumeView" string, then the **VolumeView** class shall be supported.

7.1.4.2 Storage Volume Views

7.1.4.2.1 SNIA_VolumeView and related associations

Figure 27 illustrates the **SNIA_VolumeView** and related associations.

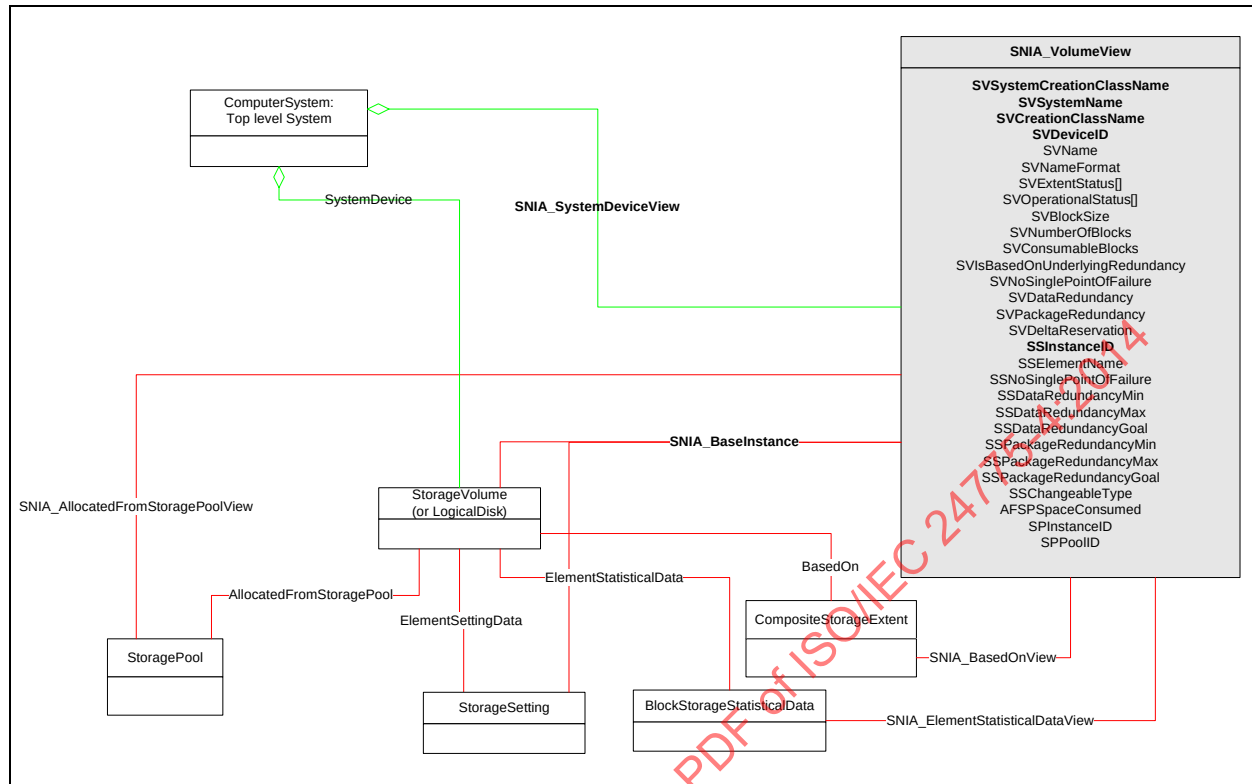


Figure 27 - SNIA_VolumeView and related associations

The SNIA_VolumeView is composed of information drawn from the following base classes:

- StorageVolume (or LogicalDisk)
- StorageSetting
- AllocatedFromStoragePool
- StoragePool

The keys for the SNIA_VolumeView are the StorageVolume and StoragePool keys from the base CIM_StorageVolume and StoragePool instances. There will be one instance of SNIA_VolumeView for each instance of StorageVolume if the StorageVolume is allocated from one StoragePool. If a StorageVolume is allocated from multiple StoragePools (e.g., Composite Volumes), there will be one instance of SNIA_StorageVolume for each StoragePool from which the StorageVolume is allocated.

The information drawn from the AllocatedFromStoragePool association is the SpaceConsumed property. The properties from all other base classes shall be supported, but may be null.

7.1.4.2.2 Mandatory, Conditional and Optional Properties of SNIA_VolumeView

Properties that are mandatory in the mandatory base classes are mandatory in the SNIA_VolumeView class. Properties that are Conditional in the base classes are conditional in the SNIA_VolumeView class. Properties that are mandatory in optional (base) classes (CompositeExtent) are "conditional" in the SNIA_VolumeView. If an optional base class is not supported by the Array or StorageVirtualizer implementation, these properties of those classes shall be present, but shall be null.

Properties in the base classes that are optional in the base class are optional in the SNIA_VolumeView.

7.1.4.2.3 Associations on SNIA_VolumeView

7.1.4.2.3.1 General

In this release of SMI-S the SNIA_VolumeView is "read only." Access to CIM class instances on which the view is based that can be updated (e.g., StorageVolume and StorageSetting) can be accessed from the SNIA_VolumeView instance via the SNIA_BaseInstance association.

In addition to the SNIA_VolumeView there are four associations that support association traversal to (or from) instances of the SNIA_VolumeView:

7.1.4.2.3.2 SNIA_SystemDeviceView

From the owning CIM_ComputerSystem a client will be able to find the SNIA_VolumeViews associated to the ComputerSystem via the SNIA_SystemDeviceView. This will return the VolumeViews that correspond to the StorageVolumes (or LogicalDisks) that would be found via association traversal from the ComputerSystem to the StorageVolumes (or LogicalDisks) via CIM_SystemDevice.

7.1.4.2.3.3 SNIA_AllocatedFromStoragePoolView

From the SNIA_VolumeView instance, the client can find the CIM_StoragePool instance by following the SNIA_AllocatedFromStoragePoolView association. Note that for one SNIA_VolumeView instance, there may be one or more CIM_StoragePools (that is, for Composite Volumes that draw from multiple StoragePools, there would be multiple SNIA_VolumeView instances that represent the composite volume.)

7.1.4.2.3.4 SNIA_BasedOnView

From the SNIA_VolumeView instance, the client can find the CIM_StorageExtent(s) on which the StorageVolume (or LogicalDisk) is based by following the BasedOnView.

Similarly, from a "top level" CIM_StorageExtent instance, a client can find the SNIA_VolumeView instance(s) that are based on that StorageExtent.

7.1.4.2.3.5 SNIA_ElementStatisticalDataView

From the SNIA_VolumeView instance, the client can find the CIM_BlockStorageStatisticalData instance for the StorageVolume or LogicalDisk of the VolumeView by following the SNIA_ElementStatisticalDataView association.

7.1.4.3 Disk Drive Views

7.1.4.3.1 General

Figure 28 illustrates the DiskDriveView class and related associations.

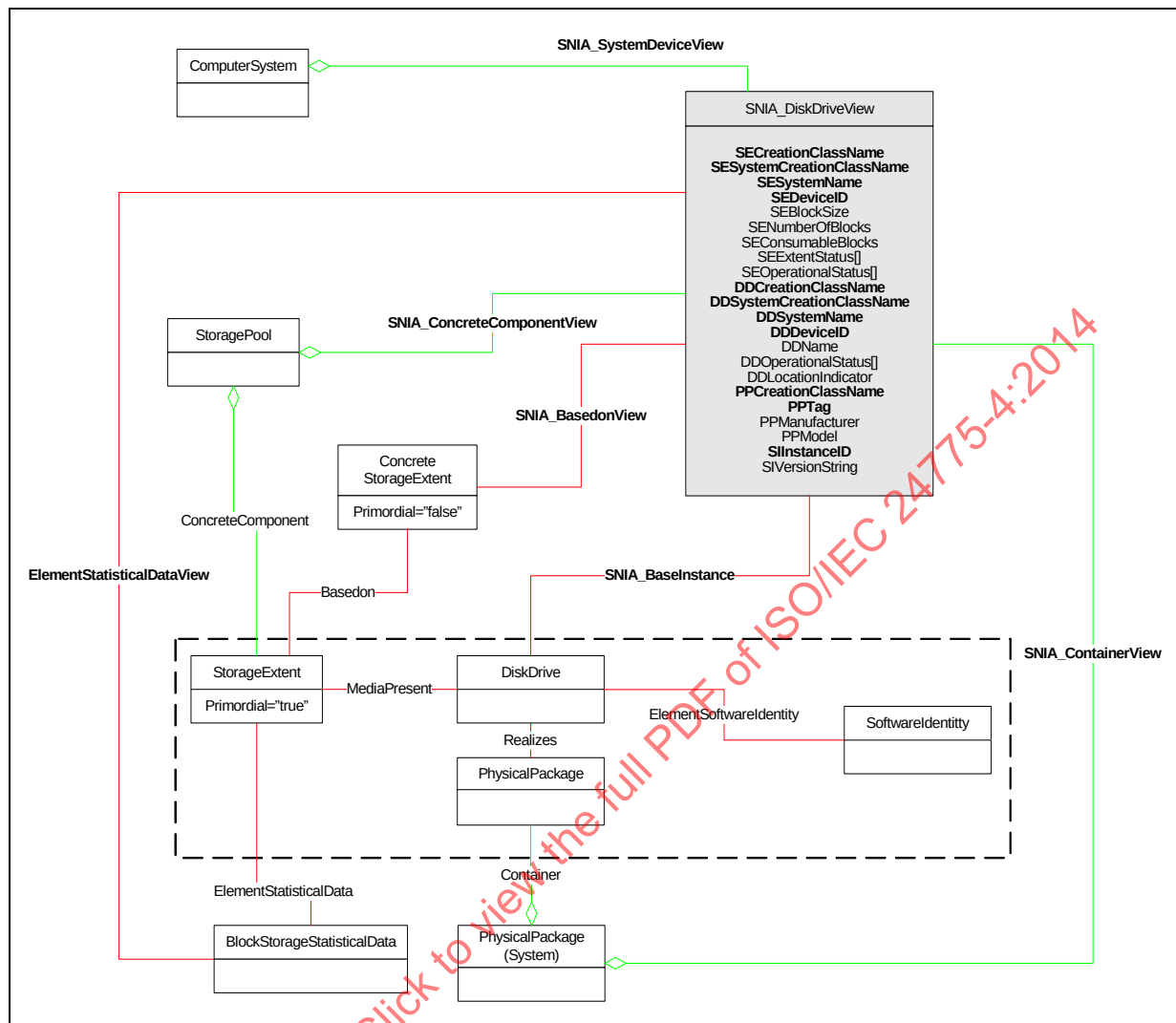


Figure 28 - SNIA_DiskDriveView and related associations

The SNIA_DiskDriveView is composed of information drawn from the following base classes:

- StorageExtent
- DiskDrive
- PhysicalPackage
- SoftwareIdentity (conditional)

The keys for the SNIA_DiskDriveView are the keys of the DiskDrive base class. There will be one instance of SNIA_DiskDriveView for each instance of a Disk Drive (primordial).

7.1.4.3.2 Mandatory, Conditional and Optional Properties of SNIA_DiskDriveView

The properties from base classes shall be supported, but may be null. Properties that are mandatory in mandatory base classes are mandatory in the SNIA_DiskDriveView class. Properties that are conditional in a base class are conditional in the SNIA_DiskDriveView class. Properties that are mandatory in optional (base) classes (BlockStorageStatisticalData and SoftwareIdentity) are also "conditional" in the SNIA_DiskDriveView. If an optional base class is not supported by the Array implementation, these properties of those classes shall be present but shall be null.

Properties in the base classes that are optional in the base class are optional in the SNIA_DiskDriveView.

7.1.4.3.3 Associations on SNIA_DiskDriveView

7.1.4.3.3.1 General

In this release of SMI-S, the SNIA_DiskDriveView is "read only." In order to support update of information in the SNIA_DiskDriveView instance, it would be necessary to update the class instances on which it is based. An association SNIA_BaseInstance is provided to the CIM_DiskDrive instance.

NOTE The SNIA_BaseInstance association is only provided to base instances that can be modified.

In addition to the SNIA_DiskDriveView there are 5 associations that support association traversal to (or from) instances of the SNIA_DiskDriveView:

7.1.4.3.3.2 SNIA_ConcreteComponentView (mandatory if the DiskDriveView is implemented)

From a primordial CIM_StoragePool instance a client will be able to find the SNIA_DiskDriveViews associated to the StoragePool via the SNIA_ConcreteComponentView. This will return the DiskDriveView instances that correspond to the Disk Drive StorageExtents that would be found via association traversal from the StoragePool to the StorageExtents via CIM_ConcreteComponent association.

Similarly, if the client has a SNIA_DiskDriveView instance, the client can find the primordial StoragePool to which the drive is assigned by following the SNIA_ConcreteComponentView association from the SNIA_DiskDriveView instance to the CIM_StoragePool instance for the StoragePool that contains the Disk Drive StorageExtent.

7.1.4.3.3.3 SNIA_ContainerView (mandatory if the DiskDriveView is implemented)

From a system chassis (or other higher level physical package) instance a client will be able to find the SNIA_DiskDriveViews associated to the CIM_PhysicalPackage instance via the SNIA_ContainerView. This will return the DiskDriveView instances that correspond to the Disk Drive PhysicalPackage that would be found via association traversal from the system CIM_PhysicalPackage to the Disk Drive CIM_PhysicalPackage via CIM_Container association.

Similarly, if the client has a SNIA_DiskDriveView instance, the client can find the higher level system CIM_PhysicalPackage instance in which the drive resides by following the SNIA_ContainerView association from the SNIA_DiskDriveView instance to the CIM_PhysicalPackage instance for the higher level system physical package that contains the Disk Drive physical package.

7.1.4.3.3.4 SNIA_BasedOnView (mandatory if the DiskDriveView and Extent Composition are implemented)

From a concrete StorageExtent (e.g., CompositeExtent) instance from Extent Composition a client will be able to find the SNIA_DiskDriveViews associated to the CIM_StorageExtent instance via the SNIA_BasedOnView. This will return the DiskDriveView instances that correspond to the Disk Drive StorageExtent that would be found via association traversal from a "most antecedent" concrete CIM_StorageExtent to the Disk Drive CIM_StorageExtent via CIM_BasedOn association.

Similarly, if the client has a SNIA_DiskDriveView instance, the client can find concrete CIM_StorageExtent instance(s) that is (are) based on the drive by following the SNIA_BasedOnView association from the SNIA_DiskDriveView instance to the CIM_StorageExtent instance(s) for the concrete storage extent(s) that is (are) based on the Disk Drive storage extent.

7.1.4.3.3.5 SNIA_SystemDeviceView (mandatory if the DiskDriveView is implemented)

From the owning CIM_ComputerSystem a client will be able to find the SNIA_DiskDriveViews associated to the ComputerSystem via the SNIA_SystemDeviceView. This will return the

DiskDriveViews that correspond to the CIM_DiskDrive instances that would be found via association traversal from the ComputerSystem to the CIM_DiskDrive instances via CIM_SystemDevice.

Similarly, if the client has a SNIA_DiskDriveView instance, the client can find the owning ComputerSystem by following the SNIA_SystemDeviceView association from the SNIA_DiskDriveView instance to the CIM_ComputerSystem instance for the ComputerSystem that scopes the CIM_DiskDrive instances.

7.1.4.3.3.6 SNIA_ElementStatisticalDataView

From the SNIA_DiskDriveView instance, the client can find the CIM_BlockStorageStatisticalData instance for the Disk Drive StorageExtent of the DiskDriveView by following the SNIA_ElementStatisticalDataView association.

7.1.4.4 Masking and Mapping Views

7.1.4.4.1 The SNIA_ExposedView Association

7.1.4.4.1.1 General

Figure 29 illustrates the SNIA_ExposedView Association.

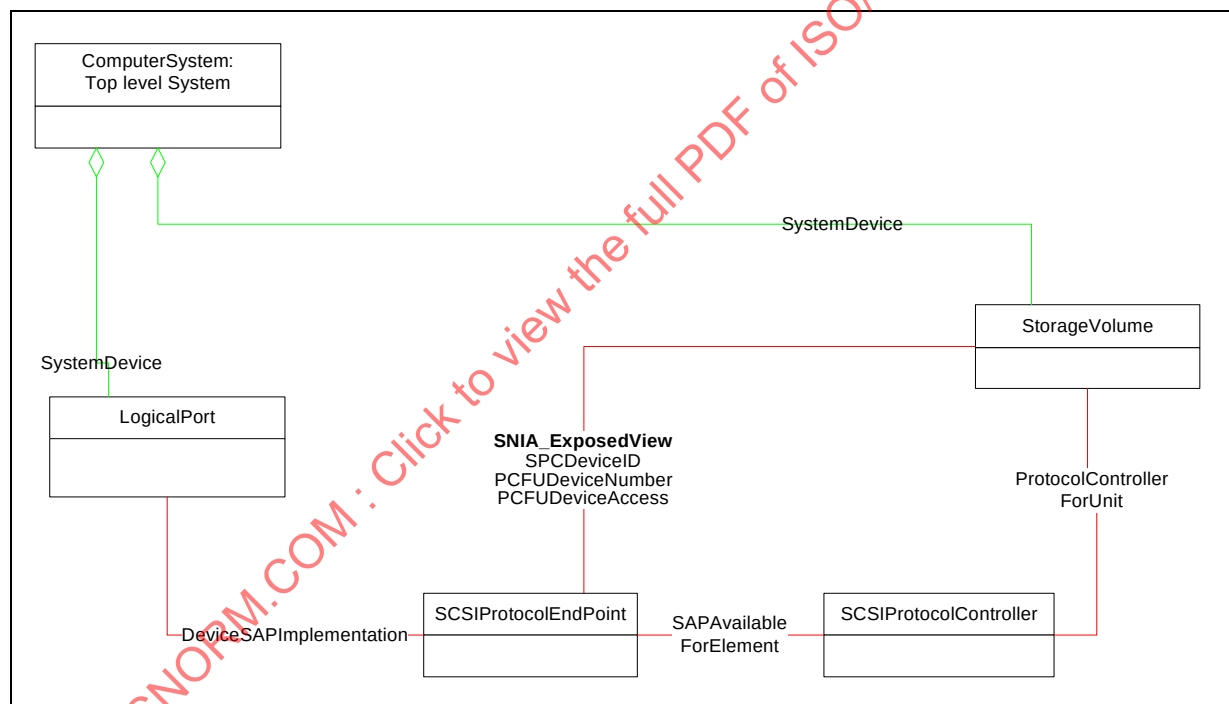


Figure 29 - SNIA_ExposedView Association

The SNIA_ExposedView association is composed of information drawn from the following base classes:

- SCSIProtocolController
- SAPAvailableForElement
- ProtocolControllerForUnit

The keys for the SNIA_ExposedView are the references to the LogicalDevice (a StorageVolume) and the reference to the SCSIProtocolEndpoint. There will be one instance of SNIA_ExposedView for each unique combination of StorageVolume and SCSIProtocolEndpoint through which the volume is exposed (in a Masking and Mapping model).

7.1.4.4.1.2 Mandatory, Conditional and Optional Properties of SNIA_ExposedView Association

In addition to the references to StorageVolume and the SCSIProtocolEndpoint the SNIA_ExposedView association also carries the DeviceID of the SCSIProtocolController and the DeviceNumber and DeviceAccess properties from the ProtocolControllerForUnit association.

In this release of SMI-S, the SNIA_ExposedView is "read only." It would be used to do association traversal from StorageVolumes to SCSIProtocolEndpoints that expose the Volumes.

7.1.4.4.2 SNIA_MaskingMapView Association

7.1.4.4.2.1 General

Figure 30 illustrates the SNIA_MaskingMapView Association.

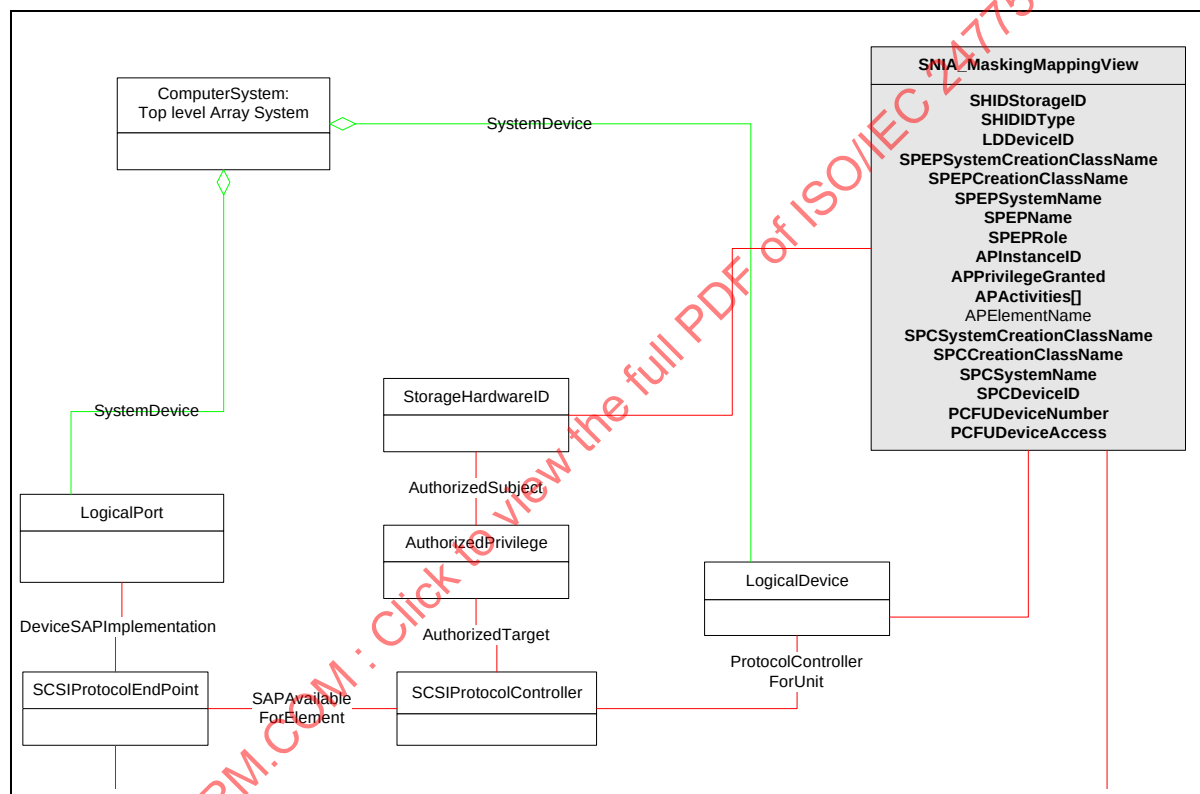


Figure 30 - SNIA_MaskingMapView Association

The SNIA_MaskingMapView association is a three way association that is composed of information drawn from the following base classes:

- StorageHardwareID
- AuthorizedPrivilege
- SCSIProtocolController
- SCSIProtocolEndpoint
- ProtocolControllerForUnit
- LogicalDevice

The keys for the SNIA_MaskingMapView are the SHID reference, the SCSIProtocolEndpoint reference and the LogicalDevice reference. There will be one instance

of SNIA_MaskingMapView for each unique combination of Storage Hardware ID (e.g., host), LogicalDevice (e.g., StorageVolume) and SCSIProtocolEndpoint (e.g., LogicalPort).

7.1.4.4.2 Mandatory, Conditional and Optional Properties of SNIA_MaskingMapView Association

In addition to the references to StorageHardwareID, LogicalDevice and the SCSIProtocolEndpoint the SNIA_MaskingMapView association also carries their properties and the AuthorizedPrivilege properties, DeviceID of the SCSIProtocolController and the DeviceNumber and DeviceAccess properties from the ProtocolControllerForUnit association. Also, for the convenience to clients, identifying properties from the LogicalDevice, StorageHardwareID and SCSIProtocolEndpoint are also pulled into the MaskingMapView. This allows a client to enumerate the SNIA_MaskingMapView association and get the identifiers for the endpoints in the association.

In this release of SMI-S, the SNIA_MaskingMapView is "read only." It would be used to do associate the StorageHardwareIDs, StorageVolumes to SCSIProtocolEndpoints.

EXPERIMENTAL

7.1.4.4.3 SNIA_MappingProtocolControllerView

Figure 31 illustrates the elements involved in supporting the SNIA_MappingProtocolControllerView.

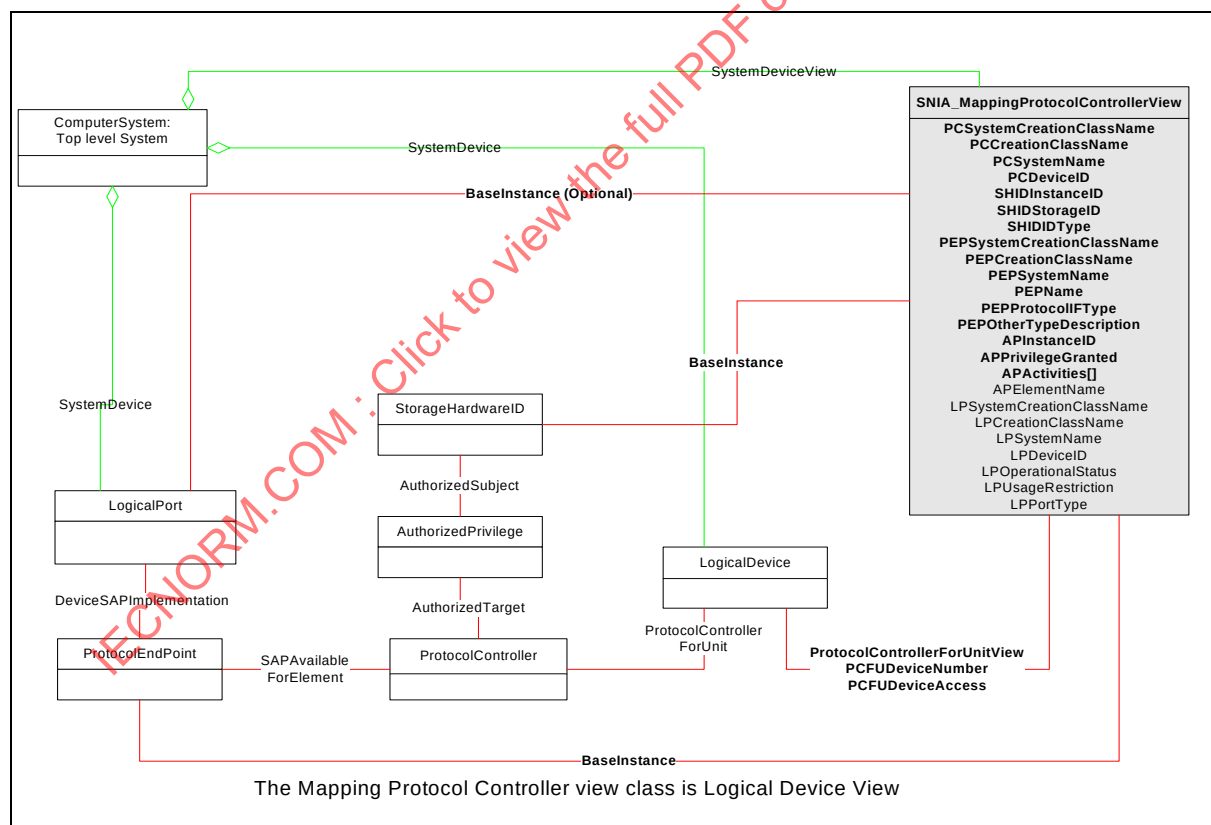


Figure 31 - The SNIA_MappingProtocolControllerView

The SNIA_MappingProtocolControllerView is composed of information drawn from the following base classes:

- LogicalPort
- ProtocolEndPoint

- ProtocolController
- AuthorizedPrivilege
- StorageHardwareID

The keys for the SNIA_MappingProtocolControllerView are the keys of the ProtocolEndpoint, ProtocolController and StorageHardwareID base classes. There will be one instance of SNIA_MappingProtocolControllerView for each unique combination of those keys.

7.1.4.4.4 Mandatory, Conditional and Optional Properties of SNIA_MappingProtocolControllerView

The properties from base classes shall be supported, but may be null. Properties that are mandatory in mandatory base classes are mandatory in the SNIA_MappingProtocolControllerView class. Properties that are conditional in a base class are conditional in the SNIA_MappingProtocolControllerView class.

Properties in the base classes that are optional in the base class are optional in the SNIA_MappingProtocolControllerView.

7.1.4.4.5 Associations on SNIA_MappingProtocolControllerView

7.1.4.4.5.1 General

In this release of SMI-S, the SNIA_MappingProtocolControllerView is "read only." In order to support update of information in the SNIA_MappingProtocolControllerView instance, it would be necessary to update the class instances on which it is based. An association SNIA_BaseInstance is provided to the CIM_StorageHardwareID, CIM_LogicalPort and CIM_ProtocolEndpoint instances.

NOTE The SNIA_BaseInstance association is only provided to base instances that can be modified.

In addition to the SNIA_MappingProtocolControllerView there are 2 associations that support association traversal to (or from) instances of the SNIA_MappingProtocolControllerView:

7.1.4.4.5.2 SNIA_ProtocolControllerForUnitView (mandatory if the MappingProtocolControllerView is implemented)

From a MappingProtocolControllerView instance a client will be able to find the CIM_LogicalDevices associated to the MappingProtocolControllerView (ProtocolController) via the SNIA_ProtocolControllerForUnitView. This will return the LogicalDevice instances that correspond to the ProtocolController of the MappingProtocolControllerView that would be found via association traversal from the ProtocolController to the LogicalDevices via CIM_ProtocolControllerForUnit association.

7.1.4.4.5.3 SNIA_SystemDeviceView (mandatory if the MappingProtocolControllerView is implemented)

From the owning CIM_ComputerSystem a client will be able to find the SNIA_MappingProtocolControllerViews associated to the ComputerSystem via the SNIA_SystemDeviceView. This will return the MappingProtocolControllerViews that correspond to the CIM_ProtocolController instances that would be found via association traversal from the ComputerSystem to the CIM_ProtocolController instances via CIM_SystemDevice.

Similarly, if the client has a SNIA_MappingProtocolControllerView instance, the client can find the owning ComputerSystem by following the SNIA_SystemDeviceView association from the SNIA_MappingProtocolControllerView instance to the CIM_ComputerSystem instance for the ComputerSystem that scopes the CIM_ProtocolController instances.

7.1.4.5 Storage Pool Views

7.1.4.5.1 SNIA_StoragePoolView

Figure 32 illustrates the elements involved in supporting the SNIA_StoragePoolView.

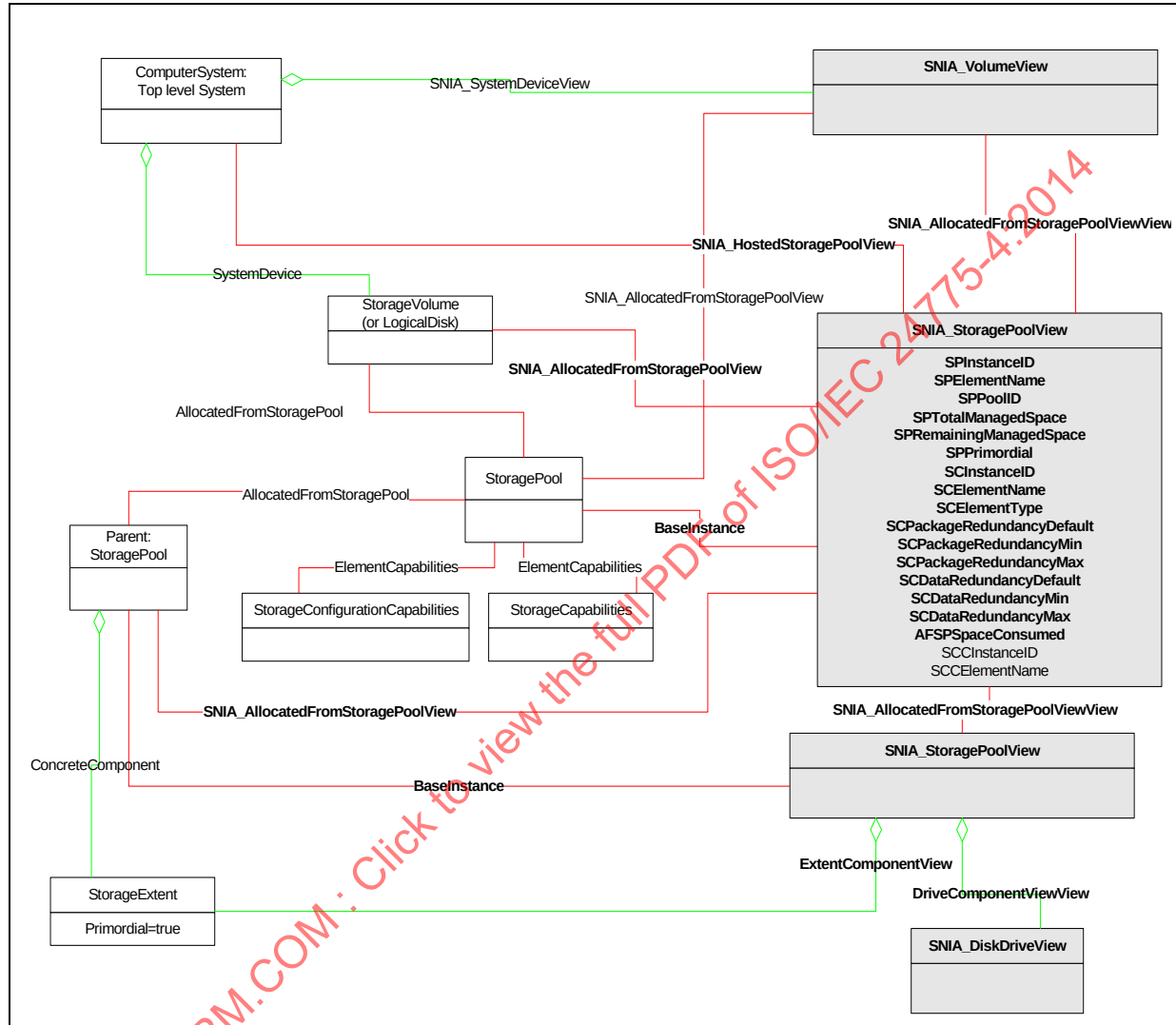


Figure 32 - The SNIA_StoragePoolView

The SNIA_StoragePoolView is composed of information drawn from the following base classes:

- StoragePool
- StorageCapabilities
- StorageConfigurationCapabilities (Optional)
- AllocatedFromStoragePool

The keys for the SNIA_StoragePoolView are the keys of the StoragePool base class. There will be one instance of SNIA_StoragePoolView for each instance of a StoragePool.

7.1.4.5.2 Mandatory, Conditional and Optional Properties of SNIA_StoragePoolView

The properties from base classes shall be supported, but may be null. Properties that are mandatory in mandatory base classes are mandatory in the SNIA_StoragePoolView class.

Properties that are conditional in a base class are conditional in the SNIA_StoragePoolView class. Properties that are mandatory in optional (base) classes (e.g., StorageConfigurationCapabilities) are "conditional" in the SNIA_StoragePoolView. If an optional base class is not supported by the implementation, these properties of those classes shall be present but shall be null.

Properties in the base classes that are optional in the base class are optional in the SNIA_StoragePoolView.

7.1.4.5.3 Associations on SNIA_StoragePoolView

7.1.4.5.3.1 General

In this release of SMI-S, the SNIA_StoragePoolView is "read only." In order to support update of information in the SNIA_StoragePoolView instance, it would be necessary to update the class instances on which it is based. An association SNIA_BaseInstance is provided to the CIM_StoragePool instance.

NOTE The SNIA_BaseInstance association is only provided to base instances that can be modified.

In addition to the SNIA_StoragePoolView there are 7 associations that support association traversal to (or from) instances of the SNIA_StoragePoolView.

7.1.4.5.3.2 SNIA_AllocatedFromStoragePoolView (StoragePoolView to StoragePool)

This association is mandatory if the StoragePoolView is implemented.

From a SNIA_StoragePoolView instance, the client can find the parent StoragePool to which the pool is allocated from by following the SNIA_AllocatedFromStoragePoolView association from the SNIA_StoragePoolView instance to the CIM_StoragePool instance for the StoragePool.

Similarly, if the client has a CIM_StoragePool instance a client will be able to find the SNIA_StoragePoolViews that are allocated from the StoragePool via the SNIA_AllocatedFromStoragePoolView. This will return the StoragePoolView instances that correspond to the StoragePools that would be found via association traversal from the StoragePool to the StoragePool via the CIM_AllocatedFromStoragePool association.

7.1.4.5.3.3 SNIA_AllocatedFromStoragePoolView (Volume to StoragePoolView)

This association is mandatory if the StoragePoolView is implemented.

From a CIM_StorageVolume (or CIM_LogicalDisk) instance, the client can find the StoragePoolView that the volume is allocated from by following the SNIA_AllocatedFromStoragePoolView association from the CIM class (StorageVolume or LogicalDisk) to the appropriate SNIA_StoragePoolView instance that corresponds to the CIM_StoragePool instance the volume is allocated from.

Similarly, if the client has a SNIA_StoragePoolView instance, the client will be able to find the CIM_StorageVolumes (or CIM_LogicalDisks) that are allocated from that StoragePoolView by following the SNIA_AllocatedFromStoragePoolView association.

7.1.4.5.3.4 SNIA_AllocatedFromStoragePoolViewView (VolumeView to StoragePoolView)

This association is mandatory if the StoragePoolView and the VolumeView are implemented.

From a SNIA_VolumeView instance, the client can find the StoragePoolView that the volume is allocated from by following the SNIA_AllocatedFromStoragePoolViewView association from the SNIA_VolumeView instance to the appropriate SNIA_StoragePoolView instance that corresponds to the CIM_StoragePool instance the volume is allocated from.

Similarly, if the client has a SNIA_StoragePoolView instance, the client will be able to find the SNIA_VolumeViews for volumes that are allocated from that StoragePoolView by following the SNIA_AllocatedFromStoragePoolViewView association.

7.1.4.5.3.5 SNIA_AllocatedFromStoragePoolViewView (StoragePoolView to StoragePoolView)

This association is mandatory if the StoragePoolView is implemented.

From a SNIA_StoragePoolView instance, the client can find the parent StoragePoolView to which the pool is allocated from by following the SNIA_AllocatedFromStoragePoolViewView association from the SNIA_StoragePoolView instance to the SNIA_StoragePoolView instance for the parent StoragePool.

Similarly, if the client has a SNIA_StoragePoolView instance a client will be able to find the SNIA_StoragePoolViews that are allocated from the StoragePool via the SNIA_AllocatedFromStoragePoolViewView. This will return the StoragePoolView instances that correspond to the StoragePools that would be found via association traversal from the StoragePool to the StoragePool via the CIM_AllocatedFromStoragePool association.

7.1.4.5.3.6 SNIA_HostedStoragePoolView

This is mandatory if the StoragePoolView is implemented.

From the owning CIM_ComputerSystem a client will be able to find the SNIA_StoragePoolViews associated to the ComputerSystem via the SNIA_HostedStoragePoolView. This will return the StoragePoolViews that correspond to the CIM_StoragePool instances that would be found via association traversal from the ComputerSystem to the CIM_StoragePool instances via CIM_HostedStoragePool.

Similarly, if the client has a SNIA_StoragePoolView instance, the client can find the owning ComputerSystem by following the SNIA_HostedStoragePoolView association from the SNIA_StoragePoolView instance to the CIM_ComputerSystem instance for the ComputerSystem that scopes the CIM_StoragePool instances.

7.1.4.5.3.7 ExtentComponentView

This is mandatory if the StoragePoolView is implemented.

From a SNIA_StoragePoolView instance, the client can find the pool component CIM_StorageExtent instances for the extents that form the pool via the SNIA_ExtentComponentView. This will return the StorageExtents that correspond to the SNIA_StoragePoolView instances that would be found via association traversal from the CIM_StoragePool instance to CIM_StorageExtent instances via CIM_ConcreteComponent.

Similarly, if the client has a CIM_StorageExtent instance, the client can find the SNIA_StoragePoolView by following the SNIA_ExtentComponentView association from the CIM_StorageExtent instance to the SNIA_StoragePoolView instance for the storage pool that has the CIM_StorageExtent as a pool component.

7.1.4.5.3.8 DriveComponentViewView

This association is mandatory if the StoragePoolView and the DiskDriveView are implemented.

From a SNIA_StoragePoolView instance, the client will be able to find the SNIA_DiskDriveViews for drives that are components of that StoragePoolView by following the SNIA_DriveComponentViewView association.

Similarly, if the client has a DiskDriveView instance, the client can find the StoragePoolView that the drive is a component of by following the SNIA_DriveComponentViewView association from the SNIA_DiskDriveView instance to the appropriate SNIA_StoragePoolView instance that corresponds to the CIM_StoragePool instance the drive is a component of.

7.1.4.6 Replication Views

7.1.4.6.1 SNIA_ReplicaPairView

Figure 33 illustrates the elements involved in supporting the SNIA_ReplicaPairView.

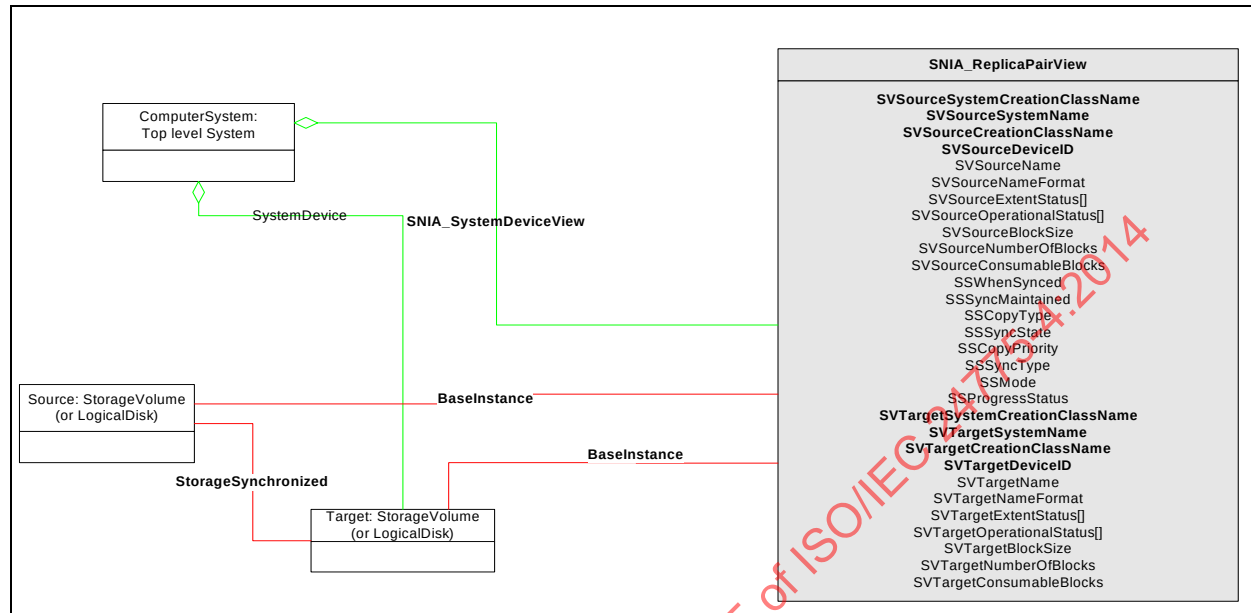


Figure 33 - The SNIA_ReplicaPairView

The SNIA_ReplicaPairView is composed of information drawn from the following base classes:

- StorageVolume (or LogicalDisk) for the Target
- StorageVolume (or LogicalDisk) for the Source
- StorageSynchronized

The keys for the SNIA_ReplicaPairView are the keys of the target StorageVolume (or LogicalDisk) base class. There will be one instance of SNIA_ReplicaPairView for each instance of a target StorageVolume (or LogicalDisk).

7.1.4.6.2 Mandatory, Conditional and Optional Properties of SNIA_ReplicaPairView

The properties from base classes shall be supported, but may be null. Properties that are mandatory in mandatory base classes are mandatory in the SNIA_ReplicaPairView class. Properties that are conditional in a base class are conditional in the SNIA_ReplicaPairView class.

Properties in the base classes that are optional in the base class are optional in the SNIA_ReplicaPairView.

7.1.4.6.3 Associations on SNIA_ReplicaPairView

7.1.4.6.3.1 General

In this release of SMI-S, the SNIA_ReplicaPairView is "read only." In order to support update of information in the SNIA_ReplicaPairView instance, it would be necessary to update the class instances on which it is based. An association SNIA_BaseInstance is provided to the CIM_StorageVolume instances (both source and target).

NOTE The SNIA_BaseInstance association is only provided to base instances that can be modified.

In addition to the SNIA_ReplicaPairView there is only one association that support association traversal to (or from) instances of the SNIA_ReplicaPairView.

7.1.4.6.3.2 SNIA_SystemDeviceView (ReplicaPairViews)

This is mandatory if the ReplicaPairView is implemented.

From the owning CIM_ComputerSystem a client will be able to find the SNIA_ReplicaPairViews associated to the ComputerSystem via the SNIA_SystemDeviceView. This will return the ReplicaPairView instances that correspond to the CIM_StorageVolume (CIM_LogicalDisk) instances of target volumes that would be found via association traversal from the ComputerSystem to the CIM_StorageVolume (or CIM_LogicalDisk) instances via CIM_SystemDevice.

Similarly, if the client has a SNIA_ReplicaPairView instance, the client can find the owning ComputerSystem by following the SNIA_SystemDeviceView association from the SNIA_ReplicaPairView instance to the CIM_ComputerSystem instance for the ComputerSystem that scopes the CIM_StorageVolume (CIM_LogicalDisk) instances.

EXPERIMENTAL

7.2 Health and Fault Management Consideration

Health and Fault Management considerations are defined in terms of the base classes (no View Classes). However, it should be noted that OperationalStatus of view classes shall be the same as the OperationalStatus of the underlying CIM classes on which the view classes are defined.

7.3 Cascading Considerations

Not defined in this International Standard.

7.4 Supported Profiles, Subprofiles, and Packages

See 7.1.1.1.

7.5 Methods of the Profile

7.5.1 Extrinsic Methods of the Profile

None.

7.5.2 Intrinsic Methods of the Profile

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

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

SNIA_View classes are modified by creating, deleting and modifying the base classes from which they are derived. The property values of SNIA_View classes are derived from the property values of associated classes. This profile does not specify the means to modify, create, or delete those classes. The base class instances may be accessed from the view class instances via association traversal through the SNIA_BaseInstance association.

7.6 Client Considerations and Recipes

7.6.1 Use Cases

7.6.1.1 Discovery of the Volumes on an Array

Table 74 identifies the elements of the use case to discover the volumes on an Array.

Table 74 - Discovery of the Volumes on an Array

Use Case Element	Description
Summary	Given an Array ComputerSystem, find the volumes (and their relevant information) on the system.
Basic Course of Events	1. Find the top level system of an array (using ElementConformsToProfile). 2. Find the related Volumes (on that system, using SystemDeviceView). 3. Locate the Component ComputerSystems (using ComponentCS). 4. Find the related Volumes on each of those systems (using SystemDeviceView).
Alternative Paths	None.
Exception Paths	None.
Triggers	Need to build or refresh a topology database for an Array.
Assumptions	None.
Preconditions	The Array provider has implemented the Block Storage Views Profile and SNIA_ViewCapabilities.SupportedViews contains "VolumeView".
Postconditions	Administrator has all Volumes, their Settings and what Pools they are allocated from.

7.6.1.2 Discovery of the Disk Drives in a Primordial Pool

Table 75 identifies the elements of the use case to discover the Disk Drives in a Primordial Pool.

Table 75 - Discovery of the Disk Drives in a Primordial Pool

Use Case Element	Description
Summary	Given an Array Primordial Pool, find the Disk Drives (and their information) that are its components.
Basic Course of Events	1. Find the related Disk Drives (in that pool, using ConcreteComponentView).
Alternative Paths	1a. Find all the disk drives on the system (using SystemDeviceView)
Exception Paths	None.
Triggers	Need to build or refresh the Drive topology database for an Array
Assumptions	None.
Preconditions	The Array provider has implemented the Block Storage Views Profile and SNIA_ViewCapabilities.SupportedViews contains "DiskDriveView".
Postconditions	Administrator has all DiskDrives and related information (scoped by the Pool or System).

7.6.1.3 Discover Volumes exposed on a (Target) Port

Table 76 identifies the elements of the use case to Discover Volumes exposed on a (Target) Port.

Table 76 - Discover Volumes exposed on a (Target) Port

Use Case Element	Description
Summary	Given an Array target port, find the volumes that are exposed through that port.
Basic Course of Events	1. Find the ProtocolEndpoint(s) associated to the Port (using DSI)2. Find the related Volumes (on that system, using ExposedView)
Alternative Paths	None.
Exception Paths	None.
Triggers	Determine Volumes accessible through a port on an Array.
Assumptions	None.
Preconditions	The Array provider has implemented the Block Storage Views Profile and SNIA_ViewCapabilities.SupportedViews contains "ExposedView".
Postconditions	Administrator has all Volumes that depend on the port for access.

7.6.1.4 Discover (target port) redundancy for a Volume

Table 77 identifies the elements of the use case to discover (target port) redundancy for a Volume.

Table 77 - Discover (target port) redundancy for a Volume

Use Case Element	Description
Summary	Given an Array volume, find the target ports through which it can be accessed.
Basic Course of Events	1. Find the ProtocolEndpoints that support the volume (using ExposedView). 2. Find the related target Ports (using DSI).
Alternative Paths	None.
Exception Paths	None.
Triggers	Need to determine what target ports are available for accessing a volume.
Assumptions	None.
Preconditions	The Array provider has implemented the Block Storage Views Profile and SNIA_ViewCapabilities.SupportedViews contains "ExposedView".
Postconditions	Administrator has the ports through which the volume may be accessed.

7.6.1.5 Discover Volumes exposed to a Host Port

Table 78 identifies the elements of the use case to discover Volumes exposed to a Host Port.

Table 78 - Discover Volumes exposed to a Host Port

Use Case Element	Description
Summary	Given an host port (Storage HardwareID), find the volumes that are mapped to that host port.
Basic Course of Events	1. Find the Volumes mapped to the host port (MaskingMapView).
Alternative Paths	None.

Table 78 - Discover Volumes exposed to a Host Port (Continued)

Use Case Element	Description
Exception Paths	None.
Triggers	Need to build or refresh a topology database for host access to Array volumes.
Assumptions	None.
Preconditions	The Array provider has implemented the Block Storage Views Profile and SNIA_ViewCapabilities.SupportedViews contains "MappingMaskingView".
Postconditions	Administrator has all Volumes that are mapped to the host port.

7.6.1.6 Discover the Mapping information for an array

Table 79 identifies the elements of the use case to discover the Mapping information for an array.

Table 79 - Discover Mapping information for an array

Use Case Element	Description
Summary	Given an Array ComputerSystem, find the masking and mapping information.
Basic Course of Events	1. Find the target ports and host ports that are connected (Using SystemDeviceView to MappingProtocolControllerView). 2. Find the Volumes for a ProtocolController (using ProtocolControllerForUnitView).
Alternative Paths	None.
Exception Paths	None.
Triggers	Need to build or refresh a topology database for masking and mapping information for an Array.
Assumptions	None.
Preconditions	The Array provider has implemented the Block Storage Views Profile and SNIA_ViewCapabilities.SupportedViews contains "MappingProtocolControllerView".
Postconditions	Administrator has all the Masking and Mapping information.

7.6.1.7 Discover the Pool topology for an array

Table 80 identifies the elements of the use case to discover the Pool topology for an array.

Table 80 - Discover the Pool topology for an array

Use Case Element	Description
Summary	Given an Array ComputerSystem, find the Pools on the system.
Basic Course of Events	1. Find the Pools and their capabilities for the system (Using HostedPoolView).
Alternative Paths	None.
Exception Paths	None.
Triggers	Need to build or refresh a topology database for pools in an Array.
Assumptions	None.
Preconditions	The Array provider has implemented the Block Storage Views Profile and SNIA_ViewCapabilities.SupportedViews contains "StoragePoolView".

Table 80 - Discover the Pool topology for an array (Continued)

Use Case Element	Description
Postconditions	Administrator has all the Pools and their capabilities information.

7.6.1.8 Discover the Replica Pairs for an array

Table 81 identifies the elements of the use case to discover the Replica Pairs for an array.

Table 81 - Discover the Replica Pairs for an array

Use Case Element	Description
Summary	Given an Array ComputerSystem, find the Replica Pairs on the system.
Basic Course of Events	1. Find the volume pairs for pairs on the array (Using SystemDeviceView to ReplicaPairView).
Alternative Paths	None.
Exception Paths	None.
Triggers	Need to build or refresh a topology database for Replicas in an Array.
Assumptions	None.
Preconditions	The Array provider has implemented the Block Storage Views Profile and SNIA_ViewCapabilities.SupportedViews contains "ReplicaPairView".
Postconditions	Administrator has all the Replicas that are defined in the Array.

7.6.2 Recipes

Not supported in this International Standard.

7.7 CIM Elements

7.7.1 General

Table 82 describes the CIM elements for Block Storage Views.

Table 82 - CIM Elements for Block Storage Views

Element Name	Requirement	Description
7.7.2 CIM_ElementCapabilities (View Capabilities)	Mandatory	Associates the top level ComputerSystem to the SNIA_ViewCapabilities supported by the implementation.
7.7.3 SNIA_AllocatedFromStoragePoolView (StoragePoolView to StoragePool)	Conditional	Conditional requirement: Required if the array property SNIA_ViewCapabilities.SupportedViews contains the string "StoragePoolView" (and the Block Service Package is implemented). This associates a SNIA_StoragePoolView instance to a CIM_StoragePool instance. This is required if the SNIA_StoragePoolView is implemented.
7.7.4 SNIA_AllocatedFromStoragePoolView (Volume to StoragePoolView)	Conditional	Conditional requirement: Required if the array property SNIA_ViewCapabilities.SupportedViews contains the string "StoragePoolView" (and the Block Service Package is implemented). This associates a CIM_StorageVolume (or CIM_LogicalDisk) instance to a SNIA_StoragePoolView. This is required if the SNIA_StoragePoolView is implemented.

Table 82 - CIM Elements for Block Storage Views (Continued)

Element Name	Requirement	Description
7.7.5 SNIA_AllocatedFromStoragePoolView (VolumeView to StoragePool)	Conditional	Conditional requirement: Required if the array property SNIA_ViewCapabilities.SupportedViews contains the string "VolumeView" (and the Block Service Package is implemented). This associates a SNIA_VolumeView instance to a CIM_StoragePool. This is required if the SNIA_VolumeView is implemented.
7.7.6 SNIA_AllocatedFromStoragePoolViewView (PoolView to PoolView)	Conditional	Experimental. Conditional requirement: Required if the array property SNIA_ViewCapabilities.SupportedViews contains the string "StoragePoolView" (and the Block Service Package is implemented). This associates a SNIA_StoragePoolView instance to its parent SNIA_StoragePoolView instance that it is allocated from.
7.7.7 SNIA_AllocatedFromStoragePoolViewView (VolumeView to PoolView)	Conditional	Experimental. Conditional requirement: Required if the array property SNIA_ViewCapabilities.SupportedViews contains the strings "StoragePoolView" and "VolumeView" (and the Block Service Package is implemented). This associates a SNIA_VolumeView instance to a SNIA_StoragePoolView instance that volume is allocated from.
7.7.8 SNIA_BaseInstance (DiskDrive)	Conditional	Conditional requirement: Required if the array property SNIA_ViewCapabilities.SupportedViews contains the string "DiskDriveView" (and the Disk Drive Lite Profile is implemented). This associates a SNIA_DiskDriveView instance to a base CIM_DiskDrive instance that can be modified. This is required if the SNIA_DiskDriveView is implemented.
7.7.9 SNIA_BaseInstance (StorageSetting)	Conditional	Conditional requirement: Required if the array property SNIA_ViewCapabilities.SupportedViews contains the string "VolumeView" (and the Block Service Package is implemented). This associates a SNIA_VolumeView class instance to a base CIM_StorageSetting class instance that can be modified. This is required if the SNIA_VolumeView is implemented.
7.7.10 SNIA_BaseInstance (Volume)	Conditional	Conditional requirement: Required if the array property SNIA_ViewCapabilities.SupportedViews contains the string "VolumeView" (and the Block Service Package is implemented). This associates a SNIA_VolumeView instance to a base CIM_StorageVolume (or CIM_LogicalDisk) instance that can be modified. This is required if the SNIA_VolumeView is implemented.
7.7.11 SNIA_BasedOnView (ExtentOnDriveExtent)	Conditional	Conditional requirement: Required if the array property SNIA_ViewCapabilities.SupportedViews contains the string "DiskDriveView" and Extent Composition is implemented. This associates a concrete CIM_StorageExtent instance to a SNIA_DiskDriveView instance. This is required if the SNIA_DiskDriveView and ExtentComposition are implemented.

Table 82 - CIM Elements for Block Storage Views (Continued)

Element Name	Requirement	Description
7.7.12 SNIA_BasedOnView (VolumeOnExtent)	Conditional	Conditional requirement: Required if the array property SNIA_ViewCapabilities.SupportedViews contains the string "VolumeView" and Extent Composition is implemented. This associates a SNIA_VolumeView instance to a base CIM_StorageExtent instance on which the volume is based. This is required if the SNIA_VolumeView and ExtentComposition are implemented.
7.7.13 SNIA_ConcreteComponentView	Conditional	Conditional requirement: Required if the array property SNIA_ViewCapabilities.SupportedViews contains the string "DiskDriveView" (and the Disk Drive Lite Profile is implemented). The SNIA_ConcreteComponentView associates the SNIA_DiskDriveView instance to the primordial StoragePool to which the disk drive StorageExtent is assigned. This is required if the SNIA_DiskDriveView is implemented.
7.7.14 SNIA_ContainerView	Conditional	Conditional requirement: Required if the array property SNIA_ViewCapabilities.SupportedViews contains the string "DiskDriveView" (and the Disk Drive Lite Profile is implemented). The SNIA_ContainerView associates the SNIA_DiskDriveView instance to the higher level physical package (e.g., System physical package) that contains the physical package of the disk drive. This is required if the SNIA_DiskDriveView is implemented.
7.7.15 SNIA_DiskDriveView	Conditional	Conditional requirement: Required if the array property SNIA_ViewCapabilities.SupportedViews contains the string "DiskDriveView" (and the Disk Drive Lite Profile is implemented). The SNIA_DiskDriveView instance represents a Disk Drive and its associated information. This is required if SNIA_ViewCapabilities.SupportedViews includes "DiskDriveView".
7.7.16 SNIA_DriveComponentViewView	Conditional	Experimental. Conditional requirement: Required if the array property SNIA_ViewCapabilities.SupportedViews contains the string "StoragePoolView" and "DiskDriveView" (and the Disk Drive Lite Profile is implemented). This associates a SNIA_StoragePoolView instance to a SNIA_DiskDriveView instance that is a component of the StoragePool.
7.7.17 SNIA_ElementStatisticalDataView (DiskDriveView)	Conditional	Conditional requirement: Required if the array property SNIA_ViewCapabilities.SupportedViews contains the string "DiskDriveView", CIM_BlockStatisticsCapabilities.ElementTypesSupported contains "10" and Block Server Performance is implemented. This associates a SNIA_DiskDriveView instance to the CIM_BlockStorageStatisticalData instance for the Disk Drive. This is required if the SNIA_DiskDriveView and the Block Server Performance Subprofile are implemented.

Table 82 - CIM Elements for Block Storage Views (Continued)

Element Name	Requirement	Description
7.7.18 SNIA_ElementStatisticalDataView (VolumeView)	Conditional	Conditional requirement: Required if the array property SNIA_ViewCapabilities.SupportedViews contains the string "VolumeView", CIM_BlockStatisticsCapabilities.ElementTypesSupported contains "8" and Block Server Performance is implemented. This associates a SNIA_VolumeView instance to the CIM_BlockStorageStatisticalData instance for the StorageVolume (or LogicalDisk). This is required if the SNIA_VolumeView and the Block Server Performance Subprofile are implemented.
7.7.19 SNIA_ExposedView	Conditional	Conditional requirement: Required if the array property SNIA_ViewCapabilities.SupportedViews contains the string "ExposedView" (and the Masking and Mapping Profile is implemented). This view associates a Target SCSIProtocolEndpoint and a LogicalDevice (e.g., StorageVolume). This is required if the SNIA_ViewCapabilities.SupportedViews includes "ExposedView".
7.7.20 SNIA_ExtentComponentView	Conditional	Experimental. Conditional requirement: Required if the array property SNIA_ViewCapabilities.SupportedViews contains the string "StoragePoolView" (and the Block Service Package is implemented). This associates a SNIA_StoragePoolView instance to a CIM_StorageExtent instance that is a component of the StoragePool.
7.7.21 SNIA_HostedStoragePoolView	Conditional	Conditional requirement: Required if the array property SNIA_ViewCapabilities.SupportedViews contains the string "StoragePoolView" (and the Block Service Package is implemented). This associates a SNIA_StoragePoolView instance to the CIM_ComputerSystem instance that hosts the underlying StoragePool.
7.7.22 SNIA_MappingProtocolControllerView	Conditional	Experimental. Conditional requirement: Required if the array property SNIA_ViewCapabilities.SupportedViews contains the string "MappingProtocolControllerView" (and the Masking and Mapping Profile is implemented). The SNIA_MappingProtocolControllerView represents the unique pairing of Host Ports and TargetPorts as represented by a ProtocolController in the Masking and Mapping profile of a block storage profile. This is required if the SNIA_ViewCapabilities.SupportedViews includes "MappingProtocolControllerView".
7.7.23 SNIA_MaskingMapView	Conditional	Conditional requirement: Required if the array property SNIA_ViewCapabilities.SupportedViews contains the string "MaskingMapView" (and the Masking and Mapping Profile is implemented). This three way association associates a CIM_LogicalDevice, CIM_StorageHardwareID and CIM_SCSIProtocolEndpoint instances to each other and derived from the Masking and Mapping subprofile model. This is required if SNIA_ViewCapabilities.SupportedViews contains "MaskingMapView".

Table 82 - CIM Elements for Block Storage Views (Continued)

Element Name	Requirement	Description
7.7.24 SNIA_ProtocolControllerForUnitView	Conditional	Experimental. Conditional requirement: Required if the array property SNIA_ViewCapabilities.SupportedViews contains the string "MappingProtocolControllerView" (and the Masking and Mapping Profile is implemented). Associates an instance of MappingProtocolControllerView to a LogicalDevice.
7.7.25 SNIA_ReplicaPairView	Conditional	Experimental. Conditional requirement: Required if the array property SNIA_ViewCapabilities.SupportedViews contains the string "ReplicaPairView" (and the Copy Services Profile is implemented). A view that combines a source and target volume and the StorageSynchronized between them.
7.7.26 SNIA_StoragePoolView	Conditional	Experimental. Conditional requirement: Required if the array property SNIA_ViewCapabilities.SupportedViews contains the string "StoragePoolView" (and the Block Service Package is implemented). A view that combines StoragePool information with the StorageCapabilities and StorageConfigurationCapabilities for the StoragePool, as well as SpaceConsumed on its parent pool.
7.7.27 SNIA_SystemDeviceView (DiskDriveViews)	Conditional	Conditional requirement: Required if the array property SNIA_ViewCapabilities.SupportedViews contains the string "DiskDriveView" (and the Disk Drive Lite Profile is implemented). This association links SNIA_DiskDriveView instances to the scoping system. This is required if the SNIA_DiskDriveView is implemented.
7.7.28 SNIA_SystemDeviceView (MappingProtocolControllerViews)	Conditional	Conditional requirement: Required if the array property SNIA_ViewCapabilities.SupportedViews contains the string "MappingProtocolControllerView" (and the Masking and Mapping Profile is implemented). This association links SNIA_MappingProtocolControllerView instances to the scoping system. This is required if the SNIA_MappingProtocolControllerView is implemented.
7.7.29 SNIA_SystemDeviceView (ReplicaPairViews)	Conditional	Conditional requirement: Required if the array property SNIA_ViewCapabilities.SupportedViews contains the string "ReplicaPairView" (and the Copy Services Profile is implemented). This association links SNIA_ReplicaPairView instances to the scoping system. This is required if the SNIA_ReplicaPairView is implemented.

Table 82 - CIM Elements for Block Storage Views (Continued)

Element Name	Requirement	Description
7.7.30 SNIA_SystemDeviceView (VolumeViews)	Conditional	Conditional requirement: Required if the array property SNIA_ViewCapabilities.SupportedViews contains the string "VolumeView" (and the Block Service Package is implemented). This association links SNIA_VolumeView instances to the scoping system. This is required if the SNIA_VolumeView is implemented.
7.7.31 SNIA_ViewCapabilities	Mandatory	The SNIA_ViewCapabilities identifies the capabilities of the implementation of view classes.
7.7.32 SNIA_VolumeView	Conditional	Conditional requirement: Required if the array property SNIA_ViewCapabilities.SupportedViews contains the string "VolumeView" (and the Block Service Package is implemented). The SNIA_VolumeView represents the storage (LogicalDisks or StorageVolumes) of a block storage profile. This is required if the SNIA_ViewCapabilities.SupportedViews includes "VolumeView".

7.7.2 CIM_ElementCapabilities (View Capabilities)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 83 describes class CIM_ElementCapabilities (View Capabilities).

Table 83 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (View Capabilities)

Properties	Requirement	Description and Notes
Capabilities	Mandatory	The ViewCapabilities.
ManagedElement	Mandatory	The top level ComputerSystem that has the ViewCapabilities.

7.7.3 SNIA_AllocatedFromStoragePoolView (StoragePoolView to StoragePool)

The SNIA_AllocatedFromStoragePoolView instance is a view that is derived from the CIM_AllocatedFromStoragePool association between two StoragePools. Note that if the StoragePoolView is allocated from multiple StoragePools there will be multiple AllocatedFromStoragePoolView instances for the StoragePool. The SNIA_AllocatedFromStoragePoolView is not subclassed from anything.

Created By: External

Modified By: External

Deleted By: External

Requirement: Required if the array property SNIA_ViewCapabilities.SupportedViews contains the string "StoragePoolView" (and the Block Service Package is implemented).

Table 84 describes class SNIA_AllocatedFromStoragePoolView (StoragePoolView to StoragePool).

Table 84 - SMI Referenced Properties/Methods for SNIA_AllocatedFromStoragePoolView (StoragePoolView to StoragePool)

Properties	Requirement	Description and Notes
AFSPSpaceConsumed	Mandatory	The space consumed from the StoragePool by the StoragePoolView. This value is the same as the AllocatedFromStoragePool.SpaceConsumed value for the base CIM_StoragePool on the antecedent StoragePool.
Antecedent	Mandatory	The parent(s) StoragePool(s) from which the StoragePoolView is allocated.
Dependent	Mandatory	The CIM_StorageVolume or CIM_LogicalDisk instance that is allocated from the StoragePoolView. There is only one CIM_StorageVolume (or CIM_LogicalDisk) instance for the combined StorageVolume (or LogicalDisk) - StoragePool pair.

7.7.4 SNIA_AllocatedFromStoragePoolView (Volume to StoragePoolView)

The SNIA_AllocatedFromStoragePoolView instance is a view that is derived from the CIM_AllocatedFromStoragePool association between the StorageVolume or LogicalDisk (of the CIM_StorageVolume or CIM_LogicalDisk) and the StoragePoolView from which the StorageVolume (or LogicalDisk) is allocated. Note that if the StorageVolume (or LogicalDisk) is allocated from multiple StoragePools there will be multiple AllocatedFromStoragePoolView instances for the StorageVolume (or LogicalDisk). The SNIA_AllocatedFromStoragePoolView is not subclassed from anything.

Created By: External

Modified By: External

Deleted By: External

Requirement: Required if the array property SNIA_ViewCapabilities.SupportedViews contains the string "StoragePoolView" (and the Block Service Package is implemented).

Table 85 describes class SNIA_AllocatedFromStoragePoolView (Volume to StoragePoolView).

Table 85 - SMI Referenced Properties/Methods for SNIA_AllocatedFromStoragePoolView (Volume to StoragePoolView)

Properties	Requirement	Description and Notes
AFSPSpaceConsumed	Mandatory	The space consumed from the StoragePoolView by the StorageVolume (or LogicalDisk). This value is the same as the AllocatedFromStoragePool.SpaceConsumed value for the base CIM_StorageVolume on the antecedent StoragePool.
Antecedent	Mandatory	A StoragePoolView from which the StorageVolume (or LogicalDisk) is allocated.
Dependent	Mandatory	The CIM_StorageVolume or CIM_LogicalDisk instance that is allocated from the StoragePoolView. There is only one CIM_StorageVolume (or CIM_LogicalDisk) instance for the combined StorageVolume (or LogicalDisk) - StoragePool pair.

7.7.5 SNIA_AllocatedFromStoragePoolView (VolumeView to StoragePool)

The SNIA_AllocatedFromStoragePoolView instance is a view that is derived from the CIM_AllocatedFromStoragePool association between the StorageVolume or LogicalDisk (of the

SNIA_VolumeView) and the StoragePool from which the StorageVolume (or LogicalDisk) is allocated. Note that if the StorageVolume (or LogicalDisk) is allocated from multiple StoragePools there will be multiple AllocatedFromStoragePoolView instances for the StorageVolume (or LogicalDisk). The SNIA_AllocatedFromStoragePoolView is not subclassed from anything.

Created By: External

Modified By: External

Deleted By: External

Requirement: Required if the array property SNIA_ViewCapabilities.SupportedViews contains the string "VolumeView" (and the Block Service Package is implemented).

Table 86 describes class SNIA_AllocatedFromStoragePoolView (VolumeView to StoragePool).

Table 86 - SMI Referenced Properties/Methods for SNIA_AllocatedFromStoragePoolView (VolumeView to StoragePool)

Properties	Requirement	Description and Notes
AFSPSpaceConsumed	Mandatory	The space consumed from the StoragePool by the StorageVolume (or LogicalDisk). This value is the same as the AllocatedFromStoragePool.SpaceConsumed value for the base CIM_StorageVolume on the antecedent StoragePool.
Antecedent	Mandatory	A StoragePool from which the StorageVolume of the SNIA_VolumeView is allocated.
Dependent	Mandatory	The SNIA_VolumeView instance that is allocated from the StoragePool. There is only one VolumeView instance for the combined StorageVolume (or LogicalDisk) - StoragePool pair.

7.7.6 SNIA_AllocatedFromStoragePoolViewView (PoolView to PoolView)

Experimental. This SNIA_AllocatedFromStoragePoolViewView is an association between a SNIA_StoragePoolView instances and the SNIA_StoragePoolView instance that they are allocated from. The SNIA_AllocatedFromStoragePoolViewView is not subclassed from anything.

Requirement: Required if the array property SNIA_ViewCapabilities.SupportedViews contains the string "StoragePoolView" (and the Block Service Package is implemented).

Table 87 describes class SNIA_AllocatedFromStoragePoolViewView (PoolView to PoolView).

Table 87 - SMI Referenced Properties/Methods for SNIA_AllocatedFromStoragePoolViewView (PoolView to PoolView)

Properties	Requirement	Description and Notes
AFSPSpaceConsumed	Mandatory	The space consumed from the StoragePoolView by the StoragePoolView. This value is the same as the AllocatedFromStoragePool.SpaceConsumed value for the base CIM_StoragePool on the antecedent StoragePool.
Dependent	Mandatory	The StoragePoolView instance that is allocated from the parent pool.
Antecedent	Mandatory	The StoragePoolView instance for a parent StoragePool.

7.7.7 SNIA_AllocatedFromStoragePoolViewView (VolumeView to PoolView)

Experimental. This SNIA_AllocatedFromStoragePoolViewView is an association between a SNIA_VolumeView instances and the SNIA_StoragePoolView instance that the Volume is allocated from. The SNIA_AllocatedFromStoragePoolViewView is not subclassed from anything.

Requirement: Required if the array property SNIA_ViewCapabilities.SupportedViews contains the strings "StoragePoolView" and "VolumeView" (and the Block Service Package is implemented).

Table 88 describes class SNIA_AllocatedFromStoragePoolViewView (VolumeView to PoolView).

Table 88 - SMI Referenced Properties/Methods for SNIA_AllocatedFromStoragePoolViewView (VolumeView to PoolView)

Properties	Requirement	Description and Notes
AFSPSpaceConsumed	Mandatory	The space consumed from the StoragePoolView by the VolumeView. This value is the same as the AllocatedFromStoragePool.SpaceConsumed value for the base CIM_StorageVolume (or CIM_LogicalDisk) on the antecedent StoragePool.
Dependent	Mandatory	The VolumeView instance that is allocated from the pool.
Antecedent	Mandatory	The StoragePoolView instance for a parent StoragePool.

7.7.8 SNIA_BaseInstance (DiskDrive)

The SNIA_BaseInstance instance is an association between a SNIA_DiskDriveView instance and a base CIM_DiskDrive instance on which the view is based. This association is provided to accommodate update operations on the base CIM_DiskDrive instances, since the properties cannot be updated in the view class. The SNIA_BaseInstance is subclassed from CIM_Dependency.

Created By: External

Modified By: External

Deleted By: External

Requirement: Required if the array property SNIA_ViewCapabilities.SupportedViews contains the string "DiskDriveView" (and the Disk Drive Lite Profile is implemented).

Table 89 describes class SNIA_BaseInstance (DiskDrive).

Table 89 - SMI Referenced Properties/Methods for SNIA_BaseInstance (DiskDrive)

Properties	Requirement	Description and Notes
Antecedent	Mandatory	The base CIM_DiskDrive instance on which the SNIA_DiskDriveView instance is based.
Dependent	Mandatory	The SNIA_DiskDriveView instance that is based on the CIM_DiskDrive instance.

7.7.9 SNIA_BaseInstance (StorageSetting)

The SNIA_BaseInstance instance is an association between the SNIA_VolumeView and the CIM_StorageSetting instance for the base StorageVolume (or LogicalDisk) on which the view is based. This association is provided to accommodate update operations on the CIM_StorageSetting instance (e.g., ModifyInstance), since the properties cannot be updated in the view class. The SNIA_BaseInstance is subclassed from CIM_Dependency.

Created By: External

Modified By: External

Deleted By: External

Requirement: Required if the array property SNIA_ViewCapabilities.SupportedViews contains the string "VolumeView" (and the Block Service Package is implemented).

Table 90 describes class SNIA_BaseInstance (StorageSetting).

Table 90 - SMI Referenced Properties/Methods for SNIA_BaseInstance (StorageSetting)

Properties	Requirement	Description and Notes
Antecedent	Mandatory	The base CIM_StorageSetting instance on which the SNIA_VolumeView instance is based.
Dependent	Mandatory	The SNIA_VolumeView instance that is based on the CIM_StorageSetting instance.

7.7.10 SNIA_BaseInstance (Volume)

The SNIA_BaseInstance instance is an association between a SNIA_VolumeView instance and a base CIM_StorageVolume (or CIM_LogicalDisk) instance on which the view is based. This association is provided to accommodate update operations on the base CIM_StorageVolume (or CIM_LogicalDisk) instances, since the properties cannot be updated in the view class. The SNIA_BaseInstance is subclassed from CIM_Dependency.

Created By: External

Modified By: External

Deleted By: External

Requirement: Required if the array property SNIA_ViewCapabilities.SupportedViews contains the string "VolumeView" (and the Block Service Package is implemented).

Table 91 describes class SNIA_BaseInstance (Volume).

Table 91 - SMI Referenced Properties/Methods for SNIA_BaseInstance (Volume)

Properties	Requirement	Description and Notes
Antecedent	Mandatory	The base CIM_StorageVolume (or CIM_LogicalDisk) instance on which the SNIA_VolumeView instance is based.
Dependent	Mandatory	The SNIA_VolumeView instance that is based on the CIM_StorageVolume (or CIM_LogicalDisk) instance.

7.7.11 SNIA_BasedOnView (ExtentOnDriveExtent)

The SNIA_BasedOnView instance is a view that is derived from CIM_BasedOn between a concrete CIM_StorageExtent instance and the primordial CIM_StorageExtent under it. The SNIA_BasedOnView is not subclassed from anything.

Created By: External

Modified By: External

Deleted By: External

Requirement: Required if the array property SNIA_ViewCapabilities.SupportedViews contains the string "DiskDriveView" and Extent Composition is implemented.

Table 92 describes class SNIA_BasedOnView (ExtentOnDriveExtent).

Table 92 - SMI Referenced Properties/Methods for SNIA_BasedOnView (ExtentOnDriveExtent)

Properties	Requirement	Description and Notes
BOStartingAddress	Optional	This is derived from the BasedOn.StartingAddress.
BOEndingAddress	Optional	This is derived from the BasedOn.EndingAddress.
BOOrderIndex	Optional	When the association is used in a concatenation composition, indicates the order in which the extents (and thus their block ranges) are concatenated.
Antecedent	Mandatory	The SNIA_DiskDriveView on which a concrete StorageExtent is based.
Dependent	Mandatory	The CIM_StorageExtent instance that is dependent on the SNIA_DiskDriveView.

7.7.12 SNIA_BasedOnView (VolumeOnExtent)

The SNIA_BasedOnView instance is a view that is derived from CIM_BasedOn between the CIM_StorageVolume instance and the first CIM_StorageExtent it is based on. The SNIA_BasedOnView is not subclassed from anything.

Created By: External

Modified By: External

Deleted By: External

Requirement: Required if the array property SNIA_ViewCapabilities.SupportedViews contains the string "VolumeView" and Extent Composition is implemented.

Table 93 describes class SNIA_BasedOnView (VolumeOnExtent).

Table 93 - SMI Referenced Properties/Methods for SNIA_BasedOnView (VolumeOnExtent)

Properties	Requirement	Description and Notes
BOStartingAddress	Optional	This is derived from the BasedOn.StartingAddress.
BOEndingAddress	Optional	This is derived from the BasedOn.EndingAddress.
BOOrderIndex	Optional	When the association is used in a concatenation composition, indicates the order in which the extents (and thus their block ranges) are concatenated.
Antecedent	Mandatory	The lower level StorageExtent on which the SNIA_VolumeView StorageVolume is based.
Dependent	Mandatory	The SNIA_VolumeView instance.

7.7.13 SNIA_ConcreteComponentView

The SNIA_ConcreteComponentView instance is a view that is derived from the CIM_ConcreteComponent between the base CIM_StorageExtent of the Disk Drive and its primordial CIM_StoragePool. The SNIA_ConcreteComponentView is not subclassed from anything.

Created By: External

Modified By: Static

Deleted By: External

Requirement: Required if the array property SNIA_ViewCapabilities.SupportedViews contains the string "DiskDriveView" (and the Disk Drive Lite Profile is implemented).

Table 94 describes class SNIA_ConcreteComponentView.

Table 94 - SMI Referenced Properties/Methods for SNIA_ConcreteComponentView

Properties	Requirement	Description and Notes
GroupComponent	Mandatory	The CIM_StoragePool to which the StorageExtent of the Disk Drive is assigned.
PartComponent	Mandatory	A SNIA_DiskDriveView instance that is assigned to the StoragePool.

7.7.14 SNIA_ContainerView

The SNIA_ContainerView instance is a view that is derived from the CIM_Container between the base CIM_PhysicalPackage of the Disk Drive and the CIM_PhysicalPackage of the ComputerSystem. The SNIA_ContainerView is not subclassed from anything.

Created By: External

Modified By: Static

Deleted By: External

Requirement: Required if the array property SNIA_ViewCapabilities.SupportedViews contains the string "DiskDriveView" (and the Disk Drive Lite Profile is implemented).

Table 95 describes class SNIA_ContainerView.

Table 95 - SMI Referenced Properties/Methods for SNIA_ContainerView

Properties	Requirement	Description and Notes
GroupComponent	Mandatory	The CIM_PhysicalPackage for the ComputerSystem instance that groups the CIM_PhysicalPackage of the Disk Drive.
PartComponent	Mandatory	A SNIA_DiskDriveView instance that includes CIM_PhysicalPackage information for the CIM_DiskDrive.

7.7.15 SNIA_DiskDriveView

The SNIA_DiskDriveView instance is a view that is derived from CIM_StorageExtent, CIM_MediaPresent, CIM_DiskDrive, CIM_Realizes, CIM_PhysicalPackage, CIM_ElementSoftwareIdentity and CIM_SoftwareIdentity. The SNIA_DiskDriveView is subclassed from CIM_ManagedElement.

Created By: External

Modified By: External

Deleted By: External

Requirement: Required if the array property SNIA_ViewCapabilities.SupportedViews contains the string "DiskDriveView" (and the Disk Drive Lite Profile is implemented).

Table 96 describes class SNIA_DiskDriveView.

Table 96 - SMI Referenced Properties/Methods for SNIA_DiskDriveView

Properties	Requirement	Description and Notes
SESystemCreationClassName	Mandatory	The SystemCreationClassName for the StorageExtent of the Disk Drive as reported in the underlying primordial StorageExtent instance for the Disk Drive.
SESystemName	Mandatory	The SystemName for the StorageExtent of the Disk Drive as reported in the underlying primordial StorageExtent instance for the Disk Drive.
SECreationClassName	Mandatory	The CreationClassName for the StorageExtent of the Disk Drive as reported in the underlying primordial StorageExtent instance for the Disk Drive.
SEDeviceID	Mandatory	The DeviceID for the StorageExtent of the Disk Drive as reported in the underlying primordial StorageExtent instance for the Disk Drive.
SEBlockSize	Mandatory	The BlockSize for the StorageExtent of the Disk Drive as reported in the underlying primordial StorageExtent instance for the Disk Drive.
SENumberOfBlocks	Mandatory	The NumberOfBlocks for the StorageExtent of the Disk Drive as reported in the underlying primordial StorageExtent instance for the Disk Drive.
SEConsumableBlocks	Mandatory	The ConsumableBlocks for the StorageExtent of the Disk Drive as reported in the underlying primordial StorageExtent instance for the Disk Drive.
SEExtentStatus	Mandatory	The ExtentStatus for the StorageExtent of the Disk Drive as reported in the underlying primordial StorageExtent instance for the Disk Drive.
SEOperationalStatus	Mandatory	The OperationalStatus for the StorageExtent of the Disk Drive as reported in the underlying primordial StorageExtent instance for the Disk Drive.
DDSystemCreationClassName	Mandatory	The SystemCreationClassName for the Disk Drive as reported in the underlying DiskDrive instance.
DDSystemName	Mandatory	The SystemName for the Disk Drive as reported in the underlying DiskDrive instance.
DDCreationClassName	Mandatory	The CreationClassName for the Disk Drive as reported in the underlying DiskDrive instance.
DDDeviceID	Mandatory	The DeviceID for the Disk Drive as reported in the underlying DiskDrive instance.
DDName	Mandatory	The Name for the Disk Drive as reported in the underlying DiskDrive instance.
DDOperationalStatus	Mandatory	The OperationalStatus for the Disk Drive as reported in the underlying DiskDrive instance.
PPCreationClassName	Mandatory	The CreationClassName for the PhysicalPackage of the Disk Drive as reported in the underlying PhysicalPackage instance for the Disk Drive.
PPTag	Mandatory	The Tag for the PhysicalPackage of the Disk Drive as reported in the underlying PhysicalPackage instance for the Disk Drive.
PPManufacturer	Mandatory	The Manufacturer for the PhysicalPackage of the Disk Drive as reported in the underlying PhysicalPackage instance for the Disk Drive.

Table 96 - SMI Referenced Properties/Methods for SNIA_DiskDriveView (Continued)

Properties	Requirement	Description and Notes
PPModel	Mandatory	The Model for the PhysicalPackage of the Disk Drive as reported in the underlying PhysicalPackage instance for the Disk Drive.
SIInstanceID	Mandatory	The InstanceID for the SoftwareIdentity of the Disk Drive as reported in the underlying SoftwareIdentity instance for the Disk Drive.
SIVersionString	Mandatory	The VersionString for the SoftwareIdentity of the Disk Drive as reported in the underlying SoftwareIdentity instance for the Disk Drive.
DDLocationIndicator	Optional	The LocationIndicator for the Disk Drive as reported in the underlying DiskDrive instance.
PPSerialNumber	Optional	The SerialNumber for the PhysicalPackage of the Disk Drive as reported in the underlying PhysicalPackage instance for the Disk Drive.
PPPartNumber	Optional	The PartNumber for the PhysicalPackage of the Disk Drive as reported in the underlying PhysicalPackage instance for the Disk Drive.
SIManufacturer	Optional	The Manufacturer for the SoftwareIdentity of the Disk Drive as reported in the underlying SoftwareIdentity instance for the Disk Drive.
SIBuildNumber	Optional	The BuildNumber for the SoftwareIdentity of the Disk Drive as reported in the underlying SoftwareIdentity instance for the Disk Drive.
SIMajorVersion	Optional	The MajorVersion for the SoftwareIdentity of the Disk Drive as reported in the underlying SoftwareIdentity instance for the Disk Drive.
SIRevisionNumber	Optional	The RevisionNumber for the SoftwareIdentity of the Disk Drive as reported in the underlying SoftwareIdentity instance for the Disk Drive.
SIMinorVersion	Optional	The MinorVersion for the SoftwareIdentity of the Disk Drive as reported in the underlying SoftwareIdentity instance for the Disk Drive.

7.7.16 SNIA_DriveComponentViewView

Experimental. The SNIA_DriveComponentViewView is an association between a SNIA_StoragePoolView instances and the SNIA_DiskDriveView instances for Disk Drives of the StoragePool. The SNIA_DriveComponentViewView is not subclassed from anything.

Requirement: Required if the array property SNIA_ViewCapabilities.SupportedViews contains the string "StoragePoolView" and "DiskDriveView" (and the Disk Drive Lite Profile is implemented).

Table 97 describes class SNIA_DriveComponentViewView.

Table 97 - SMI Referenced Properties/Methods for SNIA_DriveComponentViewView

Properties	Requirement	Description and Notes
PartComponent	Mandatory	The DiskDriveView instance.
GroupComponent	Mandatory	The StoragePoolView instance for a primordial StoragePool.

7.7.17 SNIA_ElementStatisticalDataView (DiskDriveView)

The SNIA_ElementStatisticalDataView is an association between a SNIA_DiskDriveView instance and the CIM_BlockStorageStatisticalData instance for the DiskDrive. The SNIA_BaseInstance is not subclassed from anything.

Created By: External

Modified By: External

Deleted By: External

Requirement: Required if the array property SNIA_ViewCapabilities.SupportedViews contains the string "DiskDriveView", CIM_BlockStatisticsCapabilities.ElementTypesSupported contains "10" and Block Server Performance is implemented.

Table 98 describes class SNIA_ElementStatisticalDataView (DiskDriveView).

Table 98 - SMI Referenced Properties/Methods for SNIA_ElementStatisticalDataView (DiskDriveView)

Properties	Requirement	Description and Notes
Stats	Mandatory	The CIM_BlockStorageStatisticalData instance for the DiskDrive (StorageExtent) instance.
ManagedElement	Mandatory	The SNIA_DiskDriveView instance that has the CIM_BlockStorageStatisticalData instance.

7.7.18 SNIA_ElementStatisticalDataView (VolumeView)

The SNIA_ElementStatisticalDataView is an association between a SNIA_VolumeView instance and the CIM_BlockStorageStatisticalData instance for the StorageVolume (or LogicalDisk). The SNIA_BaseInstance is not subclassed from anything.

Created By: External

Modified By: External

Deleted By: External

Requirement: Required if the array property SNIA_ViewCapabilities.SupportedViews contains the string "VolumeView", CIM_BlockStatisticsCapabilities.ElementTypesSupported contains "8" and Block Server Performance is implemented.

Table 99 describes class SNIA_ElementStatisticalDataView (VolumeView).

Table 99 - SMI Referenced Properties/Methods for SNIA_ElementStatisticalDataView (VolumeView)

Properties	Requirement	Description and Notes
Stats	Mandatory	The CIM_BlockStorageStatisticalData instance for the StorageVolume (or LogicalDisk) instance.
ManagedElement	Mandatory	The SNIA_VolumeView instance that has the CIM_BlockStorageStatisticalData instance.

7.7.19 SNIA_ExposedView

The SNIA_ExposedView instance is a view that is derived from CIM_SAPAvailableForElement, CIM_SCSIProtocolController and CIM_ProtocolControllerForUnit. The SNIA_ExposedView is not subclassed from anything.

Created By: External

Modified By: External

Deleted By: External

Requirement: Required if the array property SNIA_ViewCapabilities.SupportedViews contains the string "ExposedView" (and the Masking and Mapping Profile is implemented).

Table 100 describes class SNIA_ExposedView.

Table 100 - SMI Referenced Properties/Methods for SNIA_ExposedView

Properties	Requirement	Description and Notes
SPCSystemCreationClassName	Mandatory	The SystemCreationClassName for the SCSIProtocolController used with the underlying SCSIProtocolController instance for the SCSIProtocolEndpoint and StorageVolume.
SPCSystemName	Mandatory	The SystemName for the SCSIProtocolController used with the underlying SCSIProtocolController instance for the SCSIProtocolEndpoint and StorageVolume.
SPCCreationClassName	Mandatory	The CreationClassName for the SCSIProtocolController used with the underlying SCSIProtocolController instance for the SCSIProtocolEndpoint and StorageVolume.
SPCDeviceID	Mandatory	The DeviceID for the SCSIProtocolController used with the underlying SCSIProtocolController instance for the SCSIProtocolEndpoint and StorageVolume.
PCFUDeviceNumber	Mandatory	The DeviceNumber (LUN) for the StorageVolume when accessed through the SCSIProtocolEndpoint as reported in the underlying ProtocolControllerForUnit instance for the StorageVolume.
PCFUDeviceAccess	Mandatory	The DeviceAccess value for the StorageVolume when accessed through the SCSIProtocolEndpoint as reported in the underlying ProtocolControllerForUnit instance for the StorageVolume.
ProtocolEndpoint	Mandatory	The Target ProtocolEndpoint through which the LogicalDevice is exposed.
LogicalDevice	Mandatory	The LogicalDevice (e.g., StorageVolume) that is exposed through the Target ProtocolEndpoint.

7.7.20 SNIA_ExtentComponentView

Experimental. The SNIA_ExtentComponentView is an association between a SNIA_StoragePoolView instances and the CIM_StorageExtent instances for the StoragePool. The SNIA_ExtentComponentView is not subclassed from anything.

Requirement: Required if the array property SNIA_ViewCapabilities.SupportedViews contains the string "StoragePoolView" (and the Block Service Package is implemented).

Table 101 describes class SNIA_ExtentComponentView.

Table 101 - SMI Referenced Properties/Methods for SNIA_ExtentComponentView

Properties	Requirement	Description and Notes
PartComponent	Mandatory	A reference to a StorageExtent.
GroupComponent	Mandatory	A reference to a StoragePoolView instance that contains the Extent.

7.7.21 SNIA_HostedStoragePoolView

The SNIA_HostedStoragePoolView is an association between a SNIA_StoragePoolView instances and the CIM_ComputerSystem instance for the StoragePool. The SNIA_HostedStoragePoolView is not subclassed from anything.

Requirement: Required if the array property SNIA_ViewCapabilities.SupportedViews contains the string "StoragePoolView" (and the Block Service Package is implemented).

Table 102 describes class SNIA_HostedStoragePoolView.

Table 102 - SMI Referenced Properties/Methods for SNIA_HostedStoragePoolView

Properties	Requirement	Description and Notes
GroupComponent	Mandatory	The reference to the hosting computer system.
PartComponent	Mandatory	The reference to the hosted storage pool view.

7.7.22 SNIA_MappingProtocolControllerView

Experimental. The SNIA_MappingProtocolControllerView instance is a view that is derived from CIM_ProtocolController, CIM_StorageHardwareID, CIM_AuthorizedPrivilege, CIM_ProtocolEndPoint and CIM_LogicalPort, and their associations. The SNIA_MappingProtocolControllerView is subclassed from CIM_ManagedElement.

Created By: External

Modified By: External

Deleted By: External

Requirement: Required if the array property SNIA_ViewCapabilities.SupportedViews contains the string "MappingProtocolControllerView" (and the Masking and Mapping Profile is implemented).

Table 103 describes class SNIA_MappingProtocolControllerView.

Table 103 - SMI Referenced Properties/Methods for SNIA_MappingProtocolControllerView

Properties	Requirement	Description and Notes
PCSystemCreationClassName	Mandatory	The SystemCreationClassName as reported in the underlying ProtocolController.
PCCreationClassName	Mandatory	The CreationClassName as reported in the underlying ProtocolController.
PCSystemName	Mandatory	The SystemName as reported in the underlying ProtocolController.
PCDeviceID	Mandatory	The DeviceID as reported in the underlying ProtocolController.
SHIDInstanceID	Mandatory	The InstanceID as reported in the underlying StorageHardwareID.
SHIDStorageID	Mandatory	The StorageID as reported in the underlying StorageHardwareID.
SHIDIDType	Mandatory	The IDType as reported in the underlying StorageHardwareID.
PEPSystemCreationClassName	Mandatory	The SystemCreationClassName as reported in the underlying ProtocolEndpoint.
PEPCreationClassName	Mandatory	The CreationClassName as reported in the underlying ProtocolEndpoint.
PEPSystemName	Mandatory	The SystemName as reported in the underlying ProtocolEndpoint.
PEPName	Mandatory	The Name as reported in the underlying ProtocolEndpoint.
PEPProtocolIFType	Mandatory	The ProtocolIFType as reported in the underlying ProtocolEndpoint.

Table 103 - SMI Referenced Properties/Methods for SNIA_MappingProtocolControllerView

Properties	Requirement	Description and Notes
PEPOtherTypeDescription	Mandatory	The OtherTypeDescription as reported in the underlying ProtocolEndpoint.
PEPRole	Mandatory	The Role as reported in the underlying ProtocolEndpoint.
PEPConnectionType	Mandatory	The ConnectionType as reported in the underlying ProtocolEndpoint.
APIInstanceID	Mandatory	The InstanceID as reported in the underlying AuthorizedPrivilege.
APPrivilegeGranted	Mandatory	The PrivilegeGranted as reported in the underlying AuthorizedPrivilege.
APAActivities[]	Mandatory	The Activities[] as reported in the underlying AuthorizedPrivilege.
APElementName	Optional	The ElementName as reported in the underlying AuthorizedPrivilege.
LPSystemCreationClassName	Mandatory	The SystemCreationClassName as reported in the underlying LogicalPort. This may be NULL if the underlying LogicalPort is an Ethernet Port.
LPCreationClassName	Mandatory	The CreationClassName as reported in the underlying LogicalPort. This may be NULL if the underlying LogicalPort is an Ethernet Port.
LPSystemName	Mandatory	The SystemName as reported in the underlying LogicalPort. This may be NULL if the underlying LogicalPort is an Ethernet Port.
LPDeviceID	Mandatory	The DeviceID as reported in the underlying LogicalPort. This may be NULL if the underlying LogicalPort is an Ethernet Port.
LPOperationalStatus	Mandatory	The OperationalStatus as reported in the underlying LogicalPort. This may be NULL if the underlying LogicalPort is an Ethernet Port.
LPUsageRestriction	Mandatory	The UsageRestriction as reported in the underlying LogicalPort. This may be NULL if the underlying LogicalPort is an Ethernet Port.
LPPortType	Mandatory	The PortType as reported in the underlying LogicalPort. This may be NULL if the underlying LogicalPort is an Ethernet Port.

7.7.23 SNIA_MaskingMapView

The SNIA_MaskingMapView instance is a view that is derived from CIM_StorageHardwareID, CIM_AuthorizedSubject, CIM_AuthorizedPrivilege, CIM_AuthorizedTarget, CIM_SCSIProtocolController, CIM_SAPAvailableForElement, CIM_SCSIProtocolEndpoint, CIM_ProtocolControllerForUnit and CIM_LogicalDevice. The SNIA_MaskingMapView is not subclassed from anything.

Created By: External

Modified By: External

Deleted By: External

Requirement: Required if the array property SNIA_ViewCapabilities.SupportedViews contains the string "MaskingMapView" (and the Masking and Mapping Profile is implemented).

Table 104 describes class SNIA_MaskingMapView.

Table 104 - SMI Referenced Properties/Methods for SNIA_MaskingMapView

Properties	Requirement	Description and Notes
SHIDStorageID	Mandatory	The StorageID from the referenced CIM_StorageHardwareID instance.
SHIDIDType	Mandatory	The IDType from the referenced CIM_StorageHardwareID instance.
LDDeviceID	Mandatory	The DeviceID from the referenced CIM_LogicalDevice instance.
SPEPSystemCreationClassName	Mandatory	The SystemCreationClassName from the referenced CIM_SCSIProtocolEndpoint instance.
SPEPCreationClassName	Mandatory	The CreationClassName from the referenced CIM_SCSIProtocolEndpoint instance.
SPEPSystemName	Mandatory	The SystemName from the referenced CIM_SCSIProtocolEndpoint instance.
SPEPName	Mandatory	The Name from the referenced CIM_SCSIProtocolEndpoint instance.
SPEPRole	Mandatory	The Role from the referenced CIM_SCSIProtocolEndpoint instance.
APInstanceID	Mandatory	The InstanceID of the CIM_AuthorizedPrivilege instance.
APPrivilegeGranted	Mandatory	The PrivilegeGranted of the CIM_AuthorizedPrivilege instance.
APActivities	Mandatory	The Activities array of the CIM_AuthorizedPrivilege instance.
APElementName	Optional	The ElementName of the CIM_AuthorizedPrivilege instance.
SPCSystemCreationClassName	Mandatory	The SystemCreationClassName of the CIM_SCSIProtocolController instance.
SPCCreationClassName	Mandatory	The CreationClassName of the CIM_SCSIProtocolController instance.
SPCSystemName	Mandatory	The SystemName of the CIM_SCSIProtocolController instance.
SPCDeviceID	Mandatory	The DeviceID of the CIM_SCSIProtocolController instance.
PCFUDeviceNumber	Mandatory	The DeviceNumber (LUN) of the CIM_ProtocolControllerForUnit association instance.
PCFUDeviceAccess	Mandatory	The DeviceAccess value of the CIM_ProtocolControllerForUnit association instance.
StorageHardwareID	Mandatory	The CIM_StorageHardwareID instance that is associated to the CIM_LogicalDevice and CIM_ProtocolEndpoint instances.
LogicalDevice	Mandatory	The CIM_LogicalDevice instance that is associated to the CIM_StorageHardwareID and CIM_ProtocolEndpoint instances.
ProtocolEndpoint	Mandatory	The CIM_ProtocolEndpoint instance that is associated to the CIM_StorageHardwareID and CIM_LogicalDevice instances.

7.7.24 SNIA_ProtocolControllerForUnitView

Experimental. The SNIA_ProtocolControllerForUnitView instance is a view that associates a MappingProtocolControllerView and a LogicalDevice. It is derived from the underlying ProtocolControllerForUnit association between the underlying ProtocolController and the

LogicalDevice. Note that if the LogicalDevice is associated to multiple ProtocolControllers the DeviceNumber (LU Number) may differ for each MappingProtocolControllerView instance.

Requirement: Required if the array property SNIA_ViewCapabilities.SupportedViews contains the string "MappingProtocolControllerView" (and the Masking and Mapping Profile is implemented).

Table 105 describes class SNIA_ProtocolControllerForUnitView.

Table 105 - SMI Referenced Properties/Methods for SNIA_ProtocolControllerForUnitView

Properties	Requirement	Description and Notes
PCFUDeviceNumber	Mandatory	The DeviceNumber as reported in the underlying ProtocolControllerForUnit.
PCFUDeviceAccess	Mandatory	The DeviceAccess as reported in the underlying ProtocolControllerForUnit.
ManagedElement	Mandatory	The MappingProtocolControllerView instance.
LogicalDevice	Mandatory	The Storage Volume.

7.7.25 SNIA_ReplicaPairView

Experimental. The SNIA_ReplicaView instance is a view that is derived from a source and target CIM_StorageVolume (or CIM_LogicalDisk) and a CIM_StorageSynchronized association between them.

Requirement: Required if the array property SNIA_ViewCapabilities.SupportedViews contains the string "ReplicaPairView" (and the Copy Services Profile is implemented).

Table 106 describes class SNIA_ReplicaPairView.

Table 106 - SMI Referenced Properties/Methods for SNIA_ReplicaPairView

Properties	Requirement	Description and Notes
SVSourceSystemCreationClassName	Mandatory	The SystemCreationClassName as reported in the underlying source StorageVolume (or LogicalDisk).
SVSourceSystemName	Mandatory	The SystemName as reported in the underlying source StorageVolume (or LogicalDisk).
SVSourceCreationClassName	Mandatory	The CreationClassName as reported in the underlying source StorageVolume (or LogicalDisk).
SVSourceDeviceID	Mandatory	An opaque identifier of the underlying source StorageVolume (or LogicalDisk).
SVSourceName	Mandatory	The identifier for the underlying source StorageVolume (or LogicalDisk).
SVSourceNameFormat	Mandatory	The format of the identifier for the underlying source StorageVolume (or LogicalDisk).
SVSourceNameNamespace	Mandatory	The NameNamespace for the StorageVolume as reported in the underlying source StorageVolume instance.
SVSourceExtentStatus	Mandatory	The ExtentStatus as reported in the underlying source StorageVolume (or LogicalDisk).
SVSourceOperationalStatus	Mandatory	The OperationalStatus as reported in the underlying source StorageVolume (or LogicalDisk).
SVSourceBlockSize	Mandatory	The BlockSize as reported in the underlying source StorageVolume (or LogicalDisk).

Table 106 - SMI Referenced Properties/Methods for SNIA_ReplicaPairView (Continued)

Properties	Requirement	Description and Notes
SVSourceNumberOfBlocks	Mandatory	The number of blocks that make up the volume as reported in the underlying source StorageVolume (or LogicalDisk).
SVSourceConsumableBlocks	Mandatory	The number of usable blocks in the volume as reported in the underlying source StorageVolume (or LogicalDisk).
SVSourcePrimordial	Mandatory	This shall be Primordial='false'.
SVSourceIsBasedOnUnderlyingRedundancy	Mandatory	Whether or not redundancy is supported for the volume as reported in the underlying source StorageVolume (or LogicalDisk).
SVSourceNoSinglePointOfFailure	Mandatory	Whether or not NoSinglePointOfFailure is supported for the volume as reported in the underlying source StorageVolume (or LogicalDisk).
SVSourceDataRedundancy	Mandatory	The DataRedundancy supported by the volume as reported in the underlying source StorageVolume (or LogicalDisk).
SVSourcePackageRedundancy	Mandatory	The PackageRedundancy supported by the volume as reported in the underlying source StorageVolume (or LogicalDisk).
SVSourceDeltaReservation	Mandatory	The DeltaReservation supported by the volume as reported in the underlying source StorageVolume (or LogicalDisk).
SVSourceExtentDiscriminator	Mandatory	Experimental. The ExtentDiscriminator as reported in the underlying source StorageVolume (or LogicalDisk).
SVSourceOtherIdentifyingInfo	Optional	Other identifiers for the StorageVolume (or LogicalDisk) as reported in the underlying source StorageVolume (or LogicalDisk).
SVSourceIdentifyingDescriptions	Optional	The description of the other identifiers for the StorageVolume (or LogicalDisk) as reported in the underlying source StorageVolume (or LogicalDisk).
SVSourceElementName	Optional	The user friendly name for the underlying source StorageVolume (or LogicalDisk).
SVSourceUsage	Optional	The Usage supported by the volume as reported in the underlying source StorageVolume (or LogicalDisk).
SVSourceOtherUsageDescription	Optional	The OtherUsageDescription supported by the volume as reported in the underlying source StorageVolume (or LogicalDisk).
SVSourceClientSettableUsage	Optional	The ClientSettableUsage supported by the volume as reported in the underlying source StorageVolume (or LogicalDisk).
SSWhenSynced	Mandatory	The WhenSynced as reported in the underlying StorageSynchronized association between the source and target StorageVolumes (or LogicalDisks).
SSSyncMaintained	Mandatory	The SyncMaintained as reported in the underlying StorageSynchronized association between the source and target StorageVolumes (or LogicalDisks).
SSCopyType	Mandatory	The CopyType as reported in the underlying StorageSynchronized association between the source and target StorageVolumes (or LogicalDisks).
SSSyncState	Mandatory	The SyncState as reported in the underlying StorageSynchronized association between the source and target StorageVolumes (or LogicalDisks).

Table 106 - SMI Referenced Properties/Methods for SNIA_ReplicaPairView (Continued)

Properties	Requirement	Description and Notes
SSCopyPriority	Mandatory	The CopyPriority as reported in the underlying StorageSynchronized association between the source and target StorageVolumes (or LogicalDisks).
SSSyncType	Mandatory	The SyncType as reported in the underlying StorageSynchronized association between the source and target StorageVolumes (or LogicalDisks).
SSMode	Mandatory	The Mode as reported in the underlying StorageSynchronized association between the source and target StorageVolumes (or LogicalDisks).
SSProgressStatus	Mandatory	The ProgressStatus as reported in the underlying StorageSynchronized association between the source and target StorageVolumes (or LogicalDisks).
SVTargetSystemCreationClassName	Mandatory	The SystemCreationClassName as reported in the underlying target StorageVolume (or LogicalDisk).
SVTargetSystemName	Mandatory	The SystemName as reported in the underlying target StorageVolume (or LogicalDisk).
SVTargetCreationClassName	Mandatory	The CreationClassName as reported in the underlying target StorageVolume (or LogicalDisk).
SVTargetDeviceID	Mandatory	An opaque identifier of the underlying target StorageVolume (or LogicalDisk).
SVTargetName	Mandatory	The identifier for the underlying target StorageVolume (or LogicalDisk).
SVTargetNameFormat	Mandatory	The format of the identifier for the underlying target StorageVolume (or LogicalDisk).
SVTargetNameNamespace	Mandatory	The NameNamespace for the StorageVolume as reported in the underlying target StorageVolume instance.
SVTargetExtentStatus	Mandatory	The ExtentStatus as reported in the underlying target StorageVolume (or LogicalDisk).
SVTargetOperationalStatus	Mandatory	The OperationalStatus as reported in the underlying target StorageVolume (or LogicalDisk).
SVTargetBlockSize	Mandatory	The BlockSize as reported in the underlying target StorageVolume (or LogicalDisk).
SVTargetNumberOfBlocks	Mandatory	The number of blocks that make up the volume as reported in the underlying target StorageVolume (or LogicalDisk).
SVTargetConsumableBlocks	Mandatory	The number of usable blocks in the volume as reported in the underlying target StorageVolume (or LogicalDisk).
SVTargetPrimordial	Mandatory	This shall be Primordial='false'.
SVTargetIsBasedOnUnderlyingRedundancy	Mandatory	Whether or not redundancy is supported for the volume as reported in the underlying target StorageVolume (or LogicalDisk).
SVTargetNoSinglePointOfFailure	Mandatory	Whether or not NoSinglePointOfFailure is supported for the volume as reported in the underlying target StorageVolume (or LogicalDisk).
SVTargetDataRedundancy	Mandatory	The DataRedundancy supported by the volume as reported in the underlying target StorageVolume (or LogicalDisk).
SVTargetPackageRedundancy	Mandatory	The PackageRedundancy supported by the volume as reported in the underlying target StorageVolume (or LogicalDisk).

Table 106 - SMI Referenced Properties/Methods for SNIA_ReplicaPairView (Continued)

Properties	Requirement	Description and Notes
SVTargetDeltaReservation	Mandatory	The DeltaReservation supported by the volume as reported in the underlying target StorageVolume (or LogicalDisk).
SVTargetExtentDiscriminator	Mandatory	Experimental. The ExtentDiscriminator as reported in the underlying target StorageVolume (or LogicalDisk).
SVTargetOtherIdentifyingInfo	Optional	Other identifiers for the StorageVolume (or LogicalDisk) as reported in the underlying target StorageVolume (or LogicalDisk).
SVTargetIdentifyingDescriptions	Optional	The description of the other identifiers for the StorageVolume (or LogicalDisk) as reported in the underlying target StorageVolume (or LogicalDisk).
SVTargetElementName	Optional	The user friendly name for the underlying target StorageVolume (or LogicalDisk).
SVTargetUsage	Optional	The Usage supported by the volume as reported in the underlying target StorageVolume (or LogicalDisk).
SVTargetOtherUsageDescription	Optional	The OtherUsageDescription supported by the volume as reported in the underlying target StorageVolume (or LogicalDisk).
SVTargetClientSettableUsage	Optional	The ClientSettableUsage supported by the volume as reported in the underlying target StorageVolume (or LogicalDisk).

7.7.26 SNIA_StoragePoolView

Experimental. The SNIA_StoragePoolView is a view that is derived from CIM_StoragePool, CIM_StorageCapabilities, CIM_StorageConfigurationCapabilities, as well as the SpaceConsumed data from the CIM_AllocatedFromStoragePool (to its parent pool).

Requirement: Required if the array property SNIA_ViewCapabilities.SupportedViews contains the string "StoragePoolView" (and the Block Service Package is implemented).

Table 107 describes class SNIA_StoragePoolView.

Table 107 - SMI Referenced Properties/Methods for SNIA_StoragePoolView

Properties	Requirement	Description and Notes
SPInstanceID	Mandatory	The InstanceID as reported in the underlying StoragePool.
SPElementName	Optional	The ElementName as reported in the underlying StoragePool.
SPPoolID	Mandatory	The PoolID as reported in the underlying StoragePool.
SPRemainingManagedSpace	Mandatory	The RemainingManagedSpace as reported in the underlying StoragePool.
SPTotalManagedSpace	Mandatory	The TotalManagedSpace as reported in the underlying StoragePool.
SPPrimordial	Mandatory	The Primordial property as reported in the underlying StoragePool.
SPUsage	Optional	The Usage property as reported in the underlying StoragePool.
SPOtherUsageDescription	Optional	The OtherUsageDescription as reported in the underlying StoragePool.
SPClientSettableUsage	Optional	The ClientSettableUsage as reported in the underlying StoragePool.

Table 107 - SMI Referenced Properties/Methods for SNIA_StoragePoolView (Continued)

Properties	Requirement	Description and Notes
SCInstanceID	Mandatory	The InstanceID as reported in the underlying StorageCapabilities associated to the StoragePool.
SCElementName	Mandatory	The ElementName as reported in the underlying StorageCapabilities associated to the StoragePool.
SCElementType	Mandatory	The ElementType as reported in the underlying StorageCapabilities associated to the StoragePool.
SCNoSinglePointOfFailure	Mandatory	The NoSinglePointOfFailure as reported in the underlying StorageCapabilities associated to the StoragePool.
SCNoSinglePointOfFailureDefault	Mandatory	The NoSinglePointOfFailureDefault as reported in the underlying StorageCapabilities associated to the StoragePool.
SCPackageRedundancyDefault	Mandatory	The PackageRedundancyDefault as reported in the underlying StorageCapabilities associated to the StoragePool.
SCPackageRedundancyMin	Mandatory	The PackageRedundancyMin as reported in the underlying StorageCapabilities associated to the StoragePool.
SCPackageRedundancyMax	Mandatory	The PackageRedundancyMax as reported in the underlying StorageCapabilities associated to the StoragePool.
SCDataRedundancyDefault	Mandatory	The DataRedundancyDefault as reported in the underlying StorageCapabilities associated to the StoragePool.
SCDataRedundancyMin	Mandatory	The DataRedundancyMin as reported in the underlying StorageCapabilities associated to the StoragePool.
SCDataRedundancyMax	Mandatory	The DataRedundancyMax as reported in the underlying StorageCapabilities associated to the StoragePool.
SCExtentStripeLengthDefault	Optional	The ExtentStripeLengthDefault as reported in the underlying StorageCapabilities associated to the StoragePool.
SCParityLayoutDefault	Optional	The ParityLayoutDefault as reported in the underlying StorageCapabilities associated to the StoragePool.
SCUserDataStripeDepthDefault	Optional	The UserDataStripeDepthDefault as reported in the underlying StorageCapabilities associated to the StoragePool.
AFSPSpaceConsumed	Mandatory	The SpaceConsumed as reported in the underlying AllocatedFromStoragePool to this pool's parent pool.
SCCInstanceID	Mandatory	The InstanceID as reported in the underlying StorageConfigurationCapabilities (if any) associated to the StoragePool.
SCCElementName	Mandatory	The ElementName as reported in the underlying StorageConfigurationCapabilities (if any) associated to the StoragePool.
SCCSupportedStoragePoolFeatures	Mandatory	The SupportedStoragePoolFeatures as reported in the underlying StorageConfigurationCapabilities (if any) associated to the StoragePool.
SCCSupportedStorageElementTypes	Mandatory	The SupportedStorageElementTypes as reported in the underlying StorageConfigurationCapabilities (if any) associated to the StoragePool.
SCCSupportedStorageElementFeatures	Mandatory	The SupportedStorageElementFeatures as reported in the underlying StorageConfigurationCapabilities (if any) associated to the StoragePool.
SCCSupportedSynchronousActions	Optional	The SupportedSynchronousActions as reported in the underlying StorageConfigurationCapabilities (if any) associated to the StoragePool.

Table 107 - SMI Referenced Properties/Methods for SNIA_StoragePoolView (Continued)

Properties	Requirement	Description and Notes
SCCSupportedAsynchronousActions	Optional	The SupportedAsynchronousActions as reported in the underlying StorageConfigurationCapabilities (if any) associated to the StoragePool.
SCCSupportedStorageElementUsage	Optional	The SupportedStorageElementUsage as reported in the underlying StorageConfigurationCapabilities (if any) associated to the StoragePool.
SCCClientSettableElementUsage	Optional	The ClientSettableElementUsage as reported in the underlying StorageConfigurationCapabilities (if any) associated to the StoragePool.
SCCSupportedStoragePoolUsage	Optional	The SupportedStoragePoolUsage as reported in the underlying StorageConfigurationCapabilities (if any) associated to the StoragePool.
SCCClientSettablePoolUsage	Optional	The ClientSettablePoolUsage as reported in the underlying StorageConfigurationCapabilities (if any) associated to the StoragePool.

7.7.27 SNIA_SystemDeviceView (DiskDriveViews)

Created By: External

Modified By: Static

Deleted By: External

Requirement: Required if the array property SNIA_ViewCapabilities.SupportedViews contains the string "DiskDriveView" (and the Disk Drive Lite Profile is implemented).

Table 108 describes class SNIA_SystemDeviceView (DiskDriveViews).

Table 108 - SMI Referenced Properties/Methods for SNIA_SystemDeviceView (DiskDriveViews)

Properties	Requirement	Description and Notes
GroupComponent	Mandatory	The Computer System that contains this DiskDriveView instance.
PartComponent	Mandatory	The SNIA_DiskDriveView instance that is a device on the computer system.

7.7.28 SNIA_SystemDeviceView (MappingProtocolControllerViews)

Created By: External

Modified By: Static

Deleted By: External

Requirement: Required if the array property SNIA_ViewCapabilities.SupportedViews contains the string "MappingProtocolControllerView" (and the Masking and Mapping Profile is implemented).

Table 109 describes class SNIA_SystemDeviceView (MappingProtocolControllerViews).

Table 109 - SMI Referenced Properties/Methods for SNIA_SystemDeviceView (MappingProtocolControllerViews)

Properties	Requirement	Description and Notes
GroupComponent	Mandatory	The Computer System that contains this MappingProtocolControllerView instance.
PartComponent	Mandatory	The SNIA_MappingProtocolControllerView instance that is a device on the computer system.

7.7.29 SNIA_SystemDeviceView (ReplicaPairViews)

Created By: External

Modified By: Static

Deleted By: External

Requirement: Required if the array property SNIA_ViewCapabilities.SupportedViews contains the string "ReplicaPairView" (and the Copy Services Profile is implemented).

Table 110 describes class SNIA_SystemDeviceView (ReplicaPairViews).

Table 110 - SMI Referenced Properties/Methods for SNIA_SystemDeviceView (ReplicaPairViews)

Properties	Requirement	Description and Notes
GroupComponent	Mandatory	The Computer System that contains this ReplicaPairView instance.
PartComponent	Mandatory	The SNIA_ReplicaPairView instance that is a device on the computer system.

7.7.30 SNIA_SystemDeviceView (VolumeViews)

Created By: External

Modified By: Static

Deleted By: External

Requirement: Required if the array property SNIA_ViewCapabilities.SupportedViews contains the string "VolumeView" (and the Block Service Package is implemented).

Table 111 describes class SNIA_SystemDeviceView (VolumeViews).

Table 111 - SMI Referenced Properties/Methods for SNIA_SystemDeviceView (VolumeViews)

Properties	Requirement	Description and Notes
GroupComponent	Mandatory	The Computer System that contains this VolumeView instance.
PartComponent	Mandatory	The SNIA_VolumeView instance that is a device on the computer system.

7.7.31 SNIA_ViewCapabilities

The SNIA_ViewCapabilities instance defines the capabilities of an implementation support for SNIA_view classes. The SNIA_ViewCapabilities is subclassed from CIM_Capabilities.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 112 describes class SNIA_ViewCapabilities.

Table 112 - SMI Referenced Properties/Methods for SNIA_ViewCapabilities

Properties	Requirement	Description and Notes
InstanceID	Mandatory	An opaque, unique id for the view class capability of an implementation.
ElementName	Optional	A provider supplied user-friendly Name for this SNIA_ViewCapabilities element.
SupportedViews	Mandatory	This array of strings lists the view classes that are supported by the implementation. Valid string values are "VolumeView" "DiskDriveView" "ExposedView" "MaskingMapView" "MappingProtocolControllerView" "StoragePoolView" "ReplicaPairView" .

7.7.32 SNIA_VolumeView

The SNIA_VolumeView instance is a view that is derived from CIM_StorageVolume, CIM_ElementSettingData, CIM_StorageSetting, CIM_AllocatedFromStoragePool and CIM_StoragePool. The SNIA_VolumeView is subclassed from CIM_ManagedElement.

Created By: External

Modified By: External

Deleted By: External

Requirement: Required if the array property SNIA_ViewCapabilities.SupportedViews contains the string "VolumeView" (and the Block Service Package is implemented).

Table 113 describes class SNIA_VolumeView.

Table 113 - SMI Referenced Properties/Methods for SNIA_VolumeView

Properties	Requirement	Description and Notes
SVSystemCreationClassName	Mandatory	The SystemCreationClassName for the underlying StorageVolume (or LogicalDisk).
SVSystemName	Mandatory	The SystemName for the underlying StorageVolume (or LogicalDisk).
SVCreationClassName	Mandatory	The CreationClassName for the underlying StorageVolume (or LogicalDisk).
SVDeviceID	Mandatory	An opaque identifier of the underlying StorageVolume (or LogicalDisk).
SVName	Mandatory	The identifier for the underlying StorageVolume (or LogicalDisk).
SVNameFormat	Mandatory	The format of the identifier for the underlying StorageVolume (or LogicalDisk).
SVNameNamespace	Mandatory	The NameNamespace for the StorageVolume as reported in the underlying StorageVolume instance.
SVExtentStatus	Mandatory	The ExtentStatus as reported in the underlying StorageVolume (or LogicalDisk).
SVOperationalStatus	Mandatory	The OperationalStatus as reported in the underlying StorageVolume (or LogicalDisk).
SVBlockSize	Mandatory	
SVNumberOfBlocks	Mandatory	The number of blocks that make up the volume as reported in the underlying StorageVolume (or LogicalDisk).
SVConsumableBlocks	Mandatory	The number of usable blocks in the volume as reported in the underlying StorageVolume (or LogicalDisk).
SVIsBasedOnUnderlyingRedundancy	Mandatory	Whether or not redundancy is supported for the volume as reported in the underlying StorageVolume (or LogicalDisk).
SVNoSinglePointOfFailure	Mandatory	Whether or not NoSinglePointOfFailure is supported for the volume as reported in the underlying StorageVolume (or LogicalDisk).
SVDataRedundancy	Mandatory	The DataRedundancy supported by the volume as reported in the underlying StorageVolume (or LogicalDisk).
SVPackageRedundancy	Mandatory	The PackageRedundancy supported by the volume as reported in the underlying StorageVolume (or LogicalDisk).
SVDeltaReservation	Mandatory	The DeltaReservation supported by the volume as reported in the underlying StorageVolume (or LogicalDisk).
SVPrimordial	Mandatory	
SVExtentDiscriminator	Mandatory	Experimental.
SSInstanceID	Mandatory	The InstanceID of the StorageSetting for the volume as reported in its underlying StorageSetting.
SSElementName	Mandatory	The ElementName of the StorageSetting for the volume as reported in its underlying StorageSetting.
SSNoSinglePointOfFailure	Mandatory	Whether or not NoSinglePointOfFailure was requested in the StorageSetting for the volume as reported in its underlying StorageSetting.
SSDataRedundancyMin	Mandatory	The DataRedundancyMin value supported by the StorageSetting for the volume as reported in its underlying StorageSetting.

Table 113 - SMI Referenced Properties/Methods for SNIA_VolumeView (Continued)

Properties	Requirement	Description and Notes
SSDataRedundancyMax	Mandatory	The DataRedundancyMax value supported by the StorageSetting for the volume as reported in its underlying StorageSetting.
SSDataRedundancyGoal	Mandatory	The DataRedundancyGoal supported by the StorageSetting for the volume as reported in its underlying StorageSetting.
SSPackageRedundancyMin	Mandatory	The PackageRedundancyMin value supported by the StorageSetting for the volume as reported in its underlying StorageSetting.
SSPackageRedundancyMax	Mandatory	The PackageRedundancyMax value supported by the StorageSetting for the volume as reported in its underlying StorageSetting.
SSPackageRedundancyGoal	Mandatory	The PackageRedundancyGoal supported by the StorageSetting for the volume as reported in its underlying StorageSetting.
SSChangeableType	Mandatory	The ChangeableType defined for the StorageSetting for the volume as reported in the underlying StorageSetting.
AFSPSpaceConsumed	Mandatory	The SpaceConsumed from the StoragePool by the volume as reported in its underlying AllocatedFromStoragePool association to the StoragePool.
SPInstanceID	Mandatory	The InstanceID of the StoragePool for the volume as reported in the underlying StoragePool.
SPPoolID	Mandatory	The PoolID of the StoragePool for the volume as reported in the underlying StoragePool.
SVOtherIdentifyingInfo	Optional	Other identifiers for the StorageVolume (or LogicalDisk) as reported in the underlying StorageVolume (or LogicalDisk).
SVIdentifyingDescriptions	Optional	The description of the other identifiers for the StorageVolume (or LogicalDisk) as reported in the underlying StorageVolume (or LogicalDisk).
SVElementName	Optional	The user friendly name for the underlying StorageVolume (or LogicalDisk).
SVUsage	Optional	The Usage supported by the volume as reported in the underlying StorageVolume (or LogicalDisk).
SVOtherUsageDescription	Optional	The OtherUsageDescription supported by the volume as reported in the underlying StorageVolume (or LogicalDisk).
SVClientSettableUsage	Optional	The ClientSettableUsage supported by the volume as reported in the underlying StorageVolume (or LogicalDisk).
SSExtentStripeLength	Optional	The ExtentStripeLength value supported by the StorageSetting for the volume as reported in its underlying StorageSetting.
SSExtentStripeLengthMin	Optional	The ExtentStripeLengthMin value supported by the StorageSetting for the volume as reported in its underlying StorageSetting.
SSExtentStripeLengthMax	Optional	The ExtentStripeLengthMax supported by the StorageSetting for the volume as reported in its underlying StorageSetting.
SSParityLayout	Optional	The ParityLayout defined by the StorageSetting for the volume as reported in its underlying StorageSetting.
SSUserDataStripeDepth	Optional	The UserDataStripeDepth value supported by the StorageSetting for the volume as reported in its underlying StorageSetting.

Table 113 - SMI Referenced Properties/Methods for SNIA_VolumeView (Continued)

Properties	Requirement	Description and Notes
SSUserDataStripeDepthMin	Optional	The UserDataStripeDepthMin value supported by the StorageSetting for the volume as reported in its underlying StorageSetting.
SSUserDataStripeDepthMax	Optional	The UserDataStripeDepthMax value supported by the StorageSetting for the volume as reported in its underlying StorageSetting.
SSStorageExtentInitialUsage	Optional	The StorageExtentInitialUsage value supported by the StorageSetting for the volume as reported in its underlying StorageSetting.
SSStoragePoolInitialUsage	Optional	

EXPERIMENTAL

IECNORM.COM : Click to view the full PDF of ISO/IEC 24775-4:2014

STABLE

8 Block Server Performance Subprofile

8.1 Description

8.1.1 Synopsis

Profile Name: Block Server Performance (Component Profile)

Version: 1.5.0

Organization: SNIA

CIM Schema Version: 2.11.0

Table 114 describes the related profiles for Block Server Performance.

Table 114 - Related Profiles for Block Server Performance

Profile Name	Organization	Version	Requirement	Description
Multiple Computer System	SNIA	1.2.0	Optional	
Extent Composition	SNIA	1.5.0	Optional	
SPI Target Ports	SNIA	1.4.0	Optional	
FC Target Ports	SNIA	1.4.0	Optional	
iSCSI Target Ports	SNIA	1.2.0	Optional	
DA Target Ports	SNIA	1.4.0	Optional	
SPI Initiator Ports	SNIA	1.4.0	Optional	
FC Initiator Ports	SNIA	1.4.0	Optional	
iSCSI Initiator Ports	SNIA	1.2.0	Optional	
Disk Drive Lite	SNIA	1.5.0	Optional	
Replication Services	SNIA	1.5.0	Optional	Experimental.

NOTE Each of these subprofiles is mandatory if the element in question is to be metered. For example, in order to keep statistics on Disk Drives, it will be necessary for Disk Drives to be modeled.

Central Class: BlockStatisticsService

Scoping Class: ComputerSystem

8.1.2 Overview

The Block Server Performance Subprofile defines classes and methods for managing performance information in block servers (e.g., Arrays, Storage Virtualizers and Volume Management). Not all of the objects for which statistics are defined apply to all these profiles. For example, Storage Virtualizers don't have Disk Drives and Volume Management Profiles don't have Ports. In these cases, the profile would not support the statistics for the object that does not apply to it.

NOTE 1 Performance analysis is broader than just Arrays, Storage Virtualizers and Volume Managers. Complete analysis requires performance information from hosts and fabric. These are (or will be) addressed separately as part of the appropriate profiles.

One of the key SRM disciplines for managing block servers (e.g., arrays) is Performance Management. Currently, there are no common statistics defined that can be used to manage

multiple vendor arrays from a performance perspective. Some of the key tasks commonly performed in the discipline of Performance Management are:

- Performance Capacity Planning,
- Performance Problem Isolation,
- Peak Window Analysis,
- Block server Workload Analysis,
- Block server Performance Tuning.

In order to manage performance, a number of processes need to be in place:

- Ability to measure the performance and saturation points of components within the storage network. This subprofile describes the first increment of measurement, that of the storage system. Examples of this include:
 - Read and Write I/O counts for a LUN or a disk,
 - Number of blocks transferred per unit time,
 - Cache hit ratios.

Both specific measurements and methods to make these measurements available to SRM applications will be part of this subprofile.

- Ability to understand the relationship of facilities within the storage network and their relationship to the actual application. This is provided by mapping functions which are described in this International Standard. Mapping functions are listed within the International Standard today. As new objects (like cache which is currently not defined) and new relationships between objects are defined, these parts of this International Standard will have to be upgraded.
- Ability to understand the status and configuration of the storage network components. There is some level of this information within the SMI International Standard today, and there are expected future improvements to this area that will be in future releases. Examples of this include:
 - Cache status on or off for read or write cache,
 - How much Cache is installed,
 - Storage Volume (LUN) status, normal or degraded,
 - Cache configuration parameters,
 - LUN status,
 - Error counts on a port.

Methods to be able to tune the configuration of a storage network component. This would include setting RAID levels, setting stripe widths, setting cache tunable parameters, etc. This is an area for future development. Given that there is a wide diversity of storage architectures, this may be an area where SMI provides a framework and vendors supply the custom extensions required for their systems.

Performance Management is optimized when all four components are in place. Performance Measurement is the key deliverable that is the focus of this subprofile.

Block storage devices usually have one or more of the following elements:

- Block Server (top level ComputerSystem),
- I/O Ports (e.g., FCPorts),
- Front-end Ports,

- Back-end Ports,

NOTE 2 Port Statistics in block servers need to be coordinated with Port statistics in the Fabric Profile by applications. A mapping between fabric statistics and block server statistics is identified in 8.7.7.

- Individual Controllers (ComponentCS),
- Front-end controller(s) (ComponentCS),
- Back-end controller(s) (ComponentCS),
- Exported Elements (e.g., Volumes or Logical Disks),
- Imported Elements (e.g., Extents with ConcreteComponent association to Pools),
- Disk Drives.

In order to monitor and manage these components, it is necessary to identify performance counters for each of the above elements in the block server and externalize an interface to obtain these counters at some SRM-determined periodicity. An SRM product will also need to be able to associate these counters to the appropriate block server elements as defined in the appropriate SMI-S profiles in order to complete the full picture of the performance analysis (e.g., what disks are part of this LUN and what other LUNs have portions on this disk).

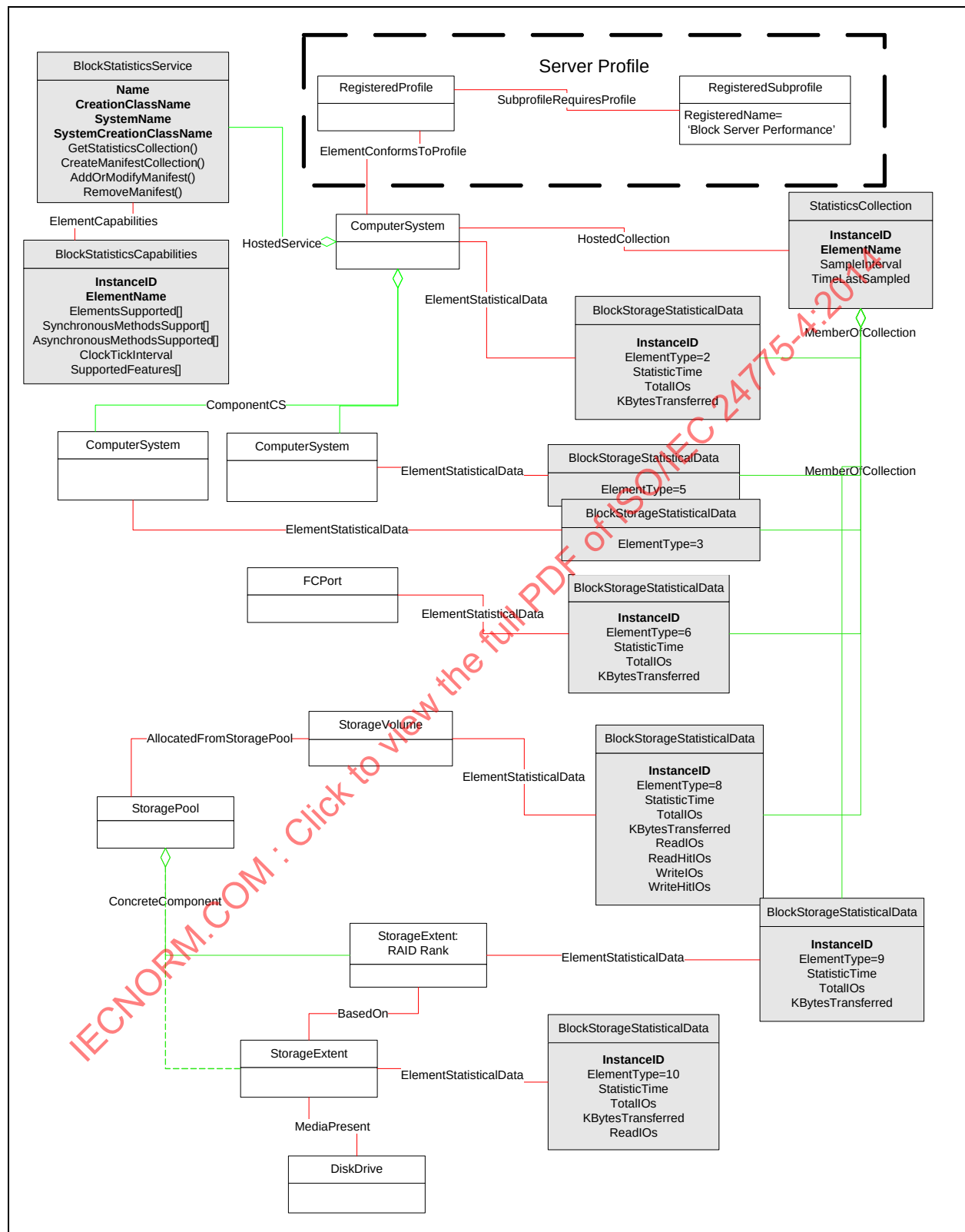
The function of this subprofile is to support the aforementioned SRM applications.

The Block Server Performance Subprofile augments the profiles and subprofiles for Arrays, Storage Virtualizers and Volume Management Profiles. Instead of being an isolated subprofile, it adds modeling constructs to existing profiles and subprofiles. Together these enhancements make up the Block Server Performance Subprofile (as would be registered in the Server Profile as a RegisteredSubprofile).

8.2 Implementation

8.2.1 Performance Additions Overview

Figure 34: "Block Server Performance Subprofile Summary Instance Diagram" provides an overview of the model (independent of profiles and subprofiles). The new classes added by the Block Server Performance Subprofile are the shaded grey boxes.



NOTE The properties listed for the statistics classes are the mandatory properties. Optional Properties are not listed in order to save space in the diagram. Optional properties can be found in 8.8 "CIM Elements".

What Figure 34 shows is a single instance of `StatisticsCollection` for the entire profile. This is the anchor point from which all statistics being kept by the profile can be found. Block statistics

are defined as a `BlockStorageStatisticalData` class, instances of which hold the statistics for particular elements (e.g., `StorageVolumes`, `ComputerSystems`, `Ports`, `Extents` and `Disk Drives`). The type of element is recorded in the instance of `BlockStorageStatisticalData` in the `ElementType` property.

All the statistics instances are related to the elements they meter via the `ElementStatisticalData` association (e.g., `BlockStorageStatisticalData` for a `StorageVolume` can be found from the `Volume` by traversing the `ElementStatisticalData` association).

All the statistics instances kept in the profile are associated to the one `StatisticsCollection` instance. Access to all the statistics for the profile is through the `StatisticsCollection`. The `StatisticsCollection` has a `HostedCollection` association to the "top level" computer system of the profile.

Note that statistics may be kept for a number of elements in the profile, including elements in subprofiles. The elements that are metered are:

The top level ComputerSystem – This provides a summary of all statistics for the whole profile (e.g., `ReadIOs` are all read IOs handled by the array, storage virtualizer or volume manager).

Component ComputerSystems – This provides a summary of all statistics that derive from a particular processor in the system cluster (e.g., all `ReadIOs` handled by a particular processor). These statistics are kept in `BlockStorageStatisticalData` instances (one for each component computer system).

Port – This provides a summary of all the statistics that derive from a particular Port on the Array or Storage Virtualizer (e.g., all `ReadIOs` that go through the particular port). These statistics are kept in `BlockStorageStatisticalData` instances (one for each Port in the system).

Note that this element does not apply to the Volume Management Profile. Volume managers do not have front-end ports. The back-end ports for volume managers are HBAs. Statistics for volume manager back end ports would be kept by the HBAs.

StorageVolume – (or `LogicalDisk`). This provides a summary of statistics for a particular `StorageVolume` (or `LogicalDisk`). For example, all the `ReadIOs` to the particular `StorageVolume` (or `LogicalDisk`). These statistics are kept in `BlockStorageStatisticalData` instances (one for each `StorageVolume` or `LogicalDisk` in the system).

StorageExtent – This provides a summary of statistics that derive from access to a particular `StorageExtent`. Note that `StorageExtent` support is ONLY PROVIDED for extents with a `ConcreteComponent` association to a concrete `StoragePool`. That is, this is not offered for intermediate extents. These statistics are kept in `BlockStorageStatisticalData` instances (one for each `Extent` that is modeled in the system).

SCSI Arbitrary Logical Units – This provides summary of statistics that derive from access to LUNs that are not `StorageVolumes` (e.g., controller commands).

Remote Replica Groups – This provides summary of statistics that derive from access remote replica volumes.

Finally, Figure 35: "Base Array Profile Block Server Performance Instance Diagram" illustrates the `BlockStatisticsService` for Bulk retrieval of all the statistics data and creation of manifest collections. These methods will be discussed later. They are shown here for completeness. Associated with the `BlockStatisticsService` is a `BlockStatisticsCapabilities` instance that identifies the specific capabilities implemented by the performance support. Specifically, it includes an "ElementsSupported" property that identifies the elements for which statistics are kept and the various retrieval mechanisms that are implemented (e.g., `Extrinsic`, `Association Traversal`, `Indications` and/or `Query`).

EXPERIMENTAL

The BlockStatisticsCapabilities also includes a SupportedFeatures property for identifying specific features of the implementation.

EXPERIMENTAL

8.2.2 Performance Additions to base Array Profile

Figure 35: "Base Array Profile Block Server Performance Instance Diagram" illustrates the class instances that would be supported if an Array only implemented the base Array Profile and the Block Server Performance Subprofile. Only the StatisticsCollection, the BlockStorageStatisticalData instance for the top level computer system, BlockStorageStatisticalData instances for front end ports and BlockStorageStatisticalData instances for Storage Volumes would be supported.

Only the GetStatisticsCollection method of the BlockStatisticsService would be supported. The actual elements for which the statistics would be kept would be reported in the "ElementsSupported" property of the BlockStatisticsCapabilities instance.

IECNORM.COM : Click to view the full PDF of ISO/IEC 24775-4:2014

Figure 36: "Base Storage Virtualizer Profile Block Server Performance Instance Diagram" illustrates the class instances that would be supported if a Storage Virtualizer only

implemented the base Storage Virtualizer Profile and the Block Server Performance Subprofile. Only the StatisticsCollection, the BlockStorageStatisticalData instance for the top level computer system, BlockStorageStatisticalData instances for front-end and back-end ports, BlockStorageStatisticalData instances for Storage Volumes and BlockStorageStatisticalData for StorageExtents would be supported.

Only the GetStatisticsCollection method of the BlockStatisticsService would be supported. The actual elements for which the statistics would be kept would be reported in the "ElementsSupported" property of the BlockStatisticsCapabilities instance.

IECNORM.COM : Click to view the full PDF of ISO/IEC 24775-4:2014

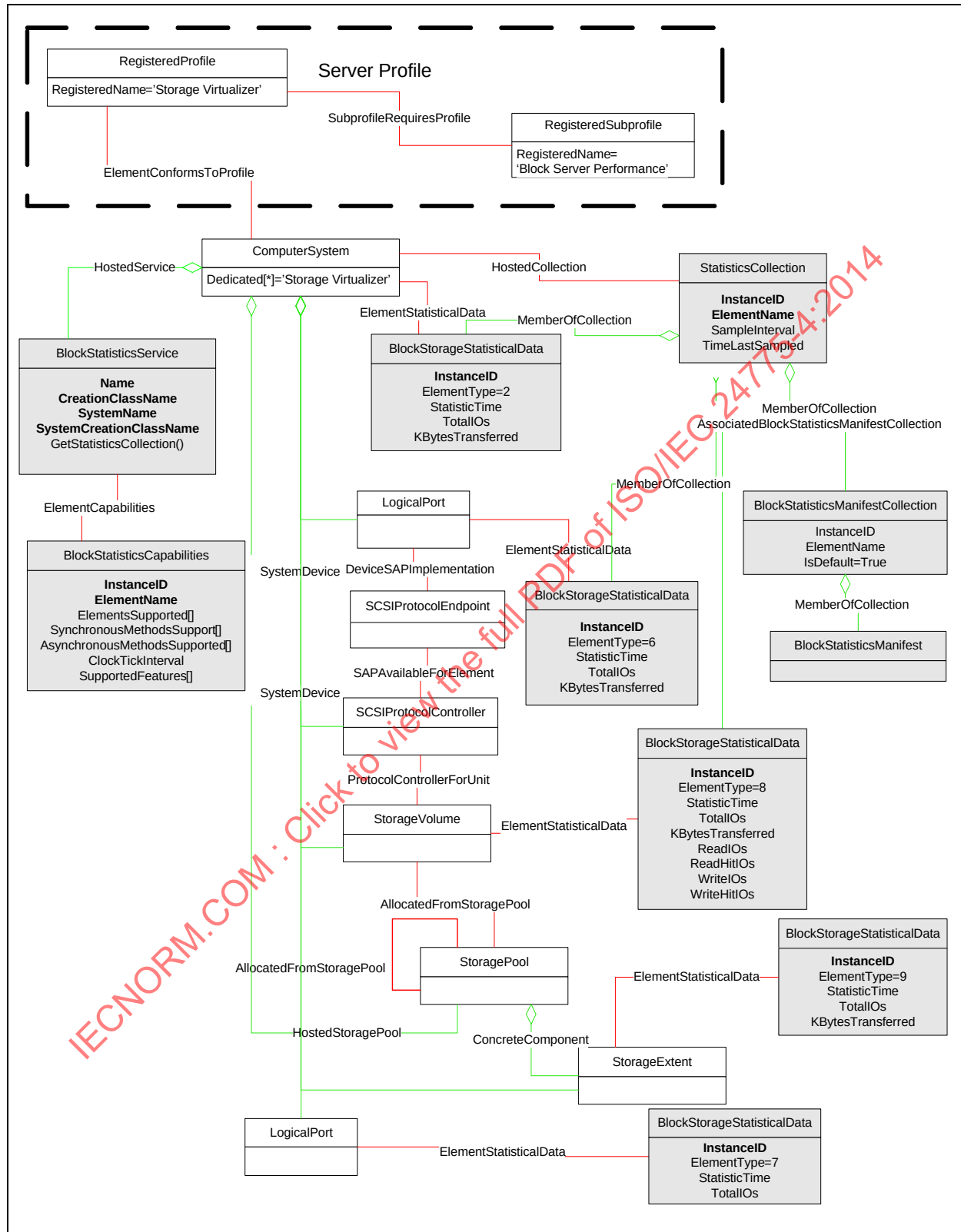


Figure 36 - Base Storage Virtualizer Profile Block Server Performance Instance Diagram

NOTE The properties listed for the statistics classes are the mandatory properties. Optional Properties are not listed in order to save space in the diagram. Optional properties can be found in 8.8 "CIM Elements".

8.2.4 Performance Additions to base Volume Management Profile

Figure 37: "Base Volume Management Profile Block Server Performance Instance Diagram" illustrates the class instances that would be supported if the volume manager only implemented the base Volume Management Profile and the Block Server Performance Subprofile. Only the StatisticsCollection, the BlockStorageStatisticalData instance for the top level computer system, BlockStorageStatisticalData instances for LogicalDisks (lower extents) and BlockStorageStatisticalData instances for LogicalDisks (exported Logical Disks) would be supported.

IECNORM.COM : Click to view the full PDF of ISO/IEC 24775-4:2014

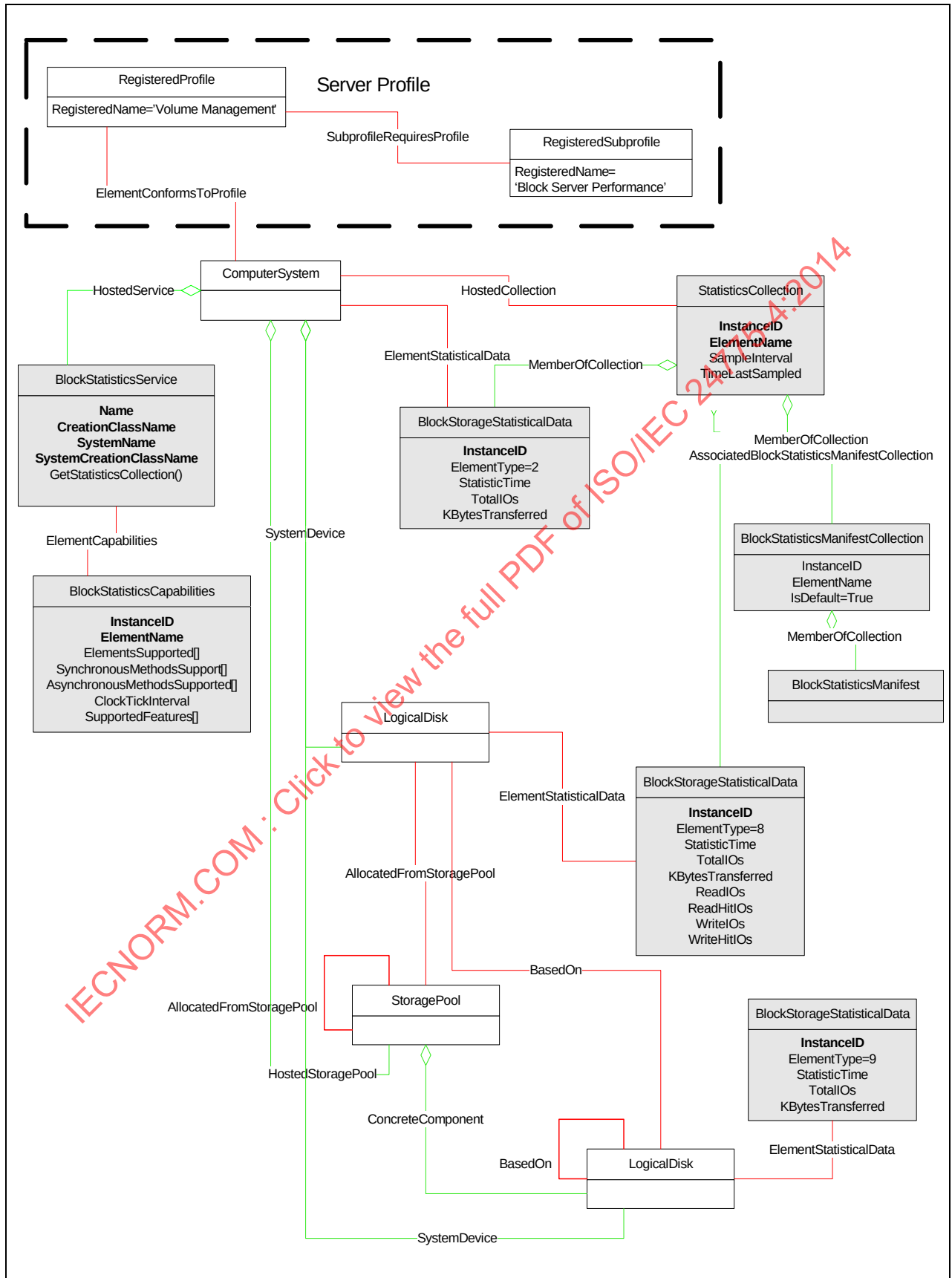


Figure 37 - Base Volume Management Profile Block Server Performance Instance Diagram

NOTE The properties listed for the statistics classes are the mandatory properties. Optional Properties are not listed in order to save space in the diagram. Optional properties can be found in 8.8 "CIM Elements".

8.2.5 Summary of BlockStorageStatisticsData support by Profile

Table 115 defines the Element Types (for BlockStorageStatisticalData instances) that may be supported by profile.

Table 115 - Summary of Element Types by Profile

ElementType	Array	Storage Virtualizer	Volume Management
Computer System	YES	YES	YES
Front-end Computer System	YES	YES	YES
Peer Computer System	YES	YES	YES
Back-end Computer System	YES	YES	YES
Front-end Port	YES	YES	NO
Back-end Port	YES	YES	NO
Volume	YES	YES	YES
Extent	YES	YES	YES
Disk Drive	YES	YES	NO
Arbitrary LUs	YES	YES	NO
Remote Replica Group	YES	YES	YES

YES means that this International Standard defines the element type for the profile. Actual support by any given implementation would be implementation dependent. But the International Standard covers defining the element type for the profile. NO means that this International Standard does not specify this element type for the profile.

8.2.6 Server Profile Support for the Block Server Performance Subprofile

At the top of Figure 35: "Base Array Profile Block Server Performance Instance Diagram" is a dashed box that illustrates a part of the Server Profile for the Array. A similar dashed box appears for Storage Virtualizer and Volume Management Profiles. The part illustrated is the particulars for the Block Server Performance Subprofile. If performance support has been implemented, then there shall be a RegisteredSubprofile instance for the Block Server Performance Subprofile.

8.2.7 Default Manifest Collection

Associated with the instance of the StatisticsCollection shall be a provider supplied (Default) CIM_BlockStatisticsManifestCollection that represents the statistics properties that are kept by the profile. The default manifest collection is indicated by the IsDefault property (=True) of the CIM_BlockStatisticsManifestCollection. For each metered object of the profile implementation the default manifest collection will have exactly one manifest that will identify which properties are included for that metered object. If a an object is not metered, then there shall not be a manifest for that element type. If an element type (e.g., StorageVolume) is metered, then there shall be a manifest for that element type.

EXPERIMENTAL

Each default manifest in the default manifest collection identifies the statistics properties included by default by the implementation. The CSVSequence property of the manifest shall identify the default sequence in which the implementation will return statistics properties within each record for the ElementType on a GetStatisticsCollection request. For each property

included in the manifest by the value "true" there should be an entry in the CSVSequence array identifying the BlockStorageStatisticalData property by name. The first three values of CSVSequence shall be "InstanceID", "ElementType" and "StatisticsTime" to allow correlation of the Manifest with the CSV record based on the ElementType.

EXPERIMENTAL

8.2.8 Performance Additions applied to Multiple Computer Systems

Figure 38: "Multiple Computer System Subprofile Block Server Performance Instance Diagram" illustrates the class instances that would be supported if an Array, Storage Virtualizer or Volume Management Profile also implemented the Multiple Computer System Subprofile (and the Block Server Performance Subprofile). In this case, additional BlockStorageStatisticalData instances would exist for the component computer systems, as well as the top level computer system.

The "ElementsSupported" property of the BlockStatisticsCapabilities instance would include "Front-end Computer System", "Back-end Computer System" and/or "Peer Computer System".

Note that support for both the Multiple Computer System Subprofile and the Block Server Performance Subprofile does not imply support for statistics at the Component Computer System level. This support is ONLY implied by the "ElementsSupported" property of the BlockStatisticsCapabilities instance.

IECNORM.COM : Click to view the full PDF of ISO/IEC 24775-4:2014

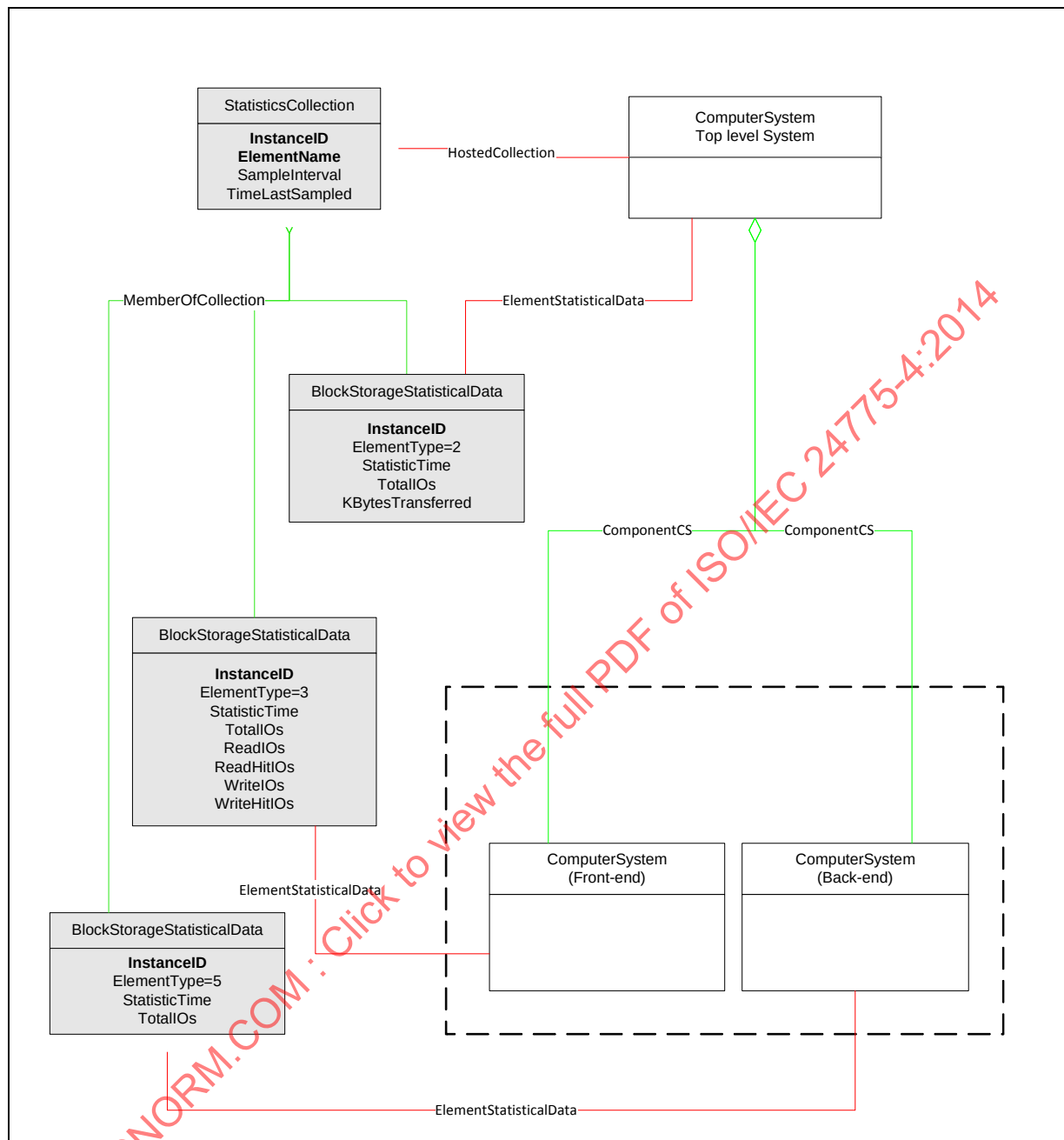


Figure 38 - Multiple Computer System Subprofile Block Server Performance Instance Diagram

NOTE The properties listed for the statistics classes are the mandatory properties. Optional Properties are not listed in order to save space in the diagram. Optional properties can be found in 8.8 "CIM Elements".

8.2.9 Performance Additions to Backend Ports

Figure 39: "Fibre Channel Initiator Port Subprofile Block Server Performance Instance Diagram" illustrates the class instances that would be supported if an Array also implemented the Fibre Channel Initiator Port Subprofile (and the Block Server Performance Subprofile). In this case, additional BlockStorageStatisticalData instances would exist for the back-end ports, as well as the front-end ports.

The "ElementsSupported" property of the BlockStatisticsCapabilities instance would include "Back-end Ports".

Note that support for both the Fibre Channel Initiator Port Subprofile and the Block Server Performance Subprofile DOES not imply support for statistics at the Back-end Port level. This support is ONLY implied by the "ElementsSupported" property of the BlockStatisticsCapabilities instance.

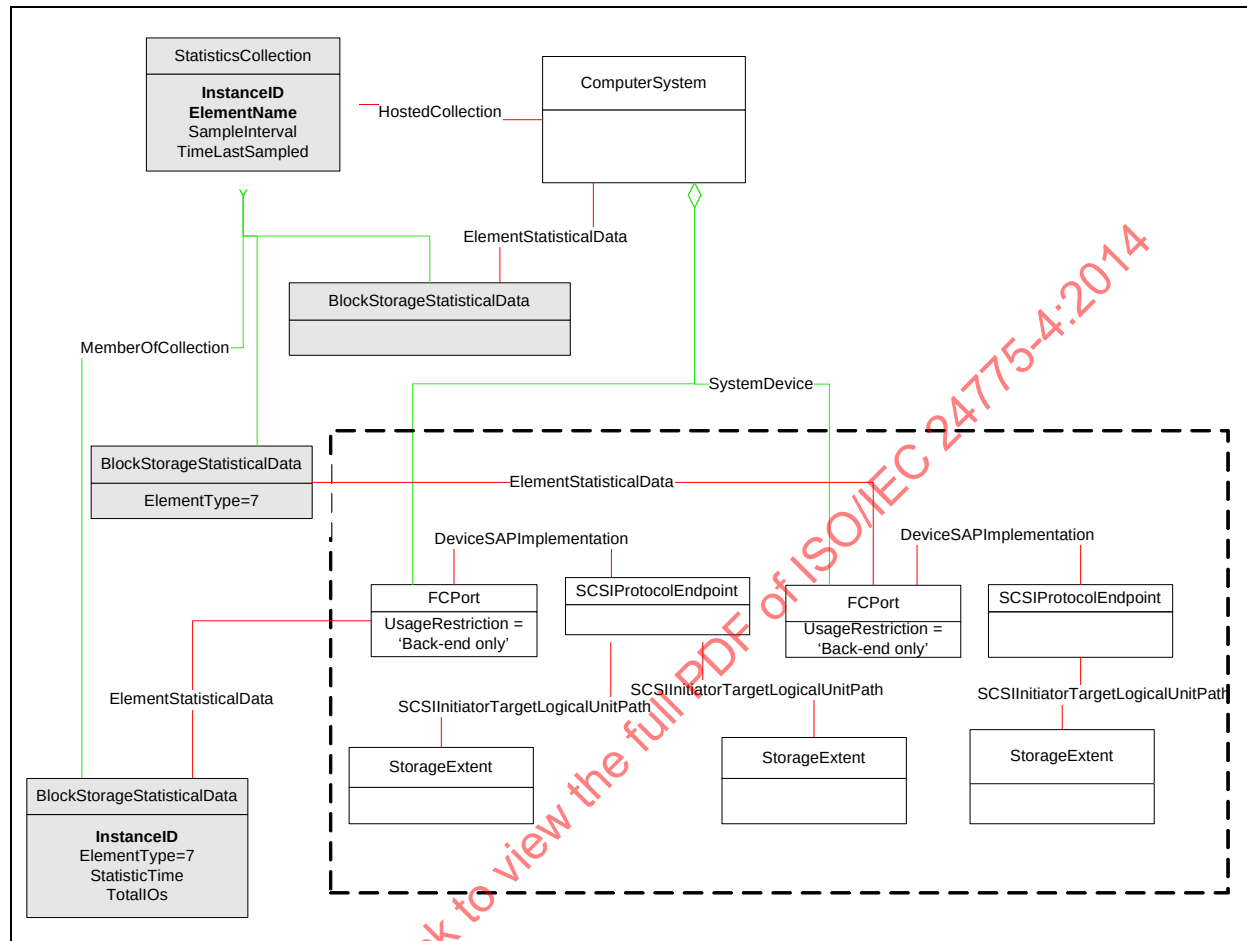


Figure 39 - Fibre Channel Initiator Port Subprofile Block Server Performance Instance Diagram

NOTE The properties listed for the statistics classes are the mandatory properties. Optional Properties are not listed in order to save space in the diagram. Optional properties can be found in 8.8 "CIM Elements".

EXPERIMENTAL

In some systems a port may be either a front-end or backend port. In this International Standard such ports would have a property that indicates that they serve both roles (UsageRestriction='4'). When a port has a UsageRestriction='4', then that port may have two BlockStorageStatisticalData records: one for the front-end port role and one for the backend port role. However, it will only have one record if only one of the port ElementTypes (6 or 7) is supported by the implementation. That is, if BlockStatisticsCapabilities.ElementTypes contains 6, but not 7, then the BlockStorageStatisticalData shall contain statistics for the front-end port role. If BlockStatisticsCapabilities.ElementTypes contains both 6 and 7, then there shall be two BlockStorageStatisticalData instances (one for the front-end port role and one for the backend port role).

EXPERIMENTAL

8.2.10 Performance Additions to Extent Composition

Figure 40: "Extent Composition Subprofile Block Server Performance Instance Diagram" illustrates the class instances that would be supported if an Array also implemented the Extent Composition Subprofile (and the Block Server Performance Subprofile). In this case, BlockStorageStatisticalData instances would exist for the Extents that are modeled.

The "ElementsSupported" property of the BlockStatisticsCapabilities instance would include "Extents".

Note that the Storage Virtualizer and Volume Management Profiles would use the "Extents" statistics for Storage Volumes (or LogicalDisks) that are imported instead of Disk extent statistics (since they do not have disk drives). Also note that an Array may model both "Extents" and "Disks" extents.

Further note that support for both the Extent Composition Subprofile and the Block Server Performance Subprofile DOES not imply support for statistics at the Extent level. This support is ONLY implied by the "ElementsSupported" property of the BlockStatisticsCapabilities instance.

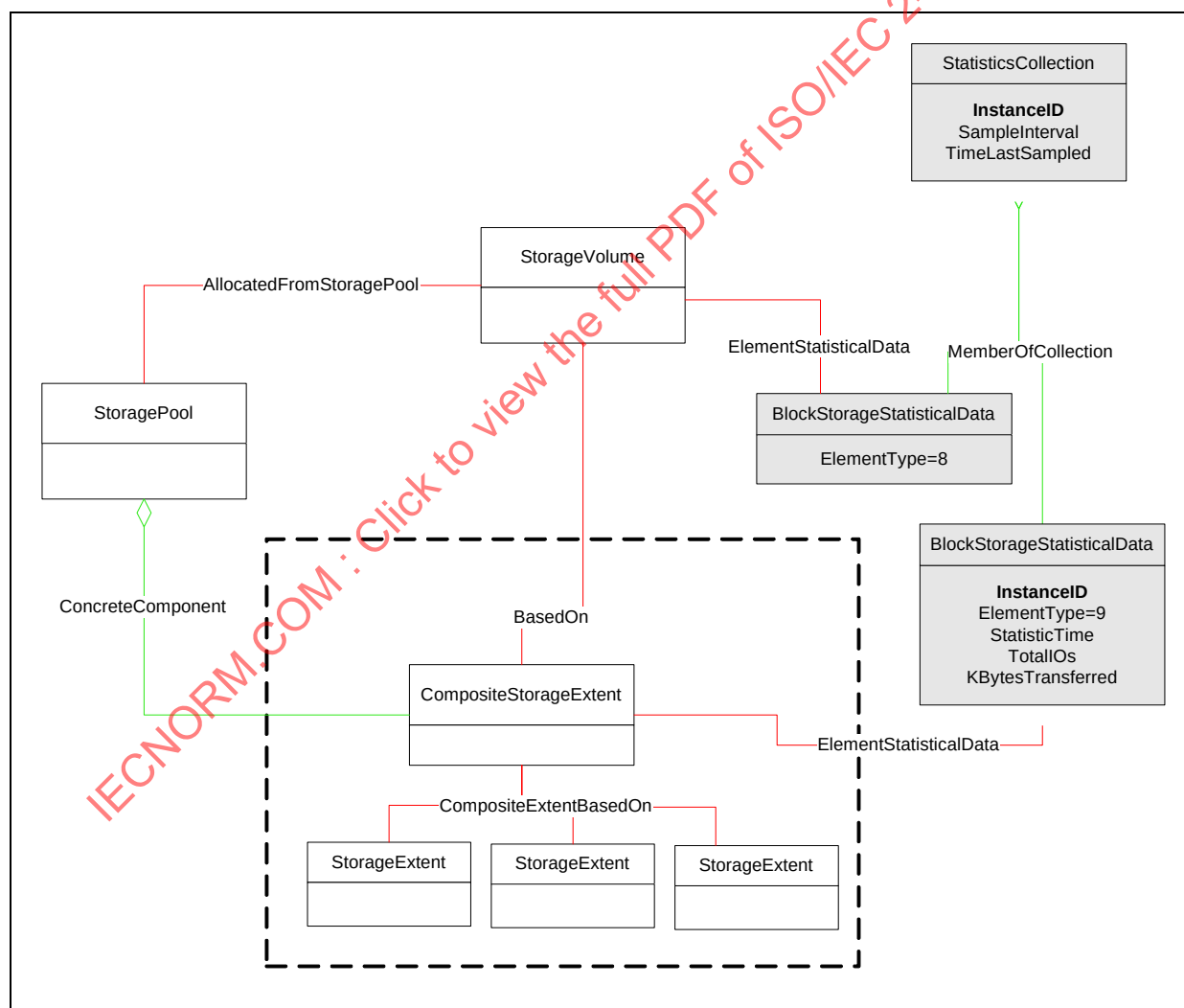


Figure 40 - Extent Composition Subprofile Block Server Performance Instance Diagram

NOTE 1 The properties listed for the statistics classes are the mandatory properties. Optional Properties are not listed in order to save space in the diagram. Optional properties can be found in 8.8 "CIM Elements".

NOTE 2 The low level extents represent Disk Drive Extents and they are not part of the Storage Virtualizer or Volume Management Profiles.

8.2.11 Performance Additions to Disk Drives

Figure 41: "Disk Drive Lite Subprofile Block Server Performance Instance Diagram" illustrates the class instances that would be supported if an Array also implemented the Disk Drive Lite (or Disk Drive) Subprofile (and the Block Server Performance Subprofile). In this case, BlockStorageStatisticalData instances would exist for each of the Disk Drives in the Array.

The "ElementsSupported" property of the BlockStatisticsCapabilities instance would include "Disks".

Note that the Volume Management Profiles would NEVER show the "Disks" statistics. Also note that an Array or Storage Virtualizer may model both "Extents" and "Disks".

Further note that support for both the Disk Drive Lite Subprofile and the Block Server Performance Subprofile DOES not imply support for statistics at the Disk Drive level. This support is ONLY implied by the "ElementsSupported" property of the BlockStatisticsCapabilities instance.

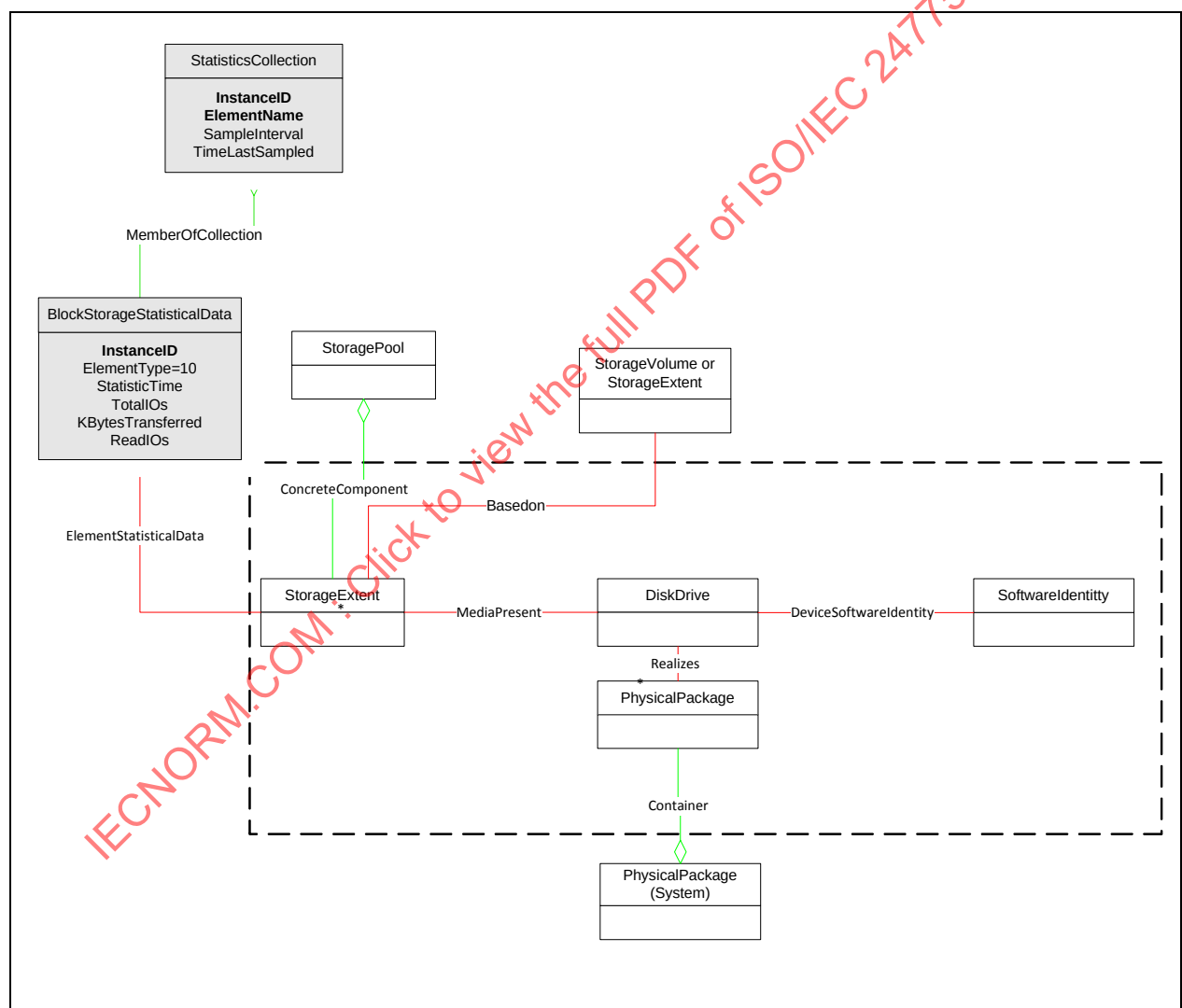


Figure 41 - Disk Drive Lite Subprofile Block Server Performance Instance Diagram

NOTE The properties listed for the statistics classes are the mandatory properties. Optional Properties are not listed in order to save space in the diagram. Optional properties can be found in 8.8 "CIM Elements".

8.2.12 Performance Additions to SCSIArbitraryLogicalUnits (Controller LUNs)

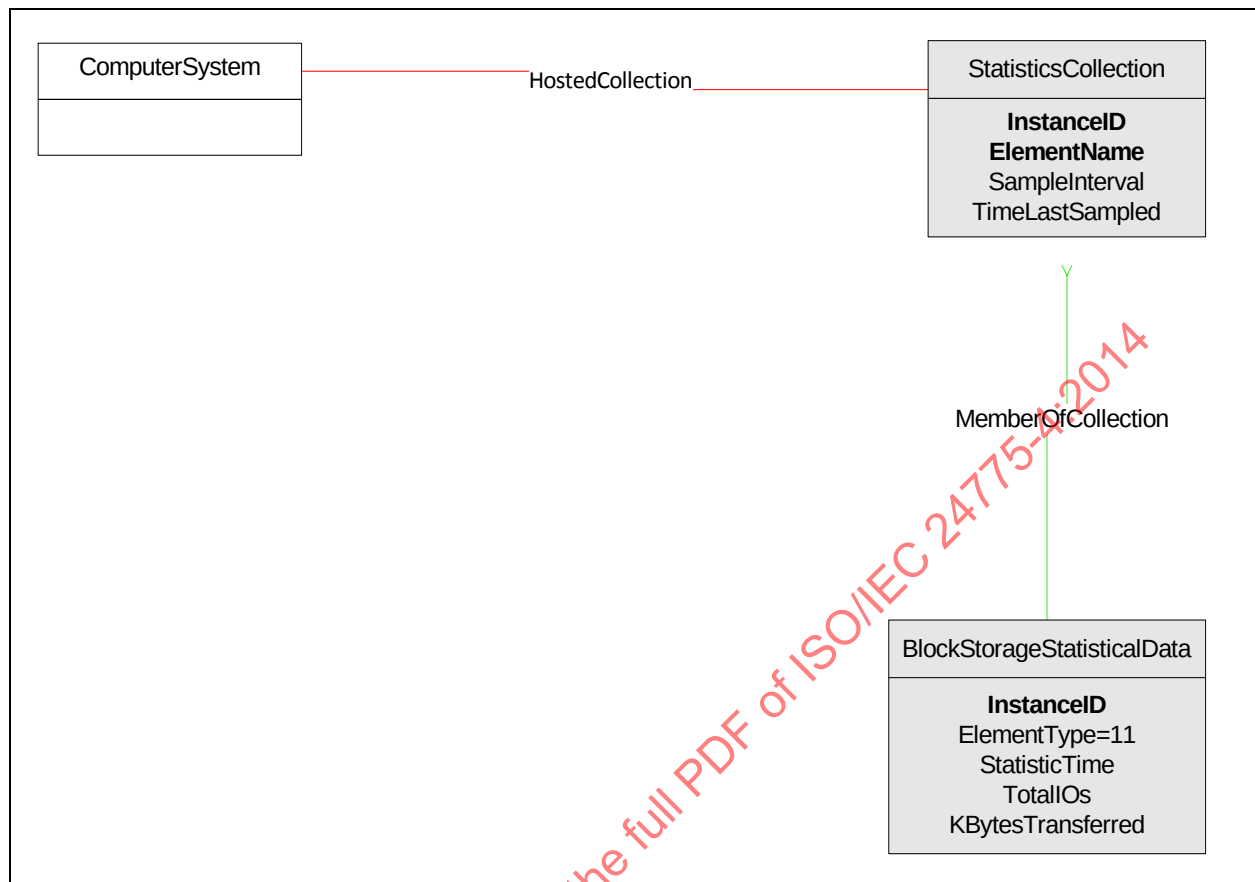


Figure 42 - SCSIArbitraryLogicalUnit Block Server Performance Instance Diagram

Figure 42: "SCSIArbitraryLogicalUnit Block Server Performance Instance Diagram" illustrates the class instances that would be supported if an Array (or Storage Virtualizer) has Controller LUNs (e.g., SCSIArbitraryLogicalUnits). In this case, BlockStorageStatisticalData instances would exist for each of the Controller LUNs (LogicalDevices or SCSIArbitraryLogicalUnits) supported by the Array (or Storage Virtualizer).

NOTE 1 There is no ElementStatisticalData association to any element. This is because the Controller LUNs are not actually part of the Array or Storage Virtualizer Profiles. But the statistics may still be collected in and kept in BlockStorageStatisticalData instances with ElementType=11.

The "ElementsSupported" property of the BlockStatisticsCapabilities instance would include "Arbitrary LUs".

NOTE 2 The properties listed for the statistics classes are the mandatory properties. Optional Properties are not listed in order to save space in the diagram. Optional properties can be found in 8.8 "CIM Elements".

EXPERIMENTAL

8.2.13 Performance Additions for Remote Mirrors

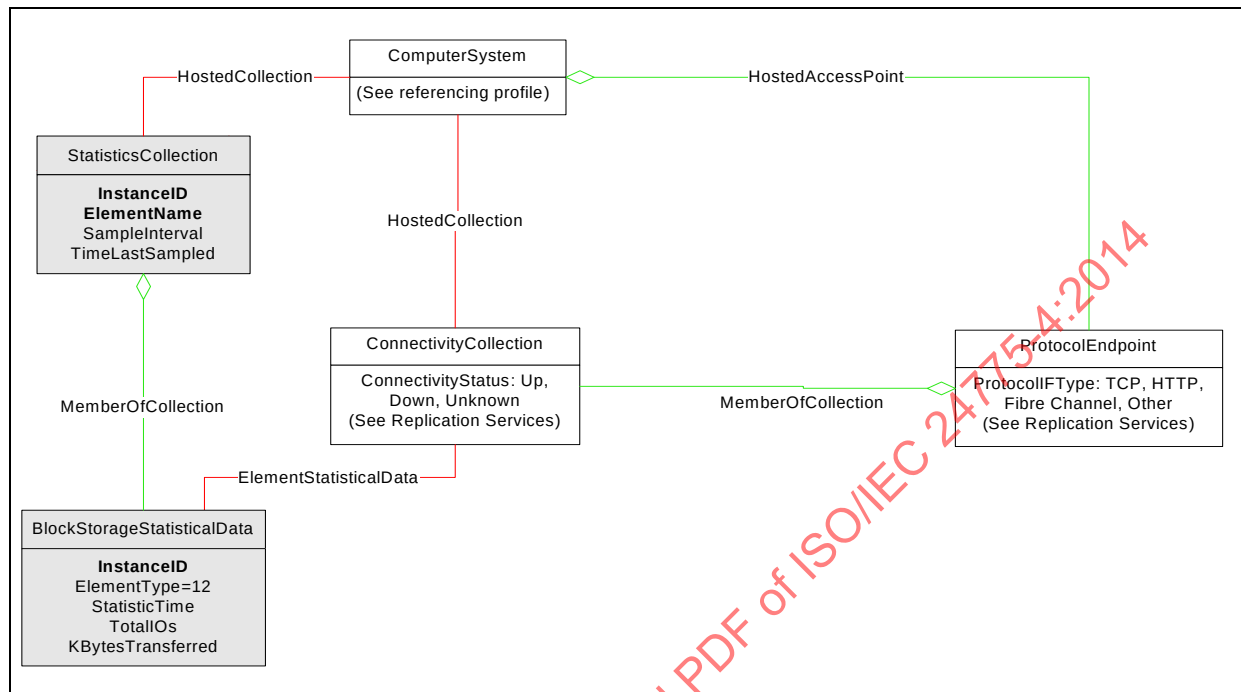


Figure 43 - Remote Mirrors Block Server Performance Instance Diagram

Figure 43: "Remote Mirrors Block Server Performance Instance Diagram" illustrates the class instances that would be supported if an Array also implemented the Remote Mirroring of the Replication Services Profile (and the Block Server Performance Subprofile). In this case, BlockStorageStatisticalData instances would exist for non-volume (e.g., metadata) IO requests. In this case, the BlockStorageStatisticalData instance is associated with the ConnectivityCollection instance that represents the connection to the remote system.

Note that statistics attributed to the ConnectivityCollection are control IOs between the mirroring arrays. Statistics that actually move data to the remote mirror are attributed to the targeted StorageVolume (or logical disk).

The "ElementsSupported" property of the BlockStatisticsCapabilities instance would include "Remote Replica Group".

Note that support for both the Replication Services Profile and the Block Server Performance Profile DOES not imply support for statistics at the Remote Replica Group level. This support is ONLY implied by the "ElementsSupported" property of the BlockStatisticsCapabilities instance.

NOTE The properties listed for the statistics classes are the mandatory properties. Optional Properties are not listed in order to save space in the diagram. Optional properties can be found in 8.8 "CIM Elements".

EXPERIMENTAL

8.2.14 Client Defined Manifest Collections

Manifest collections are either provider supplied (CIM_BlockStatisticsManifestCollection.IsDefault=True) for the profile implementation or client defined collections (CIM_BlockStatisticsManifestCollection.IsDefault=False) that indicate what statistics properties the client would like to retrieve using the GetStatisticsCollection method. For a discussion of provider supplied manifest collections, see 8.2.7.

Client defined manifest collections are a mechanism for restricting the amount of data returned on a GetStatisticsCollection request. A client defined manifest collection is identified by the IsDefault property of the collection is set to False. For each block statistics class (e.g., Computer System, Volume, Disk, etc.) a manifest can be defined which identifies which properties of the particular statistics class are to be returned on a GetStatisticsCollection request. Each of the classes of block statistic may have 0 or 1 manifest in any given manifest collection.

EXPERIMENTAL

In addition to identifying which properties the client wants returned, the client may define the sequence in which the properties are to be returned with the CSVSequence property of the manifest. Support for this function is conditional on BlockStatisticsCapabilities.SupportedFeatures including the value '3' (Client Defined Sequence). If the client does not set this property or sets it improperly, the implementation shall set the value of CSVSequence to NULL. If the SupportedFeatures does not include the value '3' the implementation will set the CSVSequence to NULL (implying the default sequence will be used).

EXPERIMENTAL

This is illustrated in Figure 44: "Block Server Performance Manifest Collections".

IECNORM.COM : Click to view the full PDF of ISO/IEC 24775-4:2014

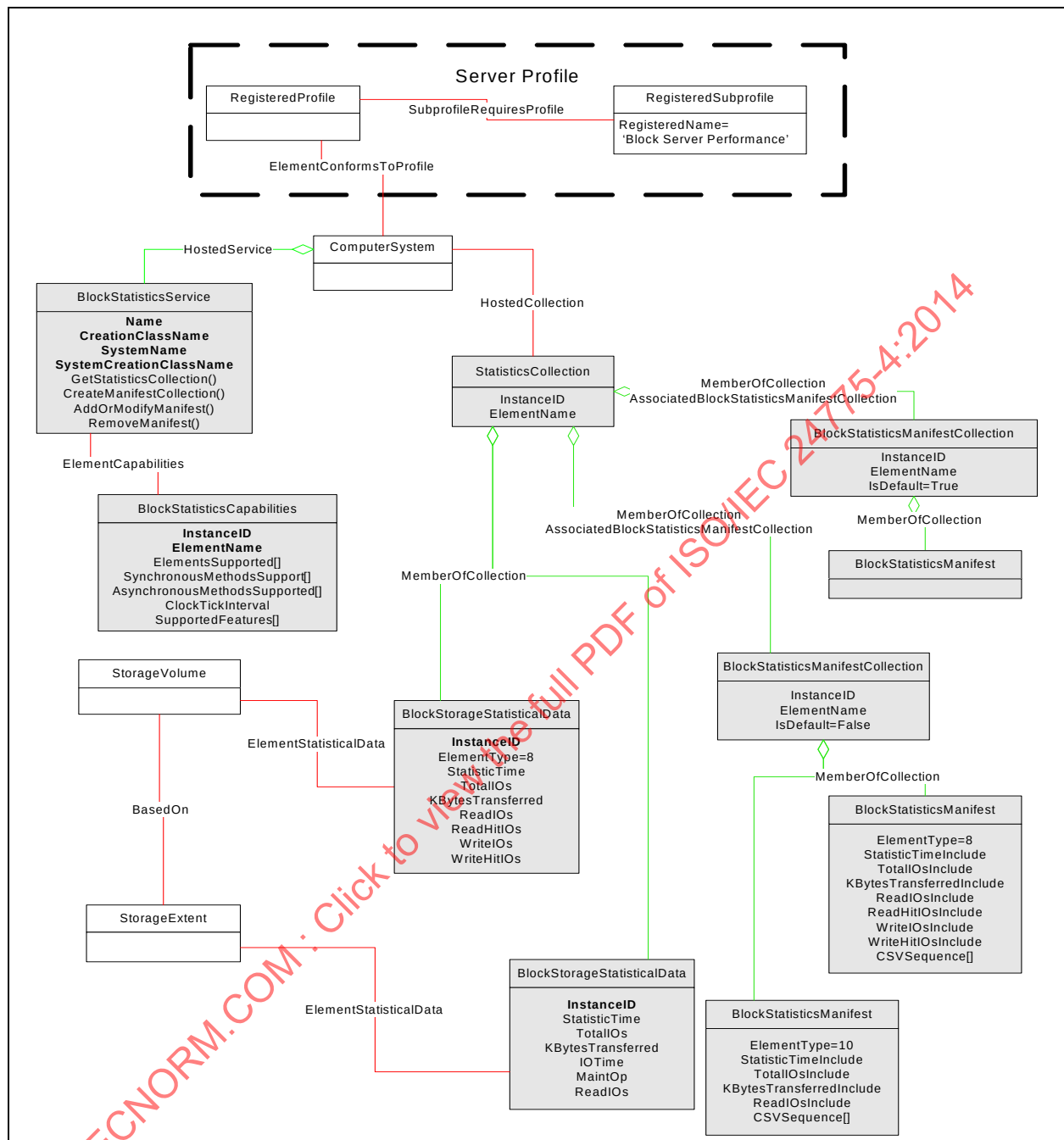


Figure 44 - Block Server Performance Manifest Collections

In this figure, manifest classes are defined for Volumes (StorageVolumes or LogicalDisks) and Disk Drives. Each property of the manifest is a Boolean that indicates whether the property is to be returned (true) or omitted (false).

Multiple client defined manifest collections can be defined in the profile. So different clients or different client applications can define different manifests for different application needs. A manifest collection can completely omit a whole class of statistics (e.g., no ComputerSystem statistics are shown in Figure 44: "Block Server Performance Manifest Collections"). Since manifest collections are "client objects", they are named (ElementName) by the client for the client's convenience. The CIM server will generate an instance ID to uniquely identify the manifest collection in the CIM Server.

Client defined manifest collections are created using the CreateManifestCollection method. Manifests are added or modified using the AddOrModifyManifest method. A manifest may be removed from the manifest collection using the RemoveManifest method.

NOTE Use of manifest collections is optional with the GetStatisticsCollection method. If NULL for the manifest collection is passed on input, then all statistics instances are assumed.

8.2.15 Capabilities Support for Block Server Performance Subprofile

8.2.15.1 General

There are two dimensions to determining what is supported with a Block Server Performance Subprofile implementation. First, there are the RegisteredSubprofiles supported by the Block server (Array, Storage Virtualizer or Volume Management Profile). In order to support statistics for a particular class of metered element, the corresponding object shall be modeled. So, if an Array has not implemented the Disk Drive Lite (or Disk Drive) Subprofile, then it shall not implement the BlockStorageStatisticalData for Disk Drives in the Block Server Performance Subprofile (and implementation of the Disk Drive Lite or Disk Drive Subprofile does not guarantee implementation of the BlockStorageStatisticalData for disk drives).

Both of these dimensions are captured in the BlockStatisticsCapabilities class instance. This is populated by the provider (not created or modified by Clients). The second dimension is techniques supported for retrieving statistics and manipulating manifest collections.

8.2.15.2 ElementsSupported

The values of interest are "Computer System", "Front-end Computer System", "Peer Computer System", "Back-end Computer System", "Front-end Port", "Back-end Port", "Volume", "Extent", "Disk Drive", "Arbitrary LUs", "Remote Replica Group".

8.2.15.3 SynchronousMethodsSupported

The values of interest are "Exec Query", "Indications", "Query Collection", "GetStatisticsCollection", "Manifest Creation", "Manifest Modification", and "Manifest Removal".

8.2.15.4 AsynchronousMethodsSupported

For the current version of the International Standard this should be NULL.

EXPERIMENTAL

8.2.15.5 SupportedFeatures

The values of interest are "none" and "Client Defined Sequence".

EXPERIMENTAL

8.2.15.6 ClockTickInterval

An internal clocking interval for all timer counters kept in the subsystem, measured in microseconds (Unit of measure in the timers, measured in microseconds). Time counters are monotonically increasing counters that contain 'ticks'. Each tick represents one ClockTickInterval.

To be a valid implementation of the Block Server Performance Subprofile, at least one of the values listed for ElementsSupported shall be supported. ElementsSupported is an array, such that all of the values can be identified.

For the methods supported properties any or all of these values can be missing (e.g., the arrays can be NULL). If all the methods supported are NULL, this means that manifest collections are not supported and neither GetStatisticsCollection nor Query are supported for retrieval of statistics. This leaves enumerations or association traversals as the only methods for retrieving the statistics.

8.3 Health and Fault Management Considerations

Not defined in this International Standard.

8.4 Cascading Considerations

Not applicable.

8.5 Supported Subprofiles and Packages

See 8.1.1 for the list of supported profiles and packages.

8.6 Methods of the Profile

8.6.1 Extrinsic Methods of the Profile

8.6.1.1 Overview

The methods supported by this subprofile are summarized in Table 116, and detailed in the subclauses that follow it.

Table 116 - Creation, Deletion and Modification Methods in Block Server Performance Subprofile

Method	Created Instances	Deleted Instances	Modified Instances
GetStatisticsCollection	None	None	None
CreateManifestCollection	BlockStatisticsManifestCollection AssociatedBlockStatisticsManifestCollection	None	None
AddOrModifyManifest	BlockStatisticsManifest (subclass) MemberOfCollection	None	BlockStatisticsManifest (subclass)
RemoveManifest	None	BlockStatisticsManifest (subclass) MemberOfCollection	None

8.6.1.2 GetStatisticsCollection

This method retrieves statistics in a well-defined bulk format. The set of statistics returned by this list is determined by the list of element types passed in to the method and the manifests for those types contained in the supplied manifest collection. The statistics are returned through a well-defined array of strings that can be parsed to retrieve the desired statistics as well as limited information about the elements that those metrics describe.

GetStatisticsCollection(

[IN (false), OUT, Description(Reference to the job (shall be null in the current version of SMI-S).)]

CIM_ConcreteJob REF **Job**,

[IN, Description(Element types for which statistics should be returned.)

ValueMap { "0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "..", "32768..65535" },

Values { "Unknown", "Reserved", "Computer System", "Front-end Computer System",

"Peer Computer System", « Back-end Computer System "Front-end Port", "Back-end Port",

"Volume", "Extent", "Disk Drive", "Arbitrary LUs", "Remote Replica Group",

"DMTF Reserved", "Vendor Specific"]]

uint16 **ElementTypes**[],

[IN, Description(The manifest collection that contains the manifests that list the metrics that

should be returned for each element type.)]

CIM_BlockStatisticsManifestCollection REF **ManifestCollection**,

[IN, Description("Specifies the format of the Statistics output parameter.")

ValueMap { "2" },

Values ("CSV")]

Uint16 **StatisticsFormat**,

[OUT, Description(The statistics for all the elements as determined by the Elements and

ManifestCollection parameters.)]

string **Statistics**[]);

Error returns are:

{ "Job Completed with No Error", "Not Supported", "Unknown", "Timeout", "Failed", "Invalid Parameter", "Method Reserved", "Method Parameters Checked - Job Started", "Element Not Supported", "Statistics Format Not Supported", "Method Reserved", "Vendor Specific"}

In this International Standard, Job Control is not supported for the GetStatisticsCollection method. This method should always return NULL for the Job parameter.

If the ElementTypes[] array is empty, then no data is returned. If the ElementTypes[] array is NULL, then all data specified in the manifest collection is returned.

If the manifest collection is empty, then no data is returned. If the manifest collection parameter is NULL, then the default manifest collection is used (Note that in SMI-S, a default manifest collection shall exist if the GetStatisticalCollection method is supported).

NOTE The ElementTypes[] and ManifestCollection parameters may identify different sets of element types. The effect of this will be for the implementation to return statistics for the element types that are in both lists (that is, the intersection of the two lists). This intersection could be empty. In this case, no data will be returned.

For the current version of SMI-S, the only recognized value for StatisticsFormat is "CSV". The method may support other values, but they are not specified by SMI-S (i.e., they would be vendor specific).

Given a client has an inventory of the metered objects with Statistics InstanceIDs that may be used to correlate with the BlockStorageStatisticalData instances, a simple CSV format is sufficient and the most efficient human-readable format for transferring bulk statistics. More specifically, the following rules constrain that format and define the content of the String[] Statistics output parameter to the GetStatisticsCollection() method:

- The Statistics[] array may contain multiple statistics records per array entry. In such cases, the total length of the concatenated record strings will not exceed 64 KB. A single statistics record will not span Array entries.

- There shall be exactly one statistics record per line in the bulk Statistics parameter. A line is terminated by:
 - a line-feed character
 - the end of a String Array Element (i.e., a statistics record cannot overlap elements of the String[] Statistics output parameter).
- Each statistics record shall contain the InstanceID of the BlockStorageStatisticalData instance, the value map (number) of the ElementType of the metered object and one value for each property that the relevant BlockStatisticsManifest specifies as "true".
- Each value in a record shall be separated from the next value by a Semi-colon (";"). This is to support internationalization of the CSV format. A provider creating a record in this format should not include white space between values in a record. A client reading a record it has received would ignore white-space between values.
- The InstanceID value is an opaque string that shall correspond to the InstanceID property from BlockStorageStatisticalData instance.
 - For the convenience of client software, that need to be able to correlate InstanceIDs between different GetStatisticsCollection method invocations, the InstanceID for BlockStorageStatisticalData instance shall be unique across all instances of the BlockStorageStatisticalData class. It is not sufficient that InstanceID is unique across subclasses of BlockStorageStatisticalData.
- The ElementType value shall be a decimal string representation of the Element Type number (e.g., "8" for StorageVolume). The StatisticTime shall be a string representation of DateTime. All other values shall be decimal string representations of their statistical values.
- NULL values shall be included in records for which a statistic is returned (specified by the manifest or by a lack of manifest for a particular element type) but there is no meaningful value available for the statistic. A NULL statistic is represented by placing a semi-colon (";") in the record without a value in the position the value would have otherwise been included. A record in which the last statistic has a NULL value shall end in a semi-colon (";").

DEPRECATED

- The first three values in a record shall be the InstanceID, ElementType and StatisticTime values from the BlockStorageStatisticalData instance. The remaining values shall be returned in the order in which they are defined by the MOF for the BlockStatisticsManifest class or subclass the record describes.

DEPRECATED

EXPERIMENTAL

- Use of the MOF for defining the sequence of statistics in a record has proven to be an unreliable means of defining the sequence of statistics in each record. If the CSVSequence is non-NULL, then the sequence of statistics will be defined by the sequence of entries in the CSVSequence array. The first three values in the CSVSequence shall be "InstanceID", "ElementType" and "StatisticTime". All other elements of the CSVSequence array may be in the order defined by the creator of the Manifest. If the CSVSequence is NULL in the Default (provider) Manifest, then the rule in the previous bullet still applies.

EXPERIMENTAL

As an additional convention, a provider should return all the records for a particular element type in consecutive String elements, and the order of the element types should be the same as the order in which the element types were specified in the input parameter to `GetStatisticsCollection()`.

Example output as it might be transmitted in CIM-XML. It shows records for 5 Volumes and 5 disks, assuming that 6 statistics were specified in the `BlockStatisticsManifest` instance for both disks and volumes. The sixth statistic is unavailable for volumes, and the fourth statistic is unavailable for disks:

```
<METHODRESPONSE NAME="GetStatisticsCollection">
  <RETURNVALUE PARAMTYPE="uint32">
    <VALUE>
      0
    </VALUE>
  </RETURNVALUE>
  <PARAMVALUE NAME="Statistics" PARAMTYPE="string">
    <VALUE.ARRAY>
      <VALUE>
        STORAGEVOLUMESTATS1;7;20040811133015.0000010-
          300;11111;22222;33333;44444;55555;
        STORAGEVOLUMESTATS2;7;20040811133015.0000020-
          300;11111;22222;33333;44444;55555;
        STORAGEVOLUMESTATS3;7;20040811133015.0000030-
          300;11111;22222;33333;44444;55555;
        STORAGEVOLUMESTATS4;7;20040811133015.0000040-
          300;11111;22222;33333;44444;55555;
        STORAGEVOLUMESTATS5;7;20040811133015.0000050-
          300;11111;22222;33333;44444;55555;
      </VALUE>
      <VALUE>
        DISKSTATS1;9;20040811133015.0000100-300;11111;22222;33333;;55555;66666
        DISKSTATS2;9;20040811133015.0000110-300;11111;22222;33333;;55555;66666
        DISKSTATS3;9;20040811133015.0000120-300;11111;22222;33333;;55555;66666
        DISKSTATS4;9;20040811133015.0000130-300;11111;22222;33333;;55555;66666
        DISKSTATS5;9;20040811133015.0000140-300;11111;22222;33333;;55555;66666
      </VALUE>
    </VALUE.ARRAY>
  </PARAMVALUE>
</METHODRESPONSE>
```

8.6.1.3 CreateManifestCollection

Creates a new manifest collection whose members serve as a filter for metrics retrieved through the `GetStatisticsCollection` method.

CreateManifestCollection(

[IN, Description(The collection of statistics that will be filtered using the new manifest collection.)]

CIM_StatisticsCollection REF **Statistics**,

[IN, Description(Client-defined name for the new manifest collection)]

string **ElementName**,

[OUT, Description(Reference to the new manifest collection.)]

CIM_BlockStatisticsManifestCollection REF **ManifestCollection**);

Error returns are:

{ "Ok", "Not Supported", "Unknown", "Timeout", "Failed", "Invalid Parameter", "Method Reserved", "Vendor Specific" }

8.6.1.4 AddOrModifyManifest

This is an extrinsic method that either creates or modifies a statistics manifest for this statistics service. A client supplies a manifest collection in which the new manifest collection will be placed or an existing manifest will be modified, the element type of the statistics that the manifest will filter, and a list of statistics that should be returned for that element type using the GetStatisticsCollection method.

AddOrModifyManifest(

[IN, Description(Manifest collection that the manifest is or should be a member of.)]

CIM_BlockStatisticsManifestCollection REF **ManifestCollection**,

[IN, Description(The element type whose statistics the manifest will filter.)

ValueMap { "0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "..", "32768..65535" },

Values { "Unknown", "Reserved", "Computer System", "Front-end Computer System",

"Peer Computer System", « Back-end Computer System "Front-end Port", "Back-end Port",

"Volume", "Extent", "Disk Drive", "Arbitrary LUs", "Remote Replica Group",

"DMTF Reserved", "Vendor Specific" }}

uint16 **ElementType**,

[IN, Description(The client-defined string that identifies the manifest created or modified by this method.)]

string **ElementName**,

[IN, Description(The statistics that will be supplied through the GetStatisticsCollection method.)]

string **StatisticsList[]**,

[OUT, Description(The Manifest that is created or modified on successful execution of the method.)]

CIM_BlockStatisticsManifest REF **Manifest**);

Error returns are:

{ "Success", "Not Supported", "Unknown", "Timeout", "Failed", "Invalid Parameter", "Method Reserved", "Element Not Supported", "Metric not supported", "ElementType Parameter Missing", "Method Reserved", "Vendor Specific" }

If the StatisticsList[] array is empty, then only InstanceID and ElementType will be returned when the manifest is referenced. If the StatisticsList[] array parameter is NULL, then all supported properties is assumed.

NOTE This would be the BlockStatisticsManifest from the default manifest collection.

EXPERIMENTAL

The sequence of properties identified in StatisticsList[] shall be used to fill in the CSVSequence array in the manifest if BlockStatisticsCapabilities.SupportedFeatures includes the value '3' (Client Defined Sequence). Otherwise the CSVSequence array will be set to NULL.

EXPERIMENTAL

8.6.1.5 RemoveManifest

This is an extrinsic method that removes manifests from a manifest collection.

RemoveManifest(

[IN, Description(Manifest collection from which the manifests will be removed.)]

CIM_BlockStatisticsManifestCollection REF **ManifestCollection**,

[IN, Description(List of manifests to be removed from the manifest collection.)]

CIM_BlockStatisticsManifest REF **Manifests[]**);

Error returns are:

{ "Success", "Not Supported", "Unknown", "Timeout", "Failed", "Invalid Parameter", "Method Reserved", "Manifest not found", "Method Reserved", "Vendor Specific" }

8.6.2 Intrinsic Methods of the Profile

8.6.2.1 Overview

NOTE Basic Write intrinsic methods are not specified for StatisticsCollection, HostedCollection, BlockStorageStatisticalData, MemberOfCollection or ElementStatisticalData.

8.6.2.2 DeleteInstance (of a CIM_BlockStatisticsManifestCollection)

DeleteInstance will delete the CIM_BlockStatisticsManifestCollection where IsDefault=False, the CIM_AssociatedBlockStatisticsManifestCollection association to the StatisticsCollection and all manifests collected by the manifest collection (and the MemberOfCollection associations to the CIM_BlockStatisticsManifestCollection).

8.6.2.3 Association Traversal

One of the ways of retrieving statistics is through association traversal from the StatisticsCollection to the individual Statistics following the MemberOfCollection association. This shall be supported by all implementations of the Block Server Performance Subprofile and would be available to clients if the provider does not support EXEC QUERY or GetStatisticsCollection approaches.

EXPERIMENTAL

8.6.2.4 CreateInstance (of a ListenerDestinationCIMXML, IndicationSubscription and possibly IndicationFilters)

CreateInstance would be required to establish subscriptions and ListenerDestinations for retrieval of statistics via indications. Depending on the support in the profile, it may also be required to create the IndicationFilter.

8.6.2.5 DeleteInstance (of a ListenerDestinationCIMXML, IndicationSubscription and possibly IndicationFilters)

DeleteInstance would be required to delete subscriptions and ListenerDestinations that were defined for retrieval of statistics via indications. Depending on the support in the profile, it may also be required to delete the IndicationFilter.

8.6.2.6 ModifyInstance (of an IndicationFilter)

ModifyInstance may also be supported for modifying IndicationFilters, assuming the profile supports client defined filters. It would not be supported for “pre-defined” filters.

8.6.2.7 EXEC QUERY

This is one of the ways of retrieving statistics.

8.6.2.8 GetInstance on QueryStatisticalCollection

This is yet another means of retrieving statistics. In this technique an instance of the QueryStatisticalCollection class is created that defines a Query for statistics and the format in which the query results are to be represented. The key properties of the QueryStatisticalCollection class are:

- Query - This is a query string that defines the statistics to be populated in the QueryStatisticalCollection instance.
- QueryLanguage - This defines the query language that is used in the query. For the current version of SMI-S, only CQL should be encoded.
- SelectedEncoding - This defines the encoding of the data that is to be populated in the QueryStatisticalCollection instance. For the current version of SMI-S, this should be CSV (for Comma Separated Values).
- SelectedNames - This is the list of statistics property names to be retrieved. These correspond to the Select List of the Query. The encoding of these names is as defined by the SelectedEncoding (for the current version of SMI-S, this would be CSV).
- SelectedTypes - This is the list of data types for the columns of the query result. Each data type specified corresponds to a column in the SelectedValues property.
- SelectedValues - This is a table of values that correspond to the query results (for the query specified in the Query property). The data types of the column of values are defined by SelectedTypes. The name of each column in the table is defined by SelectedNames. The values are encoded as defined by SelectedEncoding (i.e., CSV for the current version of SMI-S).

An example CQL query would be:

```
SELECT Stats.*
FROM CIM_BlockStorageStatisticalData Stats,
      CIM_QueryStatisticsCollection QSC,
      CIM_MemberOfCollection MoC
WHERE ObjectPath(QSC) = ObjectPath(SELF)
      AND ObjectPath(QSC) = MoC.Collection
      AND ObjectPath(Stats) = MoC.Member
      AND CurrentDateTime() >=
         Stats.StatisticTime + Stats.SampleInterval
```

A client would define a QueryStatisticalCollection instance as means of specifying what the client wants. This would be done with the CreateInstance intrinsic method. The client would delete such an instance using the DeleteInstance method. If the client wishes to change the query, the client would use the ModifyInstance intrinsic method.

Retrieving the data would be done via the `GetInstance` intrinsic. This would retrieve the `QueryStatisticalCollection` instance, which includes the table of comma separated values which are the statistics.

EXPERIMENTAL

8.7 Client Considerations and Recipes

8.7.1 Bulk Performance Statistics Gathering

```
// DESCRIPTION
//
// This recipe describes how to determine what elements are metered, what
// retrieval methods are supported and what statistics are kept for each
// metered element in Arrays, Storage Virtualizers or Volume Managers
// that
// support the Block Server Performance Subprofile and how to retrieve
// the
// statistical data.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTIONS:
// 1. The names of the top-level ComputerSystem instances for Array,
// Storage
// Virtualizer, or Volume Manager implementations supporting the Block
// Server
// Performance Subprofile have previously been discovered via SLP and are
// known
// as $StorageSystems->[].
//
// Function GetNumStatsIncluded
//
// This function counts of the number of metrics that should be included
// in a
// statistics record built using the supplied BlockManifest instance.
//
sub GetNumStatsIncluded($BlockManifest) {
    #numIncluded = 0
    if ($BlockManifest.IncludeStatisticTime)
        #numIncluded++
    if ($BlockManifest.IncludeTotalIOs)
        #numIncluded++
    if ($BlockManifest.IncludeKBytesTransferred)
        #numIncluded++
    if ($BlockManifest.IncludeIOTimeCounter)
        #numIncluded++
    if ($BlockManifest.IncludeReadIOs)
        #numIncluded++
    if ($BlockManifest.IncludeReadHitIOs)
        #numIncluded++
    if ($BlockManifest.IncludeReadIOTimeCounter)
        #numIncluded++
}
```

```
if ($BlockManifest.IncludeReadHitIOTimeCounter)
    #numIncluded++
if ($BlockManifest.IncludeKBytesRead)
    #numIncluded++
if ($BlockManifest.IncludeWriteIOs)
    #numIncluded++
if ($BlockManifest.IncludeWriteHitIOs)
    #numIncluded++
if ($BlockManifest.IncludeWriteIOTimeCounter)
    #numIncluded++
if ($BlockManifest.IncludeWriteHitIOTimeCounter)
    #numIncluded++
if ($BlockManifest.IncludeKBytesWritten)
    #numIncluded++
if ($BlockManifest.IncludeIdleTimeCounter)
    #numIncluded++
if ($BlockManifest.IncludeMaintOp)
    #numIncluded++
if ($BlockManifest.IncludeMaintTimeCounter)
    #numIncluded++
return #numIncluded
}

// Function ValidateRecords
//
// This function validates the records of a set of statistics supplied in
// the
// Bulk Statistics Format defined in the Block Server Performance
// Subprofile.
// Every statistics record should contain an InstanceID, ElementType and
// the
// number of statistics indicated by the BlockManifest. This functional
// verifies that a non-empty InstanceID was specified and that the format
// of
// metrics populated is appropriate for the data type defined each
// supported
// metric. It also checks if the metrics are null, which could occur if
// a
// client included a metric in the BlockManifest used by the
// GetStatisticsCollection function that cannot be populated.
sub ValidateRecords(#BulkStatistics[],
    $BlockManifests[],
    $BSSDs[]) {

    for (#i in #BulkStatistics[]) {

        // The function split() below parses the content of an element in
        // #BulkStatistics[] into multiple sub-strings based on occurrences
        // of carriage return. (i.e. "\n")
        #Records[] = #BulkStatistics[#i].split("\n")
        for (#j in #Records[]) {
```

```

// The function split() below further parses the content of an
// element in #Records[] into multiple sub-strings based on
// occurrences of semi-colon. The resulting elements contain (in
// order) the InstanceID and ElementType properties followed by
// the
// metrics supported.
#RecordElements[] = #Records[#j].split(";")

// Each element MUST contain at least InstanceID and
// ElementType.
if (#RecordElements[].length < 2) {
<ERROR! Statistics Record does not contain InstanceID and/or
  ElementType>
}
// The InstanceID in the record MUST match the InstanceID of a
// BSSD.
$StatsData = null
for (#k in $BSSDs[]) {
if ($BSSDs[#k]->InstanceID == #RecordElements[0]) {
    $StatsData = $BSSDs[#k]
    break
}
}
if ($StatsData == null) {
<ERROR! Statistics instance could not be found for record>
}

// The function Integer() below is used to convert a string
// representation of an integer to an int value.
#elementType = Integer(#RecordElements[1])
if (#elementType != $StatsData.ElementType) {
<ERROR! ElementTypes for statistics record and instance do not
  match>
}

// Get the BlockManifest that describes this record. If none
// exists
// then the record does not contain a valid ElementType.
$BlockManifest = &GetBlockManifestByType($BlockManifests[],
  #elementType)
if ($BlockManifest == null) {
<ERROR! ElementType in Statistics Record not recognized>
}

// There MUST be two elements in the record (i.e. InstanceID and
// ElementType) in addition to one element for each supported
// metric.
if (#RecordElements.length !=
  &GetNumStatsIncluded($BlockManifest) + 2) {
<ERROR! Statistics record does not contain the expected number
  of metrics>
}

```

```
// All default manifests MUST contain StatisticTime
if (!$BlockManifest.IncludeStatisticTime) {
<ERROR! Default manifest does not specify required property
    value IncludeStatisticTime=true>
}

// The function Datetime() below is used to convert a string
// representation of a DateTime value into a DateTime object
#StatisticTime = Datetime(#RecordElements[2])

// Copy instance for local modification
$Manifest = $BlockManifest
// Validate the metrics in each record
#CurrentProperty = 0
#CurrentPropertyName = "Unknown"
#k = 3
while (#k < #RecordElements[.length]) {
// The remaining record elements should be integral values
// Parse the next element in the record and save the relevant
// property from the BSSD instance (and its name for inclusion
// in error codes)
if ($Manifest.IncludeTotalIOs) {
    #CurrentProperty = $StatsData.TotalIOs
    #CurrentPropertyName = "TotalIOs"

    // Avoid double-checking for inclusion of this metric
    $Manifest.IncludeTotalIOs = false
} else if ($Manifest.IncludeKBytesTransferred) {
    #CurrentProperty = $StatsData.KBytesTransferred
    #CurrentPropertyName = "KBytesTransferred"

    // Avoid double-checking for inclusion of this metric
    $Manifest.IncludeKBytesTransferred = false
} else if ($Manifest.IncludeIOTimeCounter) {
    #CurrentProperty = $StatsData.IOTimeCounter
    #CurrentPropertyName = "IOTimeCounter"

    // Avoid double-checking for inclusion of this metric
    $Manifest.IncludeIOTimeCounter = false
} else if ($Manifest.IncludeReadIOs) {
    #CurrentProperty = $StatsData.ReadIOs
    #CurrentPropertyName = "ReadIOs"

    // Avoid double-checking for inclusion of this metric
    $Manifest.IncludeReadIOs = false
} else if ($Manifest.IncludeReadHitIOs) {
    #CurrentProperty = $StatsData.ReadHitIOs
    #CurrentPropertyName = "ReadHitIOs"

    // Avoid double-checking for inclusion of this metric
```

```

    $Manifest.IncludeReadHitIOs = false
  } else if ($Manifest.IncludeReadIOTimeCounter) {
    #CurrentProperty = $StatsData.ReadIOTimeCounter
    #CurrentPropertyName = "ReadIOTimeCounter"

    // Avoid double-checking for inclusion of this metric
    $Manifest.IncludeReadIOTimeCounter = false
  } else if ($Manifest.IncludeReadHitIOTimeCounter) {
    #CurrentProperty = $StatsData.ReadHitIOTimeCounter
    #CurrentPropertyName = "ReadHitIOTimeCounter"

    // Avoid double-checking for inclusion of this metric
    $Manifest.IncludeReadHitIOTimeCounter = false
  } else if ($Manifest.IncludeKBytesRead) {
    #CurrentProperty = $StatsData.KBytesRead
    #CurrentPropertyName = "KBytesRead"

    // Avoid double-checking for inclusion of this metric
    $Manifest.IncludeKBytesRead = false
  } else if ($Manifest.IncludeWriteIOs) {
    #CurrentProperty = $StatsData.WriteIOs
    #CurrentPropertyName = "WriteIOs"

    // Avoid double-checking for inclusion of this metric
    $Manifest.IncludeWriteIOs = false
  } else if ($Manifest.IncludeWriteHitIOs) {
    #CurrentProperty = $StatsData.WriteHitIOs
    #CurrentPropertyName = "WriteHitIOs"

    // Avoid double-checking for inclusion of this metric
    $Manifest.IncludeWriteHitIOs = false
  } else if ($Manifest.IncludeWriteIOTimeCounter) {
    #CurrentProperty = $StatsData.WriteIOTimeCounter
    #CurrentPropertyName = "WriteIOTimeCounter"

    // Avoid double-checking for inclusion of this metric
    $Manifest.IncludeWriteIOTimeCounter = false
  } else if ($Manifest.IncludeWriteHitIOTimeCounter) {
    #CurrentProperty = $StatsData.WriteHitIOTimeCounter
    #CurrentPropertyName = "WriteHitIOTimeCounter"

    // Avoid double-checking for inclusion of this metric
    $Manifest.IncludeWriteHitIOTimeCounter = false
  } else if ($Manifest.IncludeKBytesWritten) {
    #CurrentProperty = $StatsData.KBytesWritten
    #CurrentPropertyName = "KBytesWritten"

    // Avoid double-checking for inclusion of this metric
    $Manifest.IncludeKBytesWritten = false
  } else if ($Manifest.IncludeIdleTimeCounter) {
    #CurrentProperty = $StatsData.IdleTimeCounter

```

```
#CurrentPropertyName = "IdleTimeCounter"

// Avoid double-checking for inclusion of this metric
$Manifest.IncludeIdleTimeCounter = false
} else if ($Manifest.IncludeMaintOp) {
#CurrentProperty = $StatsData.MaintOp
#CurrentPropertyName = "MaintOp"

// Avoid double-checking for inclusion of this metric
$Manifest.IncludeMaintOp = false
} else if ($Manifest.IncludeMaintTimeCounter) {
#CurrentProperty = $StatsData.MaintTimeCounter
#CurrentPropertyName = "MaintTimeCounter"

// Avoid double-checking for inclusion of this metric
$Manifest.IncludeMaintTimeCounter = false
}

if (#CurrentPropertyName != "Unknown") {
#CurrentElement = Integer(#RecordElements[#k])
if (#StatisticTime == $BlockStats.StatisticTime) {
// record and instance property should be equal
if (#CurrentElement != #CurrentProperty) {
<ERROR! Record Element inconsistent with BSSD
property #CurrentPropertyName>
}
} else if (#StatisticTime > $BlockStats.StatisticTime) {
// record should be >= instance property
if (#CurrentElement < #CurrentProperty) {
<ERROR! Record Element inconsistent with BSSD property
#CurrentPropertyName. The counter may have
rolled back to 0>
}
} else {
// record should be <= instance property
if (#CurrentElement > #CurrentProperty) {
<ERROR! Record Element inconsistent with BSSD property
#CurrentPropertyName. The counter may have
rolled back to 0>
}
}
}
}
k++

} // while (#k < #RecordElements[].length)...
} // for (#j in #Records[])
} // for (#i in #BulkStatistics[])
}

// This function takes a container of BlockManifest instances and locates
the
```



```

// instance that represents the specified element type. Null is returned
// if
// the specified instance cannot be located in the container.
sub CIMInstance GetBlockManifestByType($BlockManifests[], #elementType)
{
    for (#i in $BlockManifests[]) {
        if ($BlockManifests[#i].ElementType == #elementType) {
            return $BlockManifests[#i]
        }
    }
    return null
}

```

```

// MAIN
//
// 1. Loop through the top-level ComputerSystems and retrieve the
// hosted BlockStatisticsService.
for (#i in $StorageSystems->[]) {

    // Step 1. Retrieve the hosted BlockStatisticsService.
    $StorageSystem-> = $StorageSystems->[#i]
    $StatServices->[] = AssociatorNames($StorageSystem->,
        "CIM_HostedService",
        "CIM_BlockStatisticsService",
        "Antecedent",
        "Dependent")
    // There should be one and only one BlockStatisticsService.
    $StatService-> = $StatServices->[0]

    // Step 2. Retrieve the capabilities describing the
    // BlockStatisticService.
    $StatCapabilities[] = Associators($StatService->,
        "CIM_ElementCapabilities",
        "CIM_BlockStatisticsCapabilities",
        "ManagedElement",
        "Capabilities",
        false,
        false,
        {"ElementTypesSupported", "SynchronousMethodsSupported"})
    // There should be one and only one BlockStatisticsCapabilities.
    $Capabilities = $StatCapabilities[0]
    #SynchCollectionRetrieval = contains(4, //
        "GetStatisticsCollection"
        $Capabilities.SynchronousMethodsSupported)

    // Step 3. Locate the StatisticsCollection
    $StatCollections->[] = AssociatorNames($StorageSystem->,
        "CIM_HostedCollection",
        "CIM_StatisticsCollection",
        "Antecedent",
        "Dependent")
}

```

```
// There should be one and only one StatisticsCollection.
$StatCollection-> = $StatCollections->[0]
// Step 4. Locate the default ManifestCollection
$ManifestCollections[] = Associators($StatCollection->,
    "CIM_AssociatedBlockStatisticsManifestCollection",
    "CIM_BlockStatisticsManifestCollection",
    "Statistics",
    "ManifestCollection",
    false,
    false,
    {"IsDefault"})
$DefaultManifestCollection = null
for (#j in $ManifestCollections[]) {
    if ($ManifestCollections[#j].IsDefault) {
        $DefaultManifestCollection = $ManifestCollections[#j]
        break
    }
}
if ($DefaultManifestCollection == null) {
    <ERROR! A default ManifestCollection MUST exist>
}
// Step 5. Locate the default BlockManifests which identify what
// statistical
// data is supported for each element type. (e.g. disk, volume, etc.)
#PropList = {"InstanceID", "ElementName", "ElementType",
    "IncludeStatisticTime", "IncludeTotalIOs",
    "IncludeKBytesTransferred", "IncludeIOTimeCounter",
    "IncludeReadIOs", "IncludeReadHitIOs",
    "IncludeReadIOTimeCounter",
    "IncludeReadHitIOTimeCounter", "IncludeKBytesRead",
    "IncludeWriteIOs", "IncludeWriteHitIOs",
    "IncludeWriteIOTimeCounter", "IncludeWriteHitIOTimeCounter",
    "IncludeKBytesWritten", "IncludeIdleTimeCounter",
    "IncludeMaintOp",
    "IncludeMaintTimeCounter"}
$DefaultBlockManifests[] = Associators(
    $DefaultManifestCollection.GetObjectPath(),
    "CIM_MemberOfCollection",
    "CIM_BlockStatisticsManifest",
    "Collection",
    "Member",
    false,
    false,
    #PropList)
// There MUST be one default Block Manifest for each element type
// supported.
if ($Capabilities.ElementTypesSupported[].length
    != $DefaultBlockManifest[].length) {
    <ERROR! Required default BlockManifests do not exist>
}

// Step 6. Traverse from the StatisticsCollection to the
```

```

// BlockStorageStatisticalData. If SyncCollectionRetrieval is
// supported,
// then this is necessary for validation of the Manifest data
// retrieved
// through the GetStatisticsCollection method. If it is not
// supported,
// then these instances must be used to retrieve the statistics
// directly.
$BlockStats[] = Associators($StatCollection->,
    "CIM_MemberOfCollection",
    "CIM_BlockStorageStatisticalData",
    "Collection",
    "Member",
    false,
    false,
    #PropList)

if (#SynchCollectionRetrieval) {

    // Step 7a. Retrieve the data specified by the default
    // ManifestCollection in bulk.
    %InArguments["ElementTypes"] =
        $Capabilities.ElementTypesSupported[]
    %InArguments["ManifestCollection"] =
        $DefaultManifestCollection.GetObjectPath()
    %InArguments["StatisticsFormat"] = 2// "CSV"
    #MethodReturn = InvokeMethod($StatService->,
        "GetStatisticsCollection",
        %InArguments,
        %OutArguments)
    if (#MethodReturn == 0) {
        #Statistics[] = %OutArguments["Statistics"]
        // Step 8. Parse the bulk statistical data retrieved to validate
        // the values (at least as much as is feasible)
        &ValidateRecords(#Statistics[], $DefaultBlockManifests[],
            $BlockStats[])
    } else {
        <ERROR! Bulk statistical data retrieval failed>
    }
} else {

    // Step 7b. Since bulk statistics retrieval is not supported, the
    // statistical data must be retrieved directly.
    for (#j in $BlockStats[]) {
        $BlockStat = $BlockStats[#j]
        $BlockManifest =
            GetBlockManifestByType($DefaultBlockManifests
                [],
                $BlockStat.ElementType)
        if ($BlockManifest == null) {
            <ERROR! The required default BlockManifest does not exist for
            this element type>

```

```
    }
    // Determine the supported statistical properties specified by
    // $BlockManifest, and retrieve the corresponding property
    // values
    // for this element type from $BlockStat.
  }
}
```

EXPERIMENTAL

8.7.2 Building an Object Map of Metered Elements

```
// DESCRIPTION
//
// This recipe describes how to build a record of all metered object
// instances
// and a topology of how the instances are related. (e.g. volume mapping
// to
// disk drives, ports used to access volumes, etc.)
//
// PRE-EXISTING CONDITIONS AND ASSUMPTIONS:
// 1. The name of a top-level ComputerSystem instance for an Array,
// Storage
// Virtualizer, or Volume Manager implementation supporting the Block
// Server
// Performance Subprofile has previously been discovered via SLP and is
// known
// as $StorageSystem->.
// 2. The element types that support performance statistics are known as
// #ElementTypes[] whose content is populated from the property value of
// CIM_BlockStatisticsCapabilities.ElementTypesSupported.
// 3. The performance statistics properties supported for each element
// type are
// known as #<ElementType>DataPropList[]. (e.g. #VolumeDataPropList[],
// #DiskDataPropList[], etc.) The content of the property lists is
// determine
// from the default instance of CIM_BlockStatisticsManifest for each
// element type.
// 4. The required properties for each element type are know as
// #<ElementType>PropList[]. (e.g. #VolumePropList[],
// #DiskDataPropList[], etc.)

// Function GetAssociatedStats
//
// This function retrieves the instance data of
// BlockStorageStatisticalData
// associated to the specified metered object. If there is no instance
// data
// associated, null is returned.
//
sub CIMInstance[] GetAssociatedStats(CIMObjectPath $MeteredObject->,
  string[] #PropList) {
```

```

$StatData[] = Associators($MeteredObject->,
    "CIM_ElementStatisticalData",
    "CIM_BlockStorageStatisticalData",
    "ManagedElement",
    "Stats",
    false,
    false,
    #PropList)
return $StatData[]
}

// This function retrieves the performance statistics of a
// CompositeExtent
// then recursively traverses the hierarchy beneath it.
sub void traverseComposition(REF $Composite->) {

    // Retrieve the performance statistics of the Composite Extent.
    $CompositeExtentStatData[] = &GetAssociatedStats($Composite->,
        #ExtentDataPropList[])
    // There may not be BlockStorageStatisticalData for each and every
    // level
    // of Composite Extents.
    if ($CompositeExtentStatData[] != null) {
        $CompositeExtentStats = $CompositeExtentStatData[0]
    }

    // Retrieve the associations in which this Composite Extent is the
    // Dependent reference. The association instances retrieved should be
    // either BasedOn or CompositeExtentBasedOn.
    $Associations[] = References($Composite->,
        "CIM_BasedOn",
        "Dependent",
        false,
        false,
        NULL)

    // There must be one or more associations involving the Composite
    // Extent
    // as the Antecedent reference.
    if ($Associations[] == null || $Associations[].length == 0) {
        <EXIT! Required associations not found>
    }

    // Determine which association class was discovered.
    #AssocClass = "CIM_BasedOn"
    if ($Associations[0] ISA CIM_CompositeExtentBasedOn) {
        #AssocClass = "CIM_CompositeExtentBasedOn"
    }

    // Retrieve the underlying Extents.
    $TargetExtents->[] = AssociatorNames($Composite->,
        #AssocClass,

```

```
        NULL,
        "Dependent",
        "Antecedent")

// Examine the QOS of the current level's Composite Extent
$CompositeExtent = GetInstance($Composite->,
    false,
    false,
    false,
    {"IsConcatenated", "ExtentStripeLength"})

// For each underlying extent at this level, traverse the sub-tree it
// is
// the sub-root of. If the extent is a CompositeExtent, then this is
// part
// of a complex RAID level; recursively invoke the Composition
// Algorithm.
// Otherwise it is just a regular StorageExtent and thus must be
// decomposed
// from it's Antecedent, so invoke the recursive Decomposition
// Algorithm.
for (#i in $TargetExtents->[]) {
    if ($TargetExtents->[#i] ISA CIM_CompositeExtent) {
        &traverseComposition($TargetExtents->[#i++])
    } else {
        &traverseDecomposition($TargetExtents->[#i++])
    }
}

// This function recursively traverses the hierarchy below a non-
// Composite
// StorageExtent.
sub void traverseDecomposition(REF $StartingExtent->) {
    // The Starting Extent is allocated partially or in full from the
    // Antecedent Extent, so a single BasedOn is expected.
    $TargetExtents[] = Associators($StartingExtent->,
        "CIM_BasedOn",
        "CIM_StorageExtent",
        "Dependent",
        "Antecedent",
        false,
        false,
        {"Primordial"})

    // Since the Starting Extent is allocated from the Antecedent, there
    // must
    // be only one Antecedent.
    if ($TargetExtents[] == null || $TargetExtents[].length != 1) {
        <ERROR! Extent allocated from multiple Antecedents>
    }
}
```

```

$TargetExtent = $TargetExtents[0]

if ($TargetExtent ISA CIM_CompositeExtent) {
    // This is a Composite Extent representing a RAID Level. Since we
    // encountered the Composite in a decomposition, it is the "top"
    // extent in a pool and the Dependent/Antecedent relationship falls
    // into one of the following scenarios:
    //
    // o The Starting Extent is a StorageVolume that is one-to-one with
    //   the Target Composite Extent.
    //
    // o The Starting Extent is a StorageVolume partially allocated from
    //   the Target Composite Extent, where the Composite is one-to-one
    //   with the Storage Pool which is a RAID Group.
    //
    // o The Starting Extent is a ComponentExtent of a Child Concrete
    //   pool and is partially allocated from the Target Composite
    //       Extent
    //   where the Composite is one-to-one with the parent RAID Group
    //       pool.
    //
    // Call the (recursive) function to analyze the sub-hierarchy
    // composed by the Target Extent.
    //
    &traverseComposition($TargetExtent.GetObjectPath())
} else {
    // Check here to see if we have reached the leaves of the hierarchy
    if ($TargetExtent.Primordial == true) {
        // Recursion ends with each Primordial Extent.
        return
    } else {
        // Since the Dependent was a regular StorageExtent, and not
        // Primordial, it must be decomposed from an Antecedent, so
        //   invoke
        // ourselves recursively.
        &traverseDecomposition($TargetExtent.GetObjectPath())
    }
}

// This function locates the logical devices on the specified
//   ComputerSystem
// and retrieves the supported statistical information. Note that the
// ComputerSystem specified may be a top-level, peer, front-end or back-
//   end
// system.
sub void discoverSupportedDeviceStats(REF $System->) {

    // Retrieve all ports on the system.
    $Ports[] = Associators($System.GetObjectPath(),
        "CIM_SystemDevice",
        "CIM_LogicalPort",

```

```
        "GroupComponent",
        "PartComponent",
        false,
        false,
        #PortPropList[])
if ($Ports[] != null && $Ports[].length > 0) {

    // Determine if performance statistics are supported for any type of
    // port.
    #SupportsPortStats = contains(6, #ElementTypes[]) // "Front-end
        Port"
        || contains(7, #ElementTypes[]) // "Back-end Port"
    for (#j in $Ports[]) {
        if (#SupportsPortStats) {

            // Retrieve the performance statistics of the system's port.
            $PortStatData[] = &GetAssociatedStats(
                $Ports[#j].getObjectPath(),
                #PortDataPropList[])
            // NOTE: Performance statistics may not be supported for
            // this particular type of port. (i.e. Front-end vs. Back-end)
            if ($PortStatData[] != null && $PortStatData[].length > 0) {
                // There should be one and only one
                // BlockStorageStatisticalData.
                $PortStats[#j] = $PortStatData[0]

                // Determine the type of this port.
                #PortType[#j] = $PortStats.ElementType
            }
        }
    }
}

// Retrieve all volumes on the system.
$Volumes[] = Associators($System.getObjectPath(),
    "CIM_SystemDevice",
    "CIM_StorageVolume",
    "GroupComponent",
    "PartComponent",
    false,
    false,
    #VolumePropList[])
if ($Volumes[] != null && $Volumes[].length > 0) {

    // Determine if performance statistics are supported for volume.
    #SupportsVolumeStats = contains(8, #ElementTypes[]) // "Volume"
    for (#k in $Volumes[]) {
        if (#SupportsVolumeStats) {

            // Retrieve the performance statistics of the volumes
            $VolumeStatData[] = &GetAssociatedStats(
```



```

        $Volumes[#k].getObjectPath(),
        #VolumeDataPropList[])
    // There should be one and only one BlockStorageStatisticalData.
    $VolumeStats = $VolumeStatData[0]
}

// Retrieve the protocol controllers through which the volume is
// visible.
$ProtocolControllers[] =
    Associators($Volumes[#k].getObjectPath(),
        "CIM_ProtocolControllerForUnit",
        "CIM_SCSIProtocolController",
        "Dependent",
        "Antecedent",
        false,
        false,
        #ProtocolControllerPropList[])
if ($ProtocolControllers[] != null
    && $ProtocolControllers[].length > 0) {
for (#l in ($ProtocolControllers[])) {

    // Retrieve the protocol controller's endpoint.
    $ProtocolEndpoints[] = Associators(
        $ProtocolControllers[#l].getObjectPath(),
        "CIM_SAPAvailableForElement",
        "CIM_SCSIProtocolEndpoint",
        "ManagedElement",
        "AvailableSAP",
        false,
        false,
        #ProtocolControllerPropList[])
    if ($ProtocolEndpoints[] != null ) {
        for (#pe in (#ProtocolEndpoints[])) {
            // There should be one and only one ProtocolEndpoint
            $ProtocolEndpoint = $ProtocolEndpoints[#pe]

            // Retrieve the ports that access this ProtocolEndpoint.
            $AccessingPorts[] = Associators(
                $ProtocolEndpoint.getObjectPath(),
                "CIM_DeviceSAPImplementation",
                "CIM_LogicalPort",
                "Dependent",
                "Antecedent",
                false,
                false,
                #PortPropList[])
        }
    }
}
}

```

```
// Determine if performance statistics are supported for
    Extents.
#SupportsExtentStats = contains(9, #ElementTypes[])// "Extent"

// NOTE: StorageExtents are investigated ONLY if performance
// statistics are supported for "Extent" and/or "Disk Drive".
// Performance statistics support for "composite" StorageExtents
// is indicated by the "Extent" capability. Performance
    statistics
// support for "primordial" StorageExtents is indicated by the
// "Disk Drive" capability.
//
// StorageExtents may not be present if the Extent Composition
// Subprofile is not supported.
if (#SupportsExtentStats) {

// Retrieve the StorageExtents that comprise the StorageVolume.
$ComponentExtents[] = Associators(
    $Volumes[#k].getObjectPath(),
    "CIM_BasedOn",
    "CIM_StorageExtent",
    "Dependent",
    "Antecedent",
    false,
    false,
    #ExtentPropList)

// Retrieve the performance statistics of the composite
// Storage Extent(s).
if ($ComponentExtents[] != null
    && $ComponentExtents[].length > 0) {
    &traverseComposition($ComponentExtents[0].getObjectPath())
}
}

// Determine if performance statistics are supported for Disk
    Drive.
#SupportsDiskStats = contains(10, #ElementTypes[])// "Disk
    Drive"
if (#SupportsDiskStats) {

// Retrieve the primordial StorageExtents to which the disk
// performance statistics will be associated.
$DiskExtents[] = &findPrimordials(
    $Volumes[#k].getObjectPath())
if ($DiskExtents[] == null || $DiskExtents[].length == 0) {
    <ERROR! Required primordial StorageExtents not found>
}
for (#m in $DiskExtents[]) {
    $DiskExtentStatData[] = &GetAssociatedStats(
        $DiskExtents[#m].getObjectPath(),
        #DiskDataPropList[])
```

```

        // There should be one and only one
        // BlockStorageStatisticalData.
        $DiskExtentStats = $DiskExtentStatData[0]
    }
}
}
}

// MAIN
//
// Step 1. Retrieve the performance statistics for the top-level system.
if (contains(2, // "Computer System"
    #ElementTypes[]) {
    $TopSystemStatData[] = &GetAssociatedStats($StorageSystem->,
        #TopSystemDataPropList[])
    // There should be one and only one BlockStorageStatisticalData.
    $TopSystemStats = $TopSystemStatData[0]
}

// Step 2. Discover the logical devices on the top-level system and their
// related performance statistics
&discoverSupportedDeviceStats($StorageSystem->)

// Step 3. Retrieve the component systems in a multiple system device.
// NOTE: Traversing ComponentCS from the top-level system to its
// component
// systems will retrieve ALL component systems. In the case of a device
// that
// supports 2-tier redundancy, the relationship between the component
// systems
// (i.e. first redundancy tier) to the sub-component systems would be
// determined
// by investigating the ConcreteIdentity and MemberOfCollection
// relationships
// to a RedundancySet. See the Multiple Computer System Subprofile for
// more
// detail.
$ComponentSystems[] = Associators($StorageSystem->,
    "CIM_ComponentCS",
    "CIM_ComputerSystem",
    "GroupComponent",
    "PartComponent",
    false,
    false,
    #ComponentSystemPropList[])
if ($ComponentSystems[] != null && $ComponentSystems[].length > 0) {

    // Step 4. Determine if performance statistics are supported for any
    // type
    // of component system.
    #SupportsComponentSystemStats =

```

```

contains(3, #ElementTypes[])// "Front-end Computer System"
|| contains(4, #ElementTypes[])// "Peer Computer System"
|| contains(5, #ElementTypes[])// "Back-end Computer System"
for (#i in $ComponentSystems[]) {
    $ComponentSystemPath = $ComponentSystems[#i].getObjectPath()
    if (#SupportsComponentSystemStats) {

        // Step 5. Retrieve the performance statistics of the component
        // system.
        $ComponentSystemStatData[] = &GetAssociatedStats(
            $ComponentSystemPath,
            #ComponentSystemDataPropList[])
        // NOTE: Performance statistics may not be supported for this
        // particular type of component system. (i.e. Front-end vs.
        // Back-end vs. Peer Computer Systems)
        if ($ComponentSystemStatData[] != null
            && $ComponentSystemStatData[].length > 0) {
            // There should be one and only one BlockStorageStatisticalData.
            $ComponentSystemStats[#i] = $ComponentSystemStatData[0]

            // Step 6. Determine the type of this component system.
            #ComponentSystemType[#i] = $ComponentSystemStats.ElementType
        }

        // Step 7. Discover the Topology of the component computer
        // systems by
        // finding the RedundancySet that each of the ComponentSystems
        // belong
        // to (if any), and the ComputerSystem that has a concrete
        // identity
        // relationship with that RedundancySet. The computer system
        // that is
        // one tier above the current component system is stored in an
        // array
        // of ParentComputerSystems, with each entry corresponding to
        // the
        // component system at the same index in the ComponentSystems
        // array.

        $RedundancySets->[] = AssociatorNames($ComponentSystemPath->,
            "CIM_MemberOfCollection",
            "CIM_RedundancySet",
            "Member",
            "Collection")
        if(RedundancySets->[] != null && $RedundancySets->[].length > 0)
        {
            if($RedundancySets->[].length > 1)
            {
                <ERROR! Component System belongs to more than one
                Redundancy
                Set>
            }
        }
    }
}

```

```

$AggregateSystems->[] = AssociatorNames($RedundancySets-
    >[0],
    "CIM_ConcreteIdentity",
    "CIM_ComputerSystem",
    "SameElement",
    "SystemElement")
if($AggregateSystems->[] == null ||
    $AggregateSystems->[].length != 1)
{
    <ERROR! Could not find Concrete Computer System for
        Redundancy
        Set>
}
$ParentComputerSystems->[#i] = $AggregateSystems->[0]
}
}
// Step 8. Discover the logical devices on the component system and
// their related performance statistics
&discoverSupportedDeviceStats($ComponentSystemPath->)
}
}

```

EXPERIMENTAL

8.7.3 Retrieving Statistics for a Specific Volume

```

// DESCRIPTION
//
// This recipe describes how to retrieve the supported performance
// statistics
// for a specific set of StorageVolumes.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTIONS:
// 1. The name of a top-level ComputerSystem instance for an Array,
//    Storage
//    Virtualizer, or Volume Manager implementation supporting the Block
//    Server
// Performance Subprofile has previously been discovered via SLP and is
//    known
// as $StorageSystem->.
// 2. A specific set of StorageVolumes is known as $StorageVolume->[].
//
// MAIN
//
// Step 1. Retrieve the hosted BlockStatisticsService.
$StatServices->[] = AssociatorNames($StorageSystem->,
    "CIM_HostedService",
    "CIM_BlockStatisticsService",
    "Antecedent",
    "Dependent")

```

```
// There should be one and only one BlockStatisticsService.
$StatService-> = $StatServices->[0]

// Step 2. Retrieve the capabilities describing the
// BlockStatisticService.
$StatCapabilities[] = Associators($StatService->,
    "CIM_ElementCapabilities",
    "CIM_BlockStatisticsCapabilities",
    "ManagedElement",
    "Capabilities",
    false,
    false,
    {"ElementTypesSupported"})
// There should be one and only one BlockStatisticsCapabilities.
$Capabilities = $StatCapabilities[0]
if !contains(8, // "Volume"
    $Capabilities.ElementTypesSupported) {

    <EXIT! StorageVolume performance statistics not supported>
}

// Step 3. Locate the default ManifestCollection
$ManifestCollections[] = Associators($StatCollection->,
    "CIM_AssociatedBlockStatisticsManifestCollection",
    "CIM_BlockStatisticsManifestCollection",
    "Statistics",
    "ManifestCollection",
    false,
    false,
    {"IsDefault"})
$DefaultManifestCollection = null
for #i in $ManifestCollections[] {
    if ($ManifestCollections[#i].IsDefault) {
        $DefaultManifestCollection = $ManifestCollections[#i]
        break
    }
}
if ($DefaultManifestCollection == null) {
    <ERROR! A default ManifestCollection MUST exist>
}

// Step 4. Locate the default BlockManifest which identifies the
// statistical
// data supported for StorageVolumes.
$VolumeManifest = null
string[] #PropList = {"ElementType", "IncludeStatisticTime",
    "IncludeTotalIOs",
    "IncludeKBytesTransferred", "IncludeIOTimeCounter",
    "IncludeReadIOs",
    "IncludeReadHitIOs", "IncludeReadIOTimeCounter",
    "IncludeReadHitIOTimeCounter", "IncludeKBytesRead",
    "IncludeWriteIOs",
```

```

        "IncludeWriteHitIOs", "IncludeWriteIOTimeCounter",
        "IncludeWriteHitIOTimeCounter", "IncludeKBytesWritten",
        "IncludeIdleTimeCounter", "IncludeMaintOp",
        "IncludeMaintTimeCounter"}
$DefaultBlockManifests[] = Associators(
    $DefaultManifestCollection.GetObjectPath(),
    "CIM_MemberOfCollection",
    "CIM_BlockStatisticsManifest",
    "Collection",
    "Member",
    false,
    false,
    #PropList)
for #i in $DefaultBlockManifests[] {
    if ($DefaultBlockManifests[#i].ElementType == 8) {
        $VolumeManifest = $DefaultBlockManifests[#i]
        break
    }
}
if ($VolumeManifest == null) {
    <ERROR! Required default BlockManifest for StorageVolume not found>
}

// Step 5. Retrieve the performance statistics for each specified
// StorageVolume.
for (#i in $StorageVolume->[]) {
    $VolumeStatData[] = Associators($StorageVolume->[#i],
        "CIM_ElementStatisticalData",
        "CIM_BlockStorageStatisticalData",
        "ManagedElement",
        "Stats",
        false,
        false,
        null)
    // There should be one and only one BlockStorageStatisticalData.
    if ($VolumeStatData[] == null || $VolumeStatData[].length != 1) {
        <ERROR! The required staticistics were not found>
    }
    $VolumeStats = $VolumeStatData[0]

    // Step 6. Extract the performance statistics supported by the
    // StorageVolume.
    if ($VolumeManifest.IncludeStatisticTime) {
        #StatisticTime = VolumeStats.StatisticTime
    } else {
        <ERROR! StatisticTime is a required property for Volumes and should
        be set to "true" in the default BlockManifest>
    }
    if ($VolumeManifest.IncludeTotalIOs) {
        #TotalIOs = VolumeStats.TotalIOs
    } else {

```

```
<ERROR! TotalIOs is a required property for Volumes and should
    be set to "true" in the default BlockManifest>
}
if ($VolumeManifest.IncludeKBytesTransferred) {
    #KBytesTransferred = VolumeStats.KBytesTransferred
} else {
    <ERROR! KBytesTransferred is a required property for Volumes and
        should be set to "true" in the default BlockManifest>
}
if ($VolumeManifest.IncludeIOTimeCounter) {
    #IOTimeCounter = VolumeStats.IOTimeCounter
}
if ($VolumeManifest.IncludeReadIOs) {
    #ReadIOs = VolumeStats.ReadIOs
} else {
    <ERROR! ReadIOs is a required property for Volumes and should
        be set to "true" in the default BlockManifest>
}
if ($VolumeManifest.IncludeReadHitIOs) {
    #ReadHitIOs = VolumeStats.ReadHitIOs
} else {
    <ERROR! ReadHitIOs is a required property for Volumes and should
        be set to "true" in the default BlockManifest>
}
if ($VolumeManifest.IncludeReadIOTimeCounter) {
    #ReadIOTimeCounter = VolumeStats.ReadIOTimeCounter
}
if ($VolumeManifest.IncludeReadHitIOTimeCounter) {
    #ReadHitIOTimeCounter = VolumeStats.ReadHitIOTimeCounter
}
if ($VolumeManifest.IncludeKBytesRead) {
    #KBytesRead = VolumeStats.KBytesRead
}
if ($VolumeManifest.IncludeWriteIOs) {
    #WriteIOs = VolumeStats.WriteIOs
} else {
    <ERROR! WriteIOs is a required property for Volumes and should
        be set to "true" in the default BlockManifest>
}
if ($VolumeManifest.IncludeWriteHitIOs) {
    #WriteHitIOs = VolumeStats.WriteHitIOs
} else {
    <ERROR! WriteHitIOs is a required property for Volumes and should
        be set to "true" in the default BlockManifest>
}
if ($VolumeManifest.IncludeWriteIOTimeCounter) {
    #WriteIOTimeCounter = VolumeStats.WriteIOTimeCounter
}
if ($VolumeManifest.IncludeWriteHitIOTimeCounter) {
    #WriteHitIOTimeCounter = VolumeStats.WriteHitIOTimeCounter
}
}
```



```

    if ($VolumeManifest.IncludeKBytesWritten) {
        #KBytesWritten = VolumeStats.KBytesWritten
    }
    if ($VolumeManifest.IncludeIdleTimeCounter) {
        #IdleTimeCounter = VolumeStats.IdleTimeCounter
    }
}

```

8.7.4 Summary of Statistics Support by Element

Not all statistics properties are kept for all elements. Table 117 illustrates the statistics properties that are kept for each of the metered elements.

Table 117 - Summary of Statistics Support by Element

Statistic Property	Top Level Computer System	Component Computer System (Front-end)	Component Computer System (Peer)	Component Computer System (Back-end)	Front-end Port	Back-end Port	Volume (LogicalDisk)	Composite Extent	Disk
StatisticTime	R	R	R	R	R	R	R	R	R
TotalIOs	R	R	R	R	R	R	R	R	R
KBytes Transferred	R	O	O	O	R	O	R	R	R
IOTimeCounter	O	O	O	O	O	O	O	N	O
ReadIOs	O	R	R	N	N	N	R	N	R
ReadHitIOs	O	R	R	N	N	N	R	N	N
ReadIOTimeCounter	O	O	O	N	N	N	O	N	O
ReadHitIO TimeCounter	O	O	O	N	N	N	O	N	N
KBytesRead	O	O	O	O	N	N	O	N	O
WriteIOs	O	R	R	N	N	N	R	N	O
WriteHitIOs	O	R	R	N	N	N	R	N	N
WriteIOTimeCounter	O	O	O	N	N	N	O	N	O
WriteHitIO TimeCounter	O	O	O	N	N	N	O	N	N
KBytesWritten	O	O	O	O	N	N	O	N	O
IdleTimeCounter	N	N	N	O	O	N	O	O	O
MaintOp	N	N	N	N	N	N	N	O	O
MaintTime-Counter	N	N	N	N	N	N	N	O	O

The legend is:

R – Required

O – Optional

N – Not specified

Notice that there is a difference between the “front-end” port and “back-end” port elements. There is a difference between the top level computer system (i.e., the Array, Storage Virtualizer or Volume Management Profile) and the component computer systems. Furthermore, there can be variations in the component computer systems. This is based on how component computer systems are configured. In some cases, these computer systems are “front-end” and “back-end” controllers. In other subsystems, they are “peer” controllers.

NOTE Controller LUNs (SCSI Arbitrary Logical Units) and Remote Replica Group are not shown in Table 117: Summary of Block Statistics Support by Element. They only require StatisticTime, TotalIOs and KBytesTransferred. All other properties are not SPECIFIED.

A complete list of definitions of the metered elements as defined by the ElementType property of BlockStorageStatisticalData follows:

- ElementType = 2 (Computer System) - These are statistics for the whole Array (virtualizer or volume manager).
- ElementType = 3 (Front-end Computer System) - This is the Computer System (controller) that provides the support for receiving the IO from host systems. The Front-end function acts as a target of IO.
- ElementType = 4 (Peer Computer System) - This is a Computer System that acts as both a front-end and back-end Computer System.
- ElementType = 5 (Back-end Computer System) - This is the Computer System (controller) that provides the support for driving the IO to the back-end storage (disk drives or external volumes). The back-end function acts as an initiator of IO.
- ElementType = 6 (Front-end Port) - A port in a disk array that connects the disk array (or Storage Virtualizer) to hosts using the storage. The Front End port is usually connected to either the Peer Computer System (controller) or to the Front-end Computer System (controller) in some Disk Arrays or Storage Virtualizers.
- ElementType = 7 (Back-end Port) - A port that can be inside the disk array housing that connects to the disk drives. This is connected to either the Peer Computer system (controller) or to the Back-end Computer System (controller) in some Disk Arrays or Storage Virtualizers.
- ElementType = 8 (Volume) - This is a Logical Unit that is the target of data IOs for storing or retrieving data. This would be a StorageVolume for Arrays or Storage Virtualizers. It would be a LogicalDisk for Volume Management Profiles.
- ElementType = 9 (Extent) - This is an intermediate Storage Extent. That is, it is not a Volume and it is not a Disk Drive. An example of the use of an Extent would be a RAID rank that creates a logical storage extent from multiple disk drives. In the case of Storage Virtualizers, this is used to represent the volumes that are imported from Arrays.
- ElementType = 10 (Disk Drive) - This is a disk drive.
- ElementType = 11 (Arbitrary LUs) - This is a Logical Unit that is the target of “control” IO functions. The Logical Unit does not contain data, but supports invocation of control functions in an Array or Storage Virtualizer.
- ElementType = 12 (Remote Replica Group) - Replication requires a local disk array and a remote disk array (in some “safe” location). The remote replica group is a group of disk drives in the remote disk array used to replicated defined data from the local disk array.

8.7.5 Formulas and Calculations

Table 117 identifies the set of statistics that are recommended for the various storage components in the array. These metrics, once collected, can be further enhanced through the definition of formulas and calculations that create additional ‘derived’ statistics.

Table 118 defines a set of such derived statistics. They are by no means the only possible derivations but serve as examples of the most commonly asked for statistics.

Table 118 - Formulas and Calculations

Calculated Statistics	
New statistic	Formula
TimeInterval	delta StatisticTime
% utilization	$100 * (\text{delta StatisticTime} - \text{delta IdleTime}) / \text{delta StatisticTime}$
I/O rate	$\text{delta TotalIOs} / \text{delta StatisticTime}$
I/O response time	$\text{delta IOTime} / \text{delta TotalIOs}$
Queue depth	$\text{delta I/O rate} * \text{delta I/O response time}$
Service Time	$\text{utilization} / \text{I/O rate}$
Wait Time	$\text{Response Time} - \text{Service Time}$
Average Read Size	$\text{delta KBytesRead} / \text{delta ReadIOs}$
Average Write Size	$\text{delta KBytesWritten} / \text{delta WriteIOs}$
% Read	$100 * (\text{delta ReadIOs} / \text{delta TotalIOs})$
% Write	$100 * (\text{delta WriteIOs} / \text{delta TotalIOs})$
% Hit	$100 * ((\text{delta ReadHitIOs} + \text{delta WriteHitIOs}) / \text{delta TotalIOs})$

8.7.6 Block Server Performance Supported Capabilities Patterns

The Capabilities patterns summarized in Table 119 are formally recognized by the Block Server Performance Subprofile of the current version of SMI-S.

Table 119 - Block Server Performance Subprofile Supported Capabilities Patterns

ElementSupported	SynchronousMethods Supported	AsynchronousMethods Supported
Any (at least one)	NULL	NULL
Any (at least one)	Neither GetStatisticsCollection nor Exec Query	NULL
Any (at least one)	GetStatisticsCollection	NULL
Any (at least one)	Any	NULL
Any (at least one)	Exec Query	NULL
Any (at least one)	GetStatisticsCollection, Query	NULL
Any (at least one)	Exec Query	NULL
Any (at least one)	"Manifest Creation", "Manifest Modification", and "Manifest Removal"	NULL
Any (at least one)	"Indications", "Query Collection"	NULL

An implementation will support GetStatisticsCollection, Query, GetStatisticsCollection and Query or neither. But if the implementation supports GetStatisticsCollection, it will shall support Synchronous execution.

If manifest collections are supported, then ALL three methods shall be supported (Creation, modification and removal).

8.7.7 Correlation of Block Storage Statistics and Fabric Statistics

A client will see statistics for Block Storage which describe statistical information relative to block access. This subprofile defines those statistics. But a client may also see statistics relative to networking activity (e.g., Port statistics). This subclause describes which metrics can be correlated between block storage statistics and port statistics.

8.8 CIM Elements

8.8.1 General

Table 120 describes the CIM elements for Block Server Performance.

Table 120 - CIM Elements for Block Server Performance

Element Name	Requirement	Description
8.8.2 CIM_AssociatedBlockStatisticsManifestCollection (Client defined collection)	Conditional	Conditional requirement: Clients can create manifests as identified by CIM_BlockStatisticsCapabilities.SynchronousMethodsSupported. This is an association between the StatisticsCollection and a client defined manifest collection.
8.8.3 CIM_AssociatedBlockStatisticsManifestCollection (Provider defined collection)	Mandatory	This is an association between the StatisticsCollection and a provider supplied (pre-defined) manifest collection that defines the statistics properties supported by the profile implementation.
8.8.4 CIM_BlockStatisticsCapabilities	Mandatory	This defines the statistics capabilities supported by the implementation of the profile.
8.8.5 CIM_BlockStatisticsManifest (Client Defined)	Conditional	Conditional requirement: Clients can modify manifests as identified by CIM_BlockStatisticsCapabilities.SynchronousMethodsSupported. An instance of this class defines the statistics properties of interest to the client for one element type.
8.8.6 CIM_BlockStatisticsManifest (Provider Support)	Mandatory	An instance of this class defines the statistics properties supported by the profile implementation for one element type.
8.8.7 CIM_BlockStatisticsManifestCollection (Client Defined)	Conditional	Conditional requirement: Clients can create manifests as identified by CIM_BlockStatisticsCapabilities.SynchronousMethodsSupported. An instance of this class defines one client defined collection of block statistics manifests (one manifest for each element type).
8.8.8 CIM_BlockStatisticsManifestCollection (Provider Defined)	Mandatory	An instance of this class defines the predefined collection of default block statistics manifests (one manifest for each element type).
8.8.9 CIM_BlockStatisticsService	Mandatory	This is a Service that provides (optional) services of bulk statistics retrieval and manifest set manipulation methods.
8.8.10 CIM_BlockStorageStatisticalData	Mandatory	This is a Subclass of CIM_StatisticalData for Block servers. It would be instantiated as specific block statistics for particular components.
8.8.11 CIM_ElementCapabilities	Mandatory	This associates the BlockStatisticsCapabilities to the BlockStatisticsService.

Table 120 - CIM Elements for Block Server Performance

Element Name	Requirement	Description
8.8.12 CIM_ElementStatisticalData (Back end Port Stats)	Conditional	Conditional requirement: Back end port statistics support. This is mandatory if CIM_BlockStatisticsCapabilities.ElementTypesSupported = "7". This associates a BlockStorageStatisticalData instance to the back end port for which the statistics are collected.
8.8.13 CIM_ElementStatisticalData (Component System Stats)	Conditional	Conditional requirement: Component Systems statistics support. This is mandatory if CIM_BlockStatisticsCapabilities.ElementTypesSupported = "3", "4" or "5". This associates a BlockStorageStatisticalData instance to the component ComputerSystem for which the statistics are collected.
8.8.14 CIM_ElementStatisticalData (Disk Stats)	Conditional	Conditional requirement: Disk Drive statistics support. This is mandatory if CIM_BlockStatisticsCapabilities.ElementTypesSupported = "10". This associates a BlockStorageStatisticalData instance to the StorageExtent (Disk Drive) for which the statistics are collected.
8.8.15 CIM_ElementStatisticalData (Extent Stats)	Conditional	Conditional requirement: Extent statistics support. This is mandatory if CIM_BlockStatisticsCapabilities.ElementTypesSupported = "9". This associates a BlockStorageStatisticalData instance to the StorageExtent (composite extent) for which the statistics are collected.
8.8.16 CIM_ElementStatisticalData (Front end Port Stats)	Conditional	Conditional requirement: Front-end port statistics support. This is mandatory if CIM_BlockStatisticsCapabilities.ElementTypesSupported = "6". This associates a BlockStorageStatisticalData instance to the target port for which the statistics are collected.
8.8.17 CIM_ElementStatisticalData (Logical Disk Stats)	Conditional	Conditional requirement: Volume statistics support in Volume Management Profiles. This is mandatory if CIM_BlockStatisticsCapabilities.ElementTypesSupported = "8", and the parent profile supports Logical Disks. This associates a BlockStorageStatisticalData instance to the volume for which the statistics are collected.

Table 120 - CIM Elements for Block Server Performance

Element Name	Requirement	Description
8.8.18 CIM_ElementStatisticalData (Remote Copy Stats)	Conditional	Conditional requirement: Remote Copy statistics support. This is mandatory if CIM_BlockStatisticsCapabilities.ElementTypesSupported = "12". This associates a BlockStorageStatisticalData instance to the remote copy service network for which the statistics are collected.
8.8.19 CIM_ElementStatisticalData (Top Level System Stats)	Conditional	Conditional requirement: Top level system statistics support. This is mandatory if CIM_BlockStatisticsCapabilities.ElementTypesSupported = "2". This associates a BlockStorageStatisticalData instance to the Top Level ComputerSystem for which the statistics are collected.
8.8.20 CIM_ElementStatisticalData (Volume Stats)	Conditional	Conditional requirement: Volume statistics support or Referenced from Array - StorageVolume is mandatory or Referenced from Storage Virtualizer - StorageVolume is mandatory. This is mandatory if CIM_BlockStatisticsCapabilities.ElementTypesSupported = "8", and the parent profile supports Storage Volumes. This associates a BlockStorageStatisticalData instance to the volume for which the statistics are collected.
8.8.21 CIM_HostedCollection (Client Defined)	Conditional	Conditional requirement: Clients can create manifests as identified by CIM_BlockStatisticsCapabilities.SynchronousMethodsSupported or Clients can create manifests as identified by CIM_BlockStatisticsCapabilities.AsynchronousMethodsSupported. This would associate a client defined BlockStatisticsManifestCollection to the top level system for the profile (e.g., array).
8.8.22 CIM_HostedCollection (Default)	Mandatory	This would associate a default BlockStatisticsManifestCollection to the top level system for the profile (e.g., array).
8.8.23 CIM_HostedCollection (Provider Supplied)	Mandatory	This would associate the StatisticsCollection to the top level system for the profile (e.g., array).
8.8.24 CIM_HostedService	Mandatory	This associates the BlockStatisticsService to the ComputerSystem that hosts it.
8.8.25 CIM_MemberOfCollection (Member of client defined collection)	Conditional	Conditional requirement: Clients can modify manifests as identified by CIM_BlockStatisticsCapabilities.SynchronousMethodsSupported. This would associate Manifests to client defined manifest collections.
8.8.26 CIM_MemberOfCollection (Member of pre-defined collection)	Mandatory	This would associate pre-defined Manifests to default manifest collection.
8.8.27 CIM_MemberOfCollection (Member of statistics collection)	Mandatory	This would associate all block statistics instances to the StatisticsCollection.

Table 120 - CIM Elements for Block Server Performance

Element Name	Requirement	Description
8.8.28 CIM_StatisticsCollection	Mandatory	This would be a collection point for all Statistics that are kept for a Block Server.
8.8.29 SNIA_BlockStatisticsCapabilities	Optional	Experimental. This is a subclass of CIM_BlockStatisticsCapabilities that adds the SupportedFeatures property.
8.8.30 SNIA_BlockStatisticsManifest (Client Defined)	Optional	Experimental. This is a subclass of CIM_BlockStatisticsManifest that adds the CSVSequence property.
8.8.31 SNIA_BlockStatisticsManifest (Provider Support)	Optional	Experimental. This is a subclass of CIM_BlockStatisticsManifest that adds the CSVSequence property.

8.8.2 CIM_AssociatedBlockStatisticsManifestCollection (Client defined collection)

The CIM_AssociatedBlockStatisticsManifestCollection associates an instance of a CIM_BlockStatisticsManifestCollection to the instance of CIM_StatisticsCollection to which it applies. Client defined manifest collections identify the Manifests (properties) for retrieval of block statistics.

CIM_AssociatedBlockStatisticsManifestCollection is not subclassed from anything.

There will be one instance of the CIM_AssociatedBlockStatisticsManifestCollection class, for each client defined manifest collection that has been created.

Created By: Extrinsic: CreateManifestCollection

Modified By: Static

Deleted By: Static

Requirement: Clients can create manifests as identified by CIM_BlockStatisticsCapabilities.SynchronousMethodsSupported.

Table 121 describes class CIM_AssociatedBlockStatisticsManifestCollection (Client defined collection).

Table 121 - SMI Referenced Properties/Methods for CIM_AssociatedBlockStatisticsManifestCollection (Client defined collection)

Properties	Requirement	Description and Notes
Statistics	Mandatory	The StatisticsCollection to which the manifest collection applies.
ManifestCollection	Mandatory	A client defined manifest collection.

8.8.3 CIM_AssociatedBlockStatisticsManifestCollection (Provider defined collection)

The CIM_AssociatedBlockStatisticsManifestCollection associates an instance of a CIM_BlockStatisticsManifestCollection to the instance of CIM_StatisticsCollection to which it applies. The default manifest collection defines the CIM_BlockStorageStatisticalData properties that are supported by the profile implementation.

CIM_AssociatedBlockStatisticsManifestCollection is not subclassed from anything.

One instance of the CIM_AssociatedBlockStatisticsManifestCollection shall exist for the default manifest collection if the Block Server Performance Subprofile is implemented.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 122 describes class CIM_AssociatedBlockStatisticsManifestCollection (Provider defined collection).

Table 122 - SMI Referenced Properties/Methods for CIM_AssociatedBlockStatisticsManifestCollection (Provider defined collection)

Properties	Requirement	Description and Notes
Statistics	Mandatory	The StatisticsCollection to which the manifest collection applies.
ManifestCollection	Mandatory	The default manifest collection.

8.8.4 CIM_BlockStatisticsCapabilities

An instance of the CIM_BlockStatisticsCapabilities class defines the specific support provided with the block statistics implementation. Note that there would be zero or one instance of this class in a profile. There would be none if the profile did not support the Block Server Performance Subprofile. There would be exactly one instance if the profile did support the Block Server Performance Subprofile.

CIM_BlockStatisticsCapabilities class is subclassed from CIM_Capabilities.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 123 describes class CIM_BlockStatisticsCapabilities.

Table 123 - SMI Referenced Properties/Methods for CIM_BlockStatisticsCapabilities

Properties	Requirement	Description and Notes
InstanceID	Mandatory	
ElementName	Mandatory	
ElementTypesSupported	Mandatory	ValueMap { "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12" }, Values {"Computer System", "Front-end Computer System", "Peer Computer System", "Back-end Computer System", "Front-end Port", "Back-endPort", "Volume", "Extent", "Disk Drive", "Arbitrary LUs", "Remote Replica Group"}.
SynchronousMethodsSupported	Mandatory	This property is mandatory, but the array may be empty. ValueMap { "2", "3", "4", "5", "6", "7", "8"}, Values {"Exec Query", "Indications", "QueryCollection", "GetStatisticsCollection", "Manifest Creation", "Manifest Modification", "Manifest Removal" }.
AsynchronousMethodsSupported	Optional	Not supported in current version of SMI-S.
ClockTickInterval	Mandatory	An internal clocking interval for all timers in the subsystem, measured in microseconds (Unit of measure in the timers, measured in microseconds). Time counters are monotonically increasing counters that contain "ticks". Each tick represents one ClockTickInterval. If ClockTickInterval contained a value of 32 then each time counter tick would represent 32 µs.

Table 123 - SMI Referenced Properties/Methods for CIM_BlockStatisticsCapabilities

Properties	Requirement	Description and Notes
Caption	Optional	Not Specified in this version of the Profile.
Description	Optional	Not Specified in this version of the Profile.
CreateGoalSettings()	Optional	Not Specified in this version of the Profile.

8.8.5 CIM_BlockStatisticsManifest (Client Defined)

The CIM_BlockStatisticsManifest class is Concrete class that defines the CIM_BlockStorageStatisticalData properties that should be returned on a GetStatisticsCollection request.

CIM_BlockStatisticsManifest is subclassed from CIM_ManagedElement.

In order for a client defined instance of the CIM_BlockStatisticsManifest class to exist, the all the manifest collection manipulation functions shall be identified in the "SynchronousMethodsSupported" property of the CIM_BlockStatisticsCapabilities (BlockStatisticsCapabilities.SynchronousMethodsSupported = "6") instance, AND a client shall have created at least ONE instance of CIM_BlockStatisticsManifestCollection.

Created By: Extrinsic: AddOrModifyManifest

Modified By: Extrinsic: AddOrModifyManifest

Deleted By: Extrinsic: RemoveManifest

Requirement: Clients can modify manifests as identified by CIM_BlockStatisticsCapabilities.SynchronousMethodsSupported.

Table 124 describes class CIM_BlockStatisticsManifest (Client Defined).

Table 124 - SMI Referenced Properties/Methods for CIM_BlockStatisticsManifest (Client Defined)

Properties	Requirement	Description and Notes
ElementName	Mandatory	A Client defined string that identifies the manifest.
InstanceID	Mandatory	The instance Identification. Within the scope of the instantiating Namespace, InstanceID opaquely and uniquely identifies an instance of this class.
ElementType	Mandatory	This value is required AND the current version of SMI-S specifies the following values: ValueMap {"2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12"} Values { "Computer System", "Front-end Computer System", "Peer Computer System", "Back-endComputer System", "Front-end Port", "Back-end Port", "Volume", "Extent", "Disk Drive", "Arbitrary LUs", "Remote Replica Group"}.
IncludeStatisticTime	Mandatory	
IncludeTotalIOs	Mandatory	
IncludeKBytesTransferred	Mandatory	
IncludeIOTimeCounter	Mandatory	
IncludeReadIOs	Mandatory	
IncludeReadHitIOs	Mandatory	
IncludeReadIOTimeCounter	Mandatory	
IncludeReadHitIOTimeCounter	Mandatory	

Table 124 - SMI Referenced Properties/Methods for CIM_BlockStatisticsManifest (Client Defined)

Properties	Requirement	Description and Notes
IncludeKBytesRead	Mandatory	
IncludeWriteIOs	Mandatory	
IncludeWriteHitIOs	Mandatory	
IncludeWriteIOTimeCounter	Mandatory	
IncludeWriteHitIOTimeCounter	Mandatory	
IncludeKBytesWritten	Mandatory	
IncludeIdleTimeCounter	Mandatory	
IncludeMaintOp	Mandatory	
IncludeMaintTimeCounter	Mandatory	
Caption	Optional	Not Specified in this version of the Profile.
Description	Optional	Not Specified in this version of the Profile.
IncludeStartStatisticTime	Optional	Not Specified in this version of the Profile.

8.8.6 CIM_BlockStatisticsManifest (Provider Support)

The CIM_BlockStatisticsManifest class is Concrete class that defines the CIM_BlockStorageStatisticalData properties that supported by the Provider. These Manifests are established by the Provider for the default manifest collection.

CIM_BlockStatisticsManifest is subclassed from CIM_ManagedElement.

At least one Provider supplied instance of the CIM_BlockStatisticsManifest class shall exist, if the Block Server Performance Subprofile is supported.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 125 describes class CIM_BlockStatisticsManifest (Provider Support).

Table 125 - SMI Referenced Properties/Methods for CIM_BlockStatisticsManifest (Provider Support)

Properties	Requirement	Description and Notes
ElementName	Mandatory	A Provider defined string that identifies the manifest in the context of the Default Manifest Collection.
InstanceID	Mandatory	The instance Identification. Within the scope of the instantiating Namespace, InstanceID opaquely and uniquely identifies an instance of this class.
ElementType	Mandatory	This value is required AND the current version of SMI-S specifies the following values: ValueMap {"2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12"} Values { "Computer System", "Front-end Computer System", "Peer Computer System", "Back-endComputer System", "Front-end Port", "Back-end Port", "Volume", "Extent", "Disk Drive", "Arbitrary LUs", "Remote Replica Group"}.
IncludeStatisticTime	Mandatory	

Table 125 - SMI Referenced Properties/Methods for CIM_BlockStatisticsManifest (Provider Support)

Properties	Requirement	Description and Notes
IncludeTotalIOs	Mandatory	
IncludeKBytesTransferred	Mandatory	
IncludeIOTimeCounter	Mandatory	
IncludeReadIOs	Mandatory	
IncludeReadHitIOs	Mandatory	
IncludeReadIOTimeCounter	Mandatory	
IncludeReadHitIOTimeCounter	Mandatory	
IncludeKBytesRead	Mandatory	
IncludeWriteIOs	Mandatory	
IncludeWriteHitIOs	Mandatory	
IncludeWriteIOTimeCounter	Mandatory	
IncludeWriteHitIOTimeCounter	Mandatory	
IncludeKBytesWritten	Mandatory	
IncludeIdleTimeCounter	Mandatory	
IncludeMaintOp	Mandatory	
IncludeMaintTimeCounter	Mandatory	
Caption	Optional	Not Specified in this version of the Profile.
Description	Optional	Not Specified in this version of the Profile.
IncludeStartStatisticTime	Optional	Not Specified in this version of the Profile.

8.8.7 CIM_BlockStatisticsManifestCollection (Client Defined)

An instance of a client defined CIM_BlockStatisticsManifestCollection defines the set of Manifests to be used in retrieval of Block statistics by the GetStatisticsCollection method.

CIM_BlockStatisticsManifestCollection is subclassed from CIM_SystemSpecificCollection.

In order for a client vdefined instance of the CIM_BlockStatisticsManifestCollection class to exist, then all the manifest collection manipulation functions shall be identified in the "SynchronousMethodsSupported" property of the CIM_BlockStatisticsCapabilities instance and a client shall have created a Manifest Collection.

Created By: Extrinsic: CreateManifestCollection

Modified By: Static

Deleted By: Static

Requirement: Clients can create manifests as identified by CIM_BlockStatisticsCapabilities.SynchronousMethodsSupported.

Table 126 describes class CIM_BlockStatisticsManifestCollection (Client Defined).

Table 126 - SMI Referenced Properties/Methods for CIM_BlockStatisticsManifestCollection (Client Defined)

Properties	Requirement	Description and Notes
InstanceID	Mandatory	
ElementName	Mandatory	A client defined user-friendly name for the manifest collection. It is set during creation of the Manifest Collection through the ElementName parameter of the CreateManifestCollection method.
IsDefault	Mandatory	Denotes whether or not this manifest collection is a provider defined default manifest collection. For the client defined manifest collections this is set to "false".
Caption	Optional	Not Specified in this version of the Profile.
Description	Optional	Not Specified in this version of the Profile.

8.8.8 CIM_BlockStatisticsManifestCollection (Provider Defined)

An instance of a default CIM_BlockStatisticsManifestCollection defines the set of Manifests that define the properties supported for each ElementType supported for the implementation. It can also be used by clients in retrieval of Block statistics by the GetStatisticsCollection method.

CIM_BlockStatisticsManifestCollection is subclassed from CIM_SystemSpecificCollection.

At least ONE CIM_BlockStatisticsManifestCollection shall exist if the Block Server Performance Subprofile is implemented. This would be the default manifest collection that defines the properties supported by the implementation.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 127 describes class CIM_BlockStatisticsManifestCollection (Provider Defined).

Table 127 - SMI Referenced Properties/Methods for CIM_BlockStatisticsManifestCollection (Provider Defined)

Properties	Requirement	Description and Notes
InstanceID	Mandatory	
ElementName	Mandatory	For the default manifest collection, this should be set to "DEFAULT".
IsDefault	Mandatory	Denotes whether or not this manifest collection is a provider defined default manifest collection. For the default manifest collection this is set to "true".
Caption	Optional	Not Specified in this version of the Profile.
Description	Optional	Not Specified in this version of the Profile.

8.8.9 CIM_BlockStatisticsService

The CIM_BlockStatisticsService class provides methods for statistics retrieval and Manifest Collection manipulation.

The CIM_BlockStatisticsService class is subclassed from CIM_Service.

There shall be an instance of the CIM_BlockStatisticsService, if the Block Server Performance Subprofile is implemented. It is not necessary to support any methods of the service, but the service shall be populated.

The methods that are supported can be determined from the SynchronousMethodsSupported and AsynchronousMethodsSupported properties of the CIM_BlockStatisticsCapabilities.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 128 describes class CIM_BlockStatisticsService.

Table 128 - SMI Referenced Properties/Methods for CIM_BlockStatisticsService

Properties	Requirement	Description and Notes
SystemCreationClassName	Mandatory	
SystemName	Mandatory	
CreationClassName	Mandatory	
Name	Mandatory	
Caption	Optional	Not Specified in this version of the Profile.
Description	Optional	Not Specified in this version of the Profile.
ElementName	Optional	Not Specified in this version of the Profile.
OperationalStatus	Optional	Not Specified in this version of the Profile.
StatusDescriptions	Optional	Not Specified in this version of the Profile.
InstallDate	Optional	Not Specified in this version of the Profile.
HealthState	Optional	Not Specified in this version of the Profile.
TimeOfLastStateChange	Optional	Not Specified in this version of the Profile.
OtherEnabledState	Optional	Not Specified in this version of the Profile.
EnabledDefault	Optional	Not Specified in this version of the Profile.
RequestedState	Optional	Not Specified in this version of the Profile.
EnabledState	Optional	Not Specified in this version of the Profile.
Started	Optional	Not Specified in this version of the Profile.
PrimaryOwnerName	Optional	Not Specified in this version of the Profile.
PrimaryOwnerContact	Optional	Not Specified in this version of the Profile.
GetStatisticsCollection()	Conditional	Conditional requirement: Clients can get statistics collections using the GetStatisticsCollection as identified by CIM_BlockStatisticsCapabilities.SynchronousMethodsSupported or Clients can get statistics collections using the GetStatisticsCollection as identified by CIM_BlockStatisticsCapabilities.AsynchronousMethodsSupported. Support for this method is conditional on CIM_BlockStatisticsCapabilities.SynchronousMethodsSupported or CIM_BlockStatisticsCapabilities.AsynchronousMethodsSupported containing '5' (GetStatisticsCollection). This method retrieves all statistics kept for the profile as directed by a manifest collection.

Table 128 - SMI Referenced Properties/Methods for CIM_BlockStatisticsService

Properties	Requirement	Description and Notes
CreateManifestCollection()	Conditional	Conditional requirement: Clients can create manifests as identified by CIM_BlockStatisticsCapabilities.SynchronousMethodsSupported or Clients can create manifests as identified by CIM_BlockStatisticsCapabilities.AsynchronousMethodsSupported. Support for this method is conditional on CIM_BlockStatisticsCapabilities.SynchronousMethodsSupported or CIM_BlockStatisticsCapabilities.AsynchronousMethodsSupported containing '6' (Manifest Creation). This method is used to create client defined manifest collections.
AddOrModifyManifest()	Conditional	Conditional requirement: Clients can modify manifests as identified by CIM_BlockStatisticsCapabilities.SynchronousMethodsSupported or Clients can modify manifests as identified by CIM_BlockStatisticsCapabilities.AsynchronousMethodsSupported. Support for this method is conditional on CIM_BlockStatisticsCapabilities.SynchronousMethodsSupported or CIM_BlockStatisticsCapabilities.AsynchronousMethodsSupported containing '7' (Manifest Modification). This method is used to add or modify block statistics manifests in a client defined manifest collection.
RemoveManifests()	Conditional	Conditional requirement: Clients can remove manifests as identified by CIM_BlockStatisticsCapabilities.SynchronousMethodsSupported or Clients can remove manifests as identified by CIM_BlockStatisticsCapabilities.AsynchronousMethodsSupported. Support for this method is conditional on CIM_BlockStatisticsCapabilities.SynchronousMethodsSupported or CIM_BlockStatisticsCapabilities.AsynchronousMethodsSupported containing '8' (Manifest Removal). This method is used to remove a block statistics manifest from a client defined manifest collection.
RequestStateChange()	Optional	Not Specified in this version of the Profile.
StopService()	Optional	Not Specified in this version of the Profile.
StartService()	Optional	Not Specified in this version of the Profile.

8.8.10 CIM_BlockStorageStatisticalData

The CIM_BlockStorageStatisticalData class defines the block statistics properties that may be kept for an metered element of the block storage entity (such as a ComputerSystem, StorageVolume, Port or Disk Drive).

CIM_BlockStorageStatisticalData is subclassed from CIM_StatisticalData.

Instances of this class will exist for each of the metered elements if the 'ElementTypesSupported' property of the CIM_BlockStatisticsCapabilities indicates that the metered element is supported. For example, 'Computer System' is identified in the 'ElementTypesSupported' property, then this indicates support for metering of the Top level computer system or 'Component Computer System'.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 129 describes class CIM_BlockStorageStatisticalData.

Table 129 - SMI Referenced Properties/Methods for CIM_BlockStorageStatisticalData

Properties	Requirement	Description and Notes
InstanceID	Mandatory	The InstanceID for BlockStorageStatisticalData instance shall be unique across all instances of the BlockStorageStatisticalData class.
StatisticTime	Mandatory	Time statistics table by object was last updated (Time Stamp in CIM 2.2 specification format).
ElementType	Mandatory	This value is required AND current version of SMI-S specifies the following values: ValueMap {"2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12"} Values { "Computer System", "Front-end Computer System", "Peer Computer System", "Back-end Computer System", "Front-end Port", "Back-end Port", "Volume", "Extent", "Disk Drive", "Arbitrary LUs", "Remote Replica Group"}.
TotalIOs	Mandatory	The cumulative count of I/Os for the object.
KBytesTransferred	Conditional	Conditional requirement: This property is required if the ElementType is 2, 6, 8, 9, 10, 11 or 12. The cumulative count of data transferred in KBytes (1024bytes = 1KByte). Note that this is mandatory for the Top level computer system, Front-end Ports, Volumes, Extents, Disk Drives, ArbitraryLUs and Remote Replica Groups, but is optional for the component computer systems and Back-end Ports.
IOTimeCounter	Optional	The cumulative elapsed I/O time(number of Clock Tick Intervals) for all cumulative I/Os as defined in "Total I/Os" above. I/O response time is added to this counter at the completion of each measured I/O using ClockTickInterval units. This value can be divided by number of IOs to obtain an average response time. Note that this is not SPECIFIED for CompositeExtents, ArbitraryLUs or Remote Replica Groups.
ReadIOs	Conditional	Conditional requirement: This property is required if the ElementType is 3, 4, 8 or 10. The cumulative count of all reads. Note that this is mandatory for "Front-end" and "Peer" component ComputerSystems, Volumes and Disk Drives, but it is optional for the Top level computer system. Note that this is not specified for Ports, CompositeExtents, "Back-end" component computer systems, ArbitraryLUs or Remote Replica Groups.
ReadHitOs	Conditional	Conditional requirement: This property is required if the ElementType is 3, 4 or 8. The cumulative count of all read cache hits (Reads from Cache). Note that this is mandatory for "Front-end" and "Peer" component ComputerSystems, and Volumes, but it is optional for the Top level computer system. Note that this is not specified for "Back-end" component computer systems, Ports, CompositeExtents, DiskDrives, ArbitraryLUs or Remote Replica Groups.

Table 129 - SMI Referenced Properties/Methods for CIM_BlockStorageStatisticalData

Properties	Requirement	Description and Notes
ReadIOTimeCounter	Optional	The cumulative elapsed time for all Read I/Os) for all cumulative Read I/Os. Note that this is optional for "Front-end" and "Peer" component ComputerSystems and the Top level computer system, Volumes and Disk Drives. Note that this is not specified for "Back-end" component computer systems, Ports, CompositeExtents, ArbitraryLUs or Remote Replica Groups.
ReadHitIOTimeCounter	Optional	The cumulative elapsed time for all Read I/Os read from cache for all cumulative Read I/Os. Note that this is optional for "Front-end" and "Peer" component ComputerSystems and the Top level computer system and Volumes. Note that this is not specified for "Back-end" component computer systems, Ports, CompositeExtents, DiskDrives, ArbitraryLUs or Remote Replica Groups.
KBytesRead	Optional	The cumulative count of data read in KBytes (1 024 bytes = 1 KByte). Note that this is optional for all ComputerSystems, Volumes, and Disk Drives. Note that this is not specified for Ports, CompositeExtents, ArbitraryLUs or Remote Replica Groups.
WriteIos	Conditional	Conditional requirement: This property is required if the ElementType is 3, 4 or 8. The cumulative count of all writes. Note that this is mandatory for "Front-end" and "Peer" component ComputerSystems and Volumes, but it is optional for the Top level computer system and Disk Drives. Note that this is not specified for "Back-end" component computer systems, Ports, CompositeExtents, ArbitraryLUs or Remote Replica Groups.
WriteHitIos	Conditional	Conditional requirement: This property is required if the ElementType is 3, 4 or 8. The cumulative count of Write Cache Hits (Writes that went directly to Cache without blocking). Note that this is mandatory for "Front-end" and "Peer" component ComputerSystems and Volumes, but it is optional for the Top level computer system. Note that this is not specified for "Back-end" component computer systems, Ports, CompositeExtents, DiskDrives, ArbitraryLUs or Remote Replica Groups.
WriteIOTimeCounter	Optional	The cumulative elapsed time for all Write I/Os for all cumulative Writes. Note that this is optional for "Front-end" and "Peer" component ComputerSystems and the Top level computer system and Volumes and Disks Drives. Note that this is not specified for "Back-end" component computer systems, Ports, CompositeExtents, ArbitraryLUs or Remote Replica Groups.

Table 129 - SMI Referenced Properties/Methods for CIM_BlockStorageStatisticalData

Properties	Requirement	Description and Notes
WriteHitIOTimeCounter	Optional	The cumulative elapsed time for all Write I/Os written to cache for all cumulative Write I/Os. Note that this is optional for "Front-end" and "Peer" component ComputerSystems and the Top level computer system and Volumes. Note that this is not specified for "Back-end" component computer systems, Ports, CompositeExtents, DiskDrives, ArbitraryLUs or Remote Replica Groups.
KBytesWritten	Optional	The cumulative count of data written in KBytes (1 024 bytes = 1 KByte). Note that this is optional for all ComputerSystems, Volumes and Disk Drives. Note that this is not specified for Ports, CompositeExtents, ArbitraryLUs or Remote Replica Groups.
IdleTimeCounter	Optional	The cumulative elapsed idle time using ClockTickInterval units (Cumulative Number of Time Units for all idle time in the array). Note that this is optional for "Back-end" component ComputerSystems, Front end Ports, Volumes, Extents and Disk Drives. Note that this is not specified for back-end Ports, Top level computer system, "Front-end" and "Peer" component computer systems, ArbitraryLUs or Remote Replica Groups.
MaintOp	Optional	The cumulative count of all disk maintenance operations (SCSI commands such as: Verify, skip-mask, XOR read, XOR write-read, etc.) This is needed to understand the load on the disks that may interfere with normal read and write operations. Note that this is optional for Extents and Disk Drives. Note that this is not specified for ComputerSystems, Ports, Volumes, ArbitraryLUs or Remote Replica Groups.
MaintTimeCounter	Optional	The cumulative elapsed disk maintenance time, maintenance response time is added to this counter at the completion of each measured maintenance operation using ClockTickInterval units. Note that this is optional for Extents and Disk Drives. Note that this is not specified for ComputerSystems, Ports, Volumes, ArbitraryLUs or Remote Replica Groups.
Caption	Optional	Not Specified in this version of the Profile.
Description	Optional	Not Specified in this version of the Profile.
ElementName	Optional	Not Specified in this version of the Profile.
StartStatisticTime	Optional	Not Specified in this version of the Profile.
ResetSelectedStats()	Optional	Not Specified in this version of the Profile.

8.8.11 CIM_ElementCapabilities

CIM_ElementCapabilities represents the association between ManagedElements (i.e., CIM_BlockStatisticsService) and their Capabilities (e.g., CIM_BlockStatisticsCapabilities). Note that the cardinality of the ManagedElement reference is Min(1), Max(1). This cardinality mandates the instantiation of the CIM_ElementCapabilities association for the referenced instance of Capabilities. ElementCapabilities describes the existence requirements and

context for the referenced instance of ManagedElement. Specifically, the ManagedElement shall exist and provides the context for the Capabilities.

CIM_ElementCapabilities is not subclassed from anything.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 130 describes class CIM_ElementCapabilities.

Table 130 - SMI Referenced Properties/Methods for CIM_ElementCapabilities

Properties	Requirement	Description and Notes
ManagedElement	Mandatory	The managed element (BlockStatisticsService).
Capabilities	Mandatory	The Capabilities instance associated with the BlockStatisticsService.

8.8.12 CIM_ElementStatisticalData (Back end Port Stats)

CIM_ElementStatisticalData is an association that relates a back end port to its statistics. Note that the cardinality of the ManagedElement reference is Min(1), Max(1). This cardinality mandates the instantiation of the CIM_ElementStatisticalData association for the referenced instance of BlockStatistics. ElementStatisticalData describes the existence requirements and context for the BlockStatistics, relative to a specific back end port.

CIM_ElementStatisticalData is not subclassed from anything.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Back end port statistics support.

Table 131 describes class CIM_ElementStatisticalData (Back end Port Stats).

Table 131 - SMI Referenced Properties/Methods for CIM_ElementStatisticalData (Back end Port Stats)

Properties	Requirement	Description and Notes
ManagedElement	Mandatory	A reference to a back end port for which the Statistics apply.
Stats	Mandatory	A reference to the BlockStorageStatisticalData that hold the statistics for the Port.

8.8.13 CIM_ElementStatisticalData (Component System Stats)

CIM_ElementStatisticalData is an association that relates a component ComputerSystem to its statistics. Note that the cardinality of the ManagedElement reference is Min(1), Max(1). This cardinality mandates the instantiation of the CIM_ElementStatisticalData association for the referenced instance of BlockStatistics. ElementStatisticalData describes the existence requirements and context for the BlockStatistics, relative to a specific component ComputerSystem.

CIM_ElementStatisticalData is not subclassed from anything.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Component Systems statistics support.

Table 132 describes class CIM_ElementStatisticalData (Component System Stats).

Table 132 - SMI Referenced Properties/Methods for CIM_ElementStatisticalData (Component System Stats)

Properties	Requirement	Description and Notes
ManagedElement	Mandatory	A reference to a component ComputerSystem for which the Statistics apply.
Stats	Mandatory	A reference to the BlockStorageStatisticalData that hold the statistics for the ComputerSystem.

8.8.14 CIM_ElementStatisticalData (Disk Stats)

CIM_ElementStatisticalData is an association that relates a StorageExtent (Disk Drive) to its statistics. Note that the cardinality of the ManagedElement reference is Min(1), Max(1). This cardinality mandates the instantiation of the CIM_ElementStatisticalData association for the referenced instance of BlockStatistics. ElementStatisticalData describes the existence requirements and context for the BlockStatistics, relative to a specific StorageExtent of a Disk Drive.

CIM_ElementStatisticalData is not subclassed from anything.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Disk Drive statistics support.

Table 133 describes class CIM_ElementStatisticalData (Disk Stats).

Table 133 - SMI Referenced Properties/Methods for CIM_ElementStatisticalData (Disk Stats)

Properties	Requirement	Description and Notes
ManagedElement	Mandatory	A reference to a Disk Drive StorageExtent for which the Statistics apply.
Stats	Mandatory	A reference to the BlockStorageStatisticalData that hold the statistics for the Disk Drive.

8.8.15 CIM_ElementStatisticalData (Extent Stats)

CIM_ElementStatisticalData is an association that relates a StorageExtent (CompositeExtent) to its statistics. Note that the cardinality of the ManagedElement reference is Min(1), Max(1). This cardinality mandates the instantiation of the CIM_ElementStatisticalData association for the referenced instance of BlockStatistics. ElementStatisticalData describes the existence requirements and context for the BlockStatistics, relative to a specific StorageExtent.

CIM_ElementStatisticalData is not subclassed from anything.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Extent statistics support.

Table 134 describes class CIM_ElementStatisticalData (Extent Stats).

Table 134 - SMI Referenced Properties/Methods for CIM_ElementStatisticalData (Extent Stats)

Properties	Requirement	Description and Notes
ManagedElement	Mandatory	A reference to a StorageExtent for which the Statistics apply.
Stats	Mandatory	A reference to the BlockStorageStatisticalData that hold the statistics for the StorageExtent.

8.8.16 CIM_ElementStatisticalData (Front end Port Stats)

CIM_ElementStatisticalData is an association that relates a target port to its statistics. Note that the cardinality of the ManagedElement reference is Min(1), Max(1). This cardinality mandates the instantiation of the CIM_ElementStatisticalData association for the referenced instance of BlockStatistics. ElementStatisticalData describes the existence requirements and context for the BlockStatistics, relative to a specific target port.

CIM_ElementStatisticalData is not subclassed from anything.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Front-end port statistics support.

Table 135 describes class CIM_ElementStatisticalData (Front end Port Stats).

Table 135 - SMI Referenced Properties/Methods for CIM_ElementStatisticalData (Front end Port Stats)

Properties	Requirement	Description and Notes
ManagedElement	Mandatory	A reference to a target port for which the Statistics apply.
Stats	Mandatory	A reference to the BlockStorageStatisticalData that hold the statistics for the Port.

8.8.17 CIM_ElementStatisticalData (Logical Disk Stats)

CIM_ElementStatisticalData is an association that relates a LogicalDisk to its statistics. Note that the cardinality of the ManagedElement reference is Min(1), Max(1). This cardinality mandates the instantiation of the CIM_ElementStatisticalData association for the referenced instance of BlockStatistics. ElementStatisticalData describes the existence requirements and context for the BlockStatistics, relative to a specific logical disk.

CIM_ElementStatisticalData is not subclassed from anything.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Volume statistics support in Volume Management Profiles.

Table 136 describes class CIM_ElementStatisticalData (Logical Disk Stats).

Table 136 - SMI Referenced Properties/Methods for CIM_ElementStatisticalData (Logical Disk Stats)

Properties	Requirement	Description and Notes
ManagedElement	Mandatory	A reference to a LogicalDisk for which the Statistics apply.
Stats	Mandatory	A reference to the BlockStorageStatisticalData that hold the statistics for the LogicalDisk.

8.8.18 CIM_ElementStatisticalData (Remote Copy Stats)

CIM_ElementStatisticalData is an association that relates a Network to its statistics. Note that the cardinality of the ManagedElement reference is Min(1), Max(1). This cardinality mandates the instantiation of the CIM_ElementStatisticalData association for the referenced instance of BlockStatistics. ElementStatisticalData describes the existence requirements and context for the BlockStatistics, relative to a specific Network.

CIM_ElementStatisticalData is not subclassed from anything.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Remote Copy statistics support.

Table 137 describes class CIM_ElementStatisticalData (Remote Copy Stats).

Table 137 - SMI Referenced Properties/Methods for CIM_ElementStatisticalData (Remote Copy Stats)

Properties	Requirement	Description and Notes
ManagedElement	Mandatory	A reference to a Network (remote replication group) for which the Statistics apply.
Stats	Mandatory	A reference to the BlockStorageStatisticalData that hold the statistics for the Network.

8.8.19 CIM_ElementStatisticalData (Top Level System Stats)

CIM_ElementStatisticalData is an association that relates a top level ComputerSystem to its statistics. Note that the cardinality of the ManagedElement reference is Min(1), Max(1). This cardinality mandates the instantiation of the CIM_ElementStatisticalData association for the referenced instance of BlockStatistics. ElementStatisticalData describes the existence requirements and context for the BlockStatistics, relative to a specific ComputerSystem.

CIM_ElementStatisticalData is not subclassed from anything.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Top level system statistics support.

Table 138 describes class CIM_ElementStatisticalData (Top Level System Stats).

Table 138 - SMI Referenced Properties/Methods for CIM_ElementStatisticalData (Top Level System Stats)

Properties	Requirement	Description and Notes
ManagedElement	Mandatory	A reference to the top level ComputerSystem for which the Statistics apply.
Stats	Mandatory	A reference to the BlockStorageStatisticalData that hold the statistics for the ComputerSystem.

8.8.20 CIM_ElementStatisticalData (Volume Stats)

CIM_ElementStatisticalData is an association that relates a StorageVolume to its statistics. Note that the cardinality of the ManagedElement reference is Min(1), Max(1). This cardinality mandates the instantiation of the CIM_ElementStatisticalData association for the referenced instance of BlockStatistics. ElementStatisticalData describes the existence requirements and context for the BlockStatistics, relative to a specific volume.

CIM_ElementStatisticalData is not subclassed from anything.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Volume statistics support or Referenced from Array - StorageVolume is mandatory or Referenced from Storage Virtualizer - StorageVolume is mandatory.

Table 139 describes class CIM_ElementStatisticalData (Volume Stats).

Table 139 - SMI Referenced Properties/Methods for CIM_ElementStatisticalData (Volume Stats)

Properties	Requirement	Description and Notes
ManagedElement	Mandatory	A reference to a StorageVolume for which the Statistics apply.
Stats	Mandatory	A reference to the BlockStorageStatisticalData that hold the statistics for the StorageVolume.

8.8.21 CIM_HostedCollection (Client Defined)

CIM_HostedCollection defines a SystemSpecificCollection in the context of a scoping 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. In the Block Server Performance Subprofile, it is used to associate a client defined BlockStatisticsManifestCollections to the top level Computer System.

CIM_HostedCollection is subclassed from CIM_HostedDependency.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Clients can create manifests as identified by CIM_BlockStatisticsCapabilities.SynchronousMethodsSupported or Clients can create manifests as identified by CIM_BlockStatisticsCapabilities.AsynchronousMethodsSupported.

Table 140 describes class CIM_HostedCollection (Client Defined).

Table 140 - SMI Referenced Properties/Methods for CIM_HostedCollection (Client Defined)

Properties	Requirement	Description and Notes
Antecedent	Mandatory	The top level ComputerSystem of the profile.
Dependent	Mandatory	A client defined BlockStatisticsManifestCollection.

8.8.22 CIM_HostedCollection (Default)

CIM_HostedCollection defines a SystemSpecificCollection in the context of a scoping 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. In the Block Server Performance Subprofile, it is used to associate the default BlockStatisticsManifestCollection to the top level Computer System.

CIM_HostedCollection is subclassed from CIM_HostedDependency.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 141 describes class CIM_HostedCollection (Default).

Table 141 - SMI Referenced Properties/Methods for CIM_HostedCollection (Default)

Properties	Requirement	Description and Notes
Antecedent	Mandatory	The top level ComputerSystem of the profile.
Dependent	Mandatory	The provider defined BlockStatisticsManifestCollection.

8.8.23 CIM_HostedCollection (Provider Supplied)

CIM_HostedCollection defines a SystemSpecificCollection in the context of a scoping 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. In the Block Server Performance Subprofile, it is used to associate the StatisticsCollection to the top level Computer System.

CIM_HostedCollection is subclassed from CIM_HostedDependency.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 142 describes class CIM_HostedCollection (Provider Supplied).

Table 142 - SMI Referenced Properties/Methods for CIM_HostedCollection (Provider Supplied)

Properties	Requirement	Description and Notes
Antecedent	Mandatory	The top level ComputerSystem of the profile.
Dependent	Mandatory	The StatisticsCollection.

8.8.24 CIM_HostedService

CIM_HostedService is an association between a Service (CIM_BlockStatisticsService) and the System (ComputerSystem) on which the functionality resides. 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.

CIM_HostedService is subclassed from CIM_HostedDependency.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 143 describes class CIM_HostedService.

Table 143 - SMI Referenced Properties/Methods for CIM_HostedService

Properties	Requirement	Description and Notes
Antecedent	Mandatory	The hosting System.
Dependent	Mandatory	The Service hosted on the System.

8.8.25 CIM_MemberOfCollection (Member of client defined collection)

This use of MemberOfCollection is to Collect all Manifests instances in a client defined manifest collection.

Created By: Extrinsic: AddOrModifyManifest

Modified By: Static

Deleted By: Extrinsic: RemoveManifest

Requirement: Clients can modify manifests as identified by CIM_BlockStatisticsCapabilities.SynchronousMethodsSupported.

Table 144 describes class CIM_MemberOfCollection (Member of client defined collection).

Table 144 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Member of client defined collection)

Properties	Requirement	Description and Notes
Collection	Mandatory	A client defined manifest collection.
Member	Mandatory	The individual Manifest Instance that is part of the set.

8.8.26 CIM_MemberOfCollection (Member of pre-defined collection)

This use of MemberOfCollection is to Collect all Manifests instances in the default manifest collection.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 145 describes class CIM_MemberOfCollection (Member of pre-defined collection).

**Table 145 - SMI Referenced Properties/Methods for CIM_MemberOfCollection
(Member of pre-defined collection)**

Properties	Requirement	Description and Notes
Collection	Mandatory	The provider defined default manifest collection.
Member	Mandatory	The individual Manifest Instance that is part of the set.

8.8.27 CIM_MemberOfCollection (Member of statistics collection)

This use of MemberOfCollection is to collect all BlockStorageStatisticalData instances (in the StatisticsCollection). Each association is created as a side effect of the metered object getting created.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 146 describes class CIM_MemberOfCollection (Member of statistics collection).

**Table 146 - SMI Referenced Properties/Methods for CIM_MemberOfCollection
(Member of statistics collection)**

Properties	Requirement	Description and Notes
Collection	Mandatory	The default manifest collection.
Member	Mandatory	The individual Manifest Instance that is part of the set.

8.8.28 CIM_StatisticsCollection

The CIM_StatisticsCollection collects all block statistics kept by the profile. There is one instance of the CIM_StatisticsCollection class and all individual element statistics can be accessed by using association traversal (using MemberOfCollection) from the StatisticsCollection.

CIM_StatisticsCollection is subclassed from CIM_SystemSpecificCollection.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 147 describes class CIM_StatisticsCollection.

Table 147 - SMI Referenced Properties/Methods for CIM_StatisticsCollection

Properties	Requirement	Description and Notes
InstanceID	Mandatory	
ElementName	Mandatory	
SampleInterval	Mandatory	Minimum recommended polling interval for an array, storage virtualizer system or volume manager. It is set by the provider and cannot be modified.

Table 147 - SMI Referenced Properties/Methods for CIM_StatisticsCollection

Properties	Requirement	Description and Notes
TimeLastSampled	Mandatory	Time statistics table by object was last updated (Time Stamp in SMI 2.2 specification format).
Caption	Optional	Not Specified in this version of the Profile.
Description	Optional	Not Specified in this version of the Profile.

8.8.29 SNIA_BlockStatisticsCapabilities

Experimental. This is a subclass of CIM_BlockStatisticsCapabilities that adds the SupportedFeatures property.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 148 describes class SNIA_BlockStatisticsCapabilities.

Table 148 - SMI Referenced Properties/Methods for SNIA_BlockStatisticsCapabilities

Properties	Requirement	Description and Notes
InstanceID	Mandatory	
ElementName	Mandatory	
ElementTypesSupported	Mandatory	
SynchronousMethodsSupported	Mandatory	
AsynchronousMethodsSupported	Optional	
ClockTickInterval	Mandatory	
SupportedFeatures	Optional	This is an array identifying features supported by the implementation. The valid values are '2' (none) or '3' (Client Defined Sequence).

8.8.30 SNIA_BlockStatisticsManifest (Client Defined)

Experimental. This is a subclass of CIM_BlockStatisticsManifest that adds the CSVSequence property.

Created By: Extrinsic: AddOrModifyManifest

Modified By: Extrinsic: AddOrModifyManifest

Deleted By: Extrinsic: RemoveManifest

Requirement: Optional.

Table 149 describes class SNIA_BlockStatisticsManifest (Client Defined).

Table 149 - SMI Referenced Properties/Methods for SNIA_BlockStatisticsManifest (Client Defined)

Properties	Requirement	Description and Notes
ElementName	Mandatory	
InstanceID	Mandatory	
ElementType	Mandatory	

Table 149 - SMI Referenced Properties/Methods for SNIA_BlockStatisticsManifest (Client Defined)

Properties	Requirement	Description and Notes
IncludeStatisticTime	Mandatory	
IncludeTotalIOs	Mandatory	
IncludeKBytesTransferred	Mandatory	
IncludeIOTimeCounter	Mandatory	
IncludeReadIOs	Mandatory	
IncludeReadHitIOs	Mandatory	
IncludeReadIOTimeCounter	Mandatory	
IncludeReadHitIOTimeCounter	Mandatory	
IncludeKBytesRead	Mandatory	
IncludeWriteIOs	Mandatory	
IncludeWriteHitIOs	Mandatory	
IncludeWriteIOTimeCounter	Mandatory	
IncludeWriteHitIOTimeCounter	Mandatory	
IncludeKBytesWritten	Mandatory	
IncludeIdleTimeCounter	Mandatory	
IncludeMaintOp	Mandatory	
IncludeMaintTimeCounter	Mandatory	
CSVSequence	Mandatory	An array of strings that define a sequence of BlockStorageStatisticalData property names. The sequence is the sequence that data is to be returned on a GetStatisticsCollection request using this manifest. The first three elements of this array should be "InstanceID", "ElementType" and "StatisticsTime" to allow applications to match the ElementType of the Manifest with the BlockStorageStatisticalData CSV record. For BlockStatisticsManifest (Client Defined) this shall be the sequence desired by the client.

8.8.31 SNIA_BlockStatisticsManifest (Provider Support)

Experimental. This is a subclass of CIM_BlockStatisticsManifest that adds the CSVSequence property.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 150 describes class SNIA_BlockStatisticsManifest (Provider Support).

Table 150 - SMI Referenced Properties/Methods for SNIA_BlockStatisticsManifest (Provider Support)

Properties	Requirement	Description and Notes
ElementName	Mandatory	
InstanceID	Mandatory	
ElementType	Mandatory	
IncludeStatisticTime	Mandatory	
IncludeTotalIOs	Mandatory	
IncludeKBytesTransferred	Mandatory	
IncludeIOTimeCounter	Mandatory	
IncludeReadIOs	Mandatory	
IncludeReadHitIOs	Mandatory	
IncludeReadIOTimeCounter	Mandatory	
IncludeReadHitIOTimeCounter	Mandatory	
IncludeKBytesRead	Mandatory	
IncludeWriteIOs	Mandatory	
IncludeWriteHitIOs	Mandatory	
IncludeWriteIOTimeCounter	Mandatory	
IncludeWriteHitIOTimeCounter	Mandatory	
IncludeKBytesWritten	Mandatory	
IncludeIdleTimeCounter	Mandatory	
IncludeMaintOp	Mandatory	
IncludeMaintTimeCounter	Mandatory	
CSVSequence	Mandatory	An array of strings that define a sequence of BlockStorageStatisticalData property names. The sequence is the sequence that data is to be returned on a GetStatisticsCollection request using this manifest. The first three elements of this array shall be "InstanceID", "ElementType" and "StatisticsTime" to allow applications to match the ElementType of the Manifest with the BlockStorageStatisticalData CSV record. For BlockStatisticsManifest (Provider Support) this shall be the default sequence provided by the provider.

STABLE

EXPERIMENTAL

9 CKD Block Services Profile

9.1 Description

9.1.1 Synopsis

Profile Name: CKD Block Services

Version: 1.2.0

Organization: SNIA

CIM schema version: 2.15

Central Class: CIM_StoragePool

Scoping Class: CIM_System

9.1.2 Overview

The CKD Block Services Profile models CKD (Count Key Data) storage of a block server storage system. CKD storage is storage that is formatted to support Count and Key fields to support mainframe access. CKD storage is at the StorageVolume level (which means the StorageVolume is access using single byte FC protocols) or at the StoragePool level (that is, a StoragePool may be dedicated to holding CKD StorageVolumes).

The CKD Block Services Profile is a component profile (subprofile) that provides a way for storage profiles to model mainframe storage. With this support a client will be able to distinguish non-CKD storage that is provided for non-CKD access from CKD storage that is provided for mainframe access. This is an important distinction for management, since storage that is available to one (e.g., SCSI access) is typically not usable by the other (e.g., mainframe access), although there are some devices that do support sharing a volume across CKD and non-CKD hosts. Similarly, management functions for other functions of block servers (e.g., masking and mapping) are somewhat different for CKD storage than non-CKD storage. So, it is important for management applications to be aware of the distinctions.

The CKD Block Services requires and specializes the Block Services Package. That is, the functions of the Block Services Package apply for CKD storage as well as non-CKD storage. The CKD Block Services Profile extends the model for CKD storage.

9.1.3 Implementation

9.1.3.1 Block Services Support for CKD Storage

Some profile implementations may support Extended Count Key Data formatted storage. This support is provided using existing classes, but adds some new properties as illustrated in Figure 45: "Block Services Support for Count Key Data Storage".

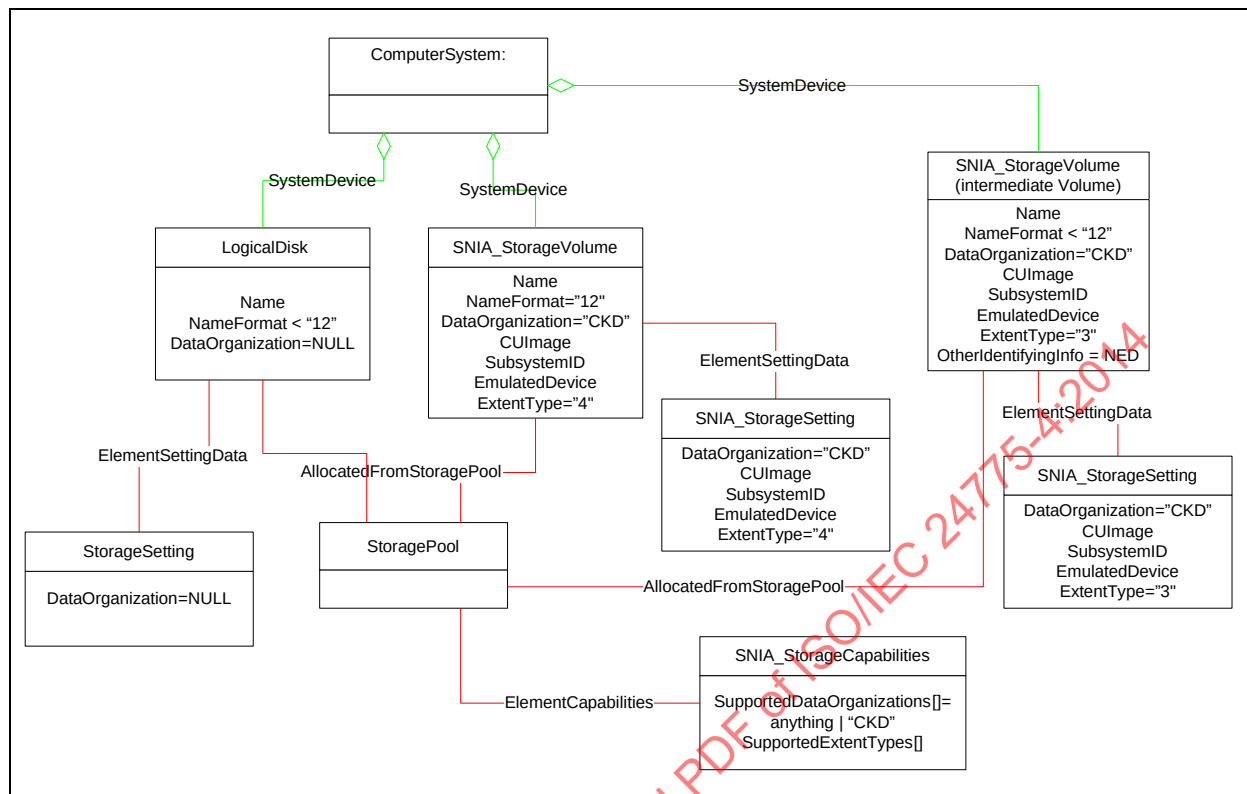


Figure 45 - Block Services Support for Count Key Data Storage

CKD storage may apply to StoragePools (via StorageCapabilities), StorageVolumes or LogicalDisks. CKD storage is indicated by the DataOrganization property in StorageVolume and LogicalDisk classes. For SMI-S the values of this property shall be "4" for CKD Volumes (or LogicalDisks). The capability of a StoragePool to support either (or both) non-CKD or CKD volumes is indicated by the SupportedDataOrganizations[] property of StorageCapabilities associated to the StoragePool.

DataOrganization can be specified on StorageSetting to indicate that an CKD Volume is desired on either of the Volume creation methods. If this property is left as null, it will be set according to the StoragePool that is being used. If the StoragePool supports both non-CKD and CKD storage, then the default will be to create a non-CKD volume (or LogicalDisk) for backward compatibility. This property exists in StorageVolume, LogicalDisk, and StorageSetting classes.

An additional difference between non-CKD and CKD Volumes are the NameFormats supported. For CKD Volumes, the volumes follow a Node Element Descriptor (NED) format. For non-CKD volumes there are a variety of formats that may be supported.

Certain instrumentation supports the use of a volume for both CKD and non-CKD hosts. These volumes are called Intermediate volumes in this International Standard. A StorageVolume can be classified as non-CKD, CKD, or both. The StorageVolume.DataOrganization property indicates the data format of the volume, while the new StorageVolume ExtentType property indicates the type of host access allowed (CKD, non-CKD, both). Since this volume is shared across CKD and non-CKD hosts, it has a different name for each host. The Name property is used by Intermediate volumes for non-CKD hosts to provide for backwards compatibility, and the OtherIdentifyingInfo[] and IdentifyingDescriptions[] holds the CKD name and format information.

There is also a CUImage property on both the SNIA_StorageVolume and the SNIA_StorageSetting. In the SB architecture and CKD access the CKD Volume has a "home" ProtocolController (in a Masking and Mapping sense). This property is covered in more detail

in (need a Masking and Mapping reference here). But an CKD Volume cannot exist without an associated CUIImage (ProtocolController). This is accommodated by the CUIImage property on StorageSetting. That is, on creation of an CKD Volume the CUIImage parameter is passed as part of the StorageSetting for the Volume being created. The CUIImage in the SNIA_StorageSetting is the CUIImage requested and the CUIImage in the SNIA_StorageVolume is the CUIImage assigned. CUIImage is not supported for LogicalDisks.

A host can see more than 16 CU images by changing the SSID associated with the image. For example, there can be two CU images with the same image number but with different SSIDs. Thus, the same CU image numbers can be in use multiple times within the array and the host as long as each image has a unique SubsystemID. The second CU image with the same number is known as a "split".

Mainframe systems use the SubsystemID to locate physical disk controllers, and all devices in the CU image shall have the same SubsystemID. If the CU image that is specified does not exist yet, the SubsystemID of the first device is used as the SubsystemID of the CU image. If the CU image already exists and contains other devices (and thus a SubsystemID), the SubsystemIDs of the newly mapped devices are changed to match the existing SubsystemID of the CU image.

9.1.3.2 Use Cases for CKD Storage

9.1.3.2.1 Summarize Pools and Capacities by SupportedDataOrganizations

Primordial StoragePools may be capable of supporting non-CKD, CKD or both non-CKD and CKD storage. This can be determined by inspecting the SupportedDataOrganizations property of the StorageCapabilities of the primordial StoragePool. If the property is NULL or not "4", then the pool only supports non-CKD storage and all concrete StoragePools allocated from this Primordial StoragePool shall only support non-CKD storage. Similarly, if the property only identifies "4" (Count Key Data), then the pool only supports CKD storage and all concrete StoragePools allocated from this primordial StoragePool shall only support CKD storage.

If the StorageCapabilities.SupportedDataOrganizations property for primordial StoragePool identifies both "4" (Count Key Data) and something else (including NULL), then the storage allocated from the pool can be either non-CKD or CKD storage. It will be necessary to follow the AllocatedFromStoragePool association to the concrete StoragePools above the primordial StoragePool. As the client moves up the AllocatedFromStoragePool association, it would keep track of the SpaceConsumed value in the AllocatedFromStoragePool. If all concrete StoragePools are also capable of both non-CKD and CKD storage, then the primordial capacity of the storage is considered capable of supporting both non-CKD and CKD Volumes (or LogicalDisks).

If, however, the client reaches a concrete StoragePool that is only capable of supporting non-CKD or CKD storage, then the SpaceConsumed value by that StoragePool would be considered either non-CKD or CKD. It may be necessary to "pro-rate" the SpaceConsumed value to determine the actual primordial storage that has been allocated to non-CKD or CKD.

9.1.3.2.2 Find the Capacity of CKD Capable Storage

Building on the previous use case, a client would determine the capacity of primordial StoragePools that are only CKD capable (that is, StorageCapabilities.SupportedDataOrganization = "4" and only "4"). This capacity is dedicated to CKD storage.

Next the client would consider primordial StoragePools that are capable of both non-CKD and CKD storage. The client would inspect the concrete StoragePools that are allocated from those primordial StoragePools. If any are identified as CKD only, the SpaceConsumed property on the AllocatedFromStoragePool will indicate the primordial storage that is dedicated to CKD.

If the concrete StoragePool just above the primordial StoragePool is also capable of supporting non-CKD or CKD storage, divide the SpaceConsumed value by the TotalManagedSpace value of the concrete StoragePool and save this "multiplier".

The client would continue executing the previous step until it finds a concrete StoragePool that only supports non-CKD storage. At this point, the client would multiply all the multipliers it has saved away to derive the amount of primordial space that has been dedicated to non-CKD storage. This value would be subtracted from the TotalManagedSpace value of the primordial StoragePool to determine the primordial capacity available for CKD storage. The client would execute this logic on all upper level concrete StoragePools that are identified as non-CKD only to get the remaining primordial capacity available for CKD storage.

9.1.3.2.3 Create an CKD Volume

To create an CKD Volume (or LogicalDisk) a client would create a StorageSetting (or select a SettingAssociated to Capabilities) with DataOrganization set to "4" and the CUIImage set to a valid CUIImage value.

With the appropriate CKD Volume Setting the client would issue either CreateOrModifyElementFromStoragePool or CreateOrModifyElementFromElements.

9.2 Health and Fault Management Consideration

No change for CKD.

9.3 Cascading Considerations

No change for CKD.

9.4 Supported Profiles, Subprofiles, and Packages

Table 151 describes the supported profiles for CKD Block Services.

Table 151 - Supported Profiles for CKD Block Services

Profile Name	Organization	Version	Requirement	Description
Job Control	SNIA	1.5.0	Optional	
Extent Composition	SNIA	1.5.0	Optional	
Block Services	SNIA	1.5.0	Mandatory	

9.5 Methods of the Profile

All methods of the Block Services Package should work for CKD storage (subject to restrictions of particular profile implementations).

9.6 Client Considerations and Recipes

No change for CKD.

9.7 Registered Name and Version

CKD Block Services version 1.3.0 (Component Profile)

Specializes SNIA Block Services version 1.5.0

9.8 CIM Elements

9.8.1 General

Table 152 describes the CIM elements for CKD Block Services.

Table 152 - CIM Elements for CKD Block Services

Element Name	Requirement	Description
9.8.2 CIM_AllocatedFromStoragePool	Mandatory	The CIM_AllocatedFromStoragePool associates a Storage Element (Volume, LogicalDisk or StoragePool) to its parent StoragePool. There are no enhancements for CKD.
9.8.3 CIM_AllocatedFromStoragePool (Pool from Pool)	Mandatory	AllocatedFromStoragePool.
9.8.4 CIM_AllocatedFromStoragePool (Volume or LogicalDisk from Pool)	Conditional	Conditional requirement: Referenced from Array - StorageVolume is mandatory or Referenced from Storage Virtualizer - StorageVolume is mandatory or Referenced from Host Hardware RAID Controller - StorageVolume is mandatory. AllocatedFromStoragePool.
9.8.5 CIM_ElementCapabilities	Mandatory	The CIM_ElementCapabilities associates a StoragePool or StorageConfigurationService to its Capabilities (StorageCapabilities and StorageConfigurationCapabilities). There are no enhancements for CKD.
9.8.6 CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StorageVolume or LogicalDisk)	Optional	Expressed the ability for the element to be named or have its state changed.
9.8.7 CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StoragePool)	Optional	Expressed the ability for the element to be named or have its state changed.
9.8.8 CIM_ElementCapabilities (StorageCapabilities to StorageConfigurationService)	Optional	Associates StorageCapabilities with StorageConfigurationService. This StorageCapabilities shall represent the capabilities of the entire implementation.
9.8.9 CIM_ElementCapabilities (StorageCapabilities to StoragePool)	Mandatory	Associates StorageCapabilities with StoragePool. This StorageCapabilities shall represent the capabilities of the StoragePool to which it is associated.
9.8.10 CIM_ElementCapabilities (StorageConfigurationCapabilities to StorageConfigurationService)	Mandatory	Associates StorageConfigurationCapabilities with StorageConfigurationService.
9.8.11 CIM_ElementCapabilities (StorageConfigurationCapabilities to concrete StoragePool)	Optional	Associates StorageConfigurationCapabilities with StoragePool.
9.8.12 CIM_ElementCapabilities (StorageConfigurationCapabilities to primordial StoragePool)	Optional	Associates StorageConfigurationCapabilities with StoragePool.
9.8.13 CIM_ElementCapabilities (Used to declare the naming capabilities of the StoragePool)	Optional	Associates EnabledLogicalElementCapabilities with StorageConfigurationService.
9.8.14 CIM_ElementCapabilities (Used to declare the naming capabilities of the StorageVolume or LogicalDisk)	Optional	Associates EnabledLogicalElementCapabilities with StorageConfigurationService.
9.8.15 CIM_ElementSettingData	Mandatory	The CIM_ElementSettingData associates a StorageVolume (or LogicalDisk) to its StorageSetting. There are no enhancements for CKD.

Table 152 - CIM Elements for CKD Block Services

Element Name	Requirement	Description
9.8.16 CIM_EnabledLogicalElementCapabilities (For StorageConfigurationService)	Optional	This class is used to express the naming and possible requested state change possibilities for storage elements.
9.8.17 CIM_EnabledLogicalElementCapabilities (For StoragePool)	Optional	This class is used to express the naming and possible requested state change possibilities for storage pools.
9.8.18 CIM_FilterCollection (Block Services Predefined FilterCollection)	Conditional	Experimental. Conditional requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='5' (Predefined Filter Collections). This is a collection of predefined IndicationFilters to which a client may subscribe.
9.8.19 CIM_HostedCollection (System to predefined IndicationFilters)	Conditional	Experimental. Conditional requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='5' (Predefined Filter Collections).
9.8.20 CIM_HostedService	Optional	This associates the StorageConfigurationService to its scoping system. This is unchanged for CKD storage.
9.8.21 CIM_HostedStoragePool	Mandatory	
9.8.22 CIM_IndicationFilter (Logical Disk Creation)	Conditional	Experimental. Conditional requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='3' (Predefined Filters). This is the 'pre-defined' CIM_IndicationFilter instance for the addition of a new LogicalDisk instance.
9.8.23 CIM_IndicationFilter (Logical Disk Deletion)	Conditional	Experimental. Conditional requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='3' (Predefined Filters). This is the 'pre-defined' CIM_IndicationFilter instance for the deletion of a LogicalDisk instance.
9.8.24 CIM_IndicationFilter (Logical Disk OperationalStatus)	Conditional	Experimental. Conditional requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='3' (Predefined Filters). This is the 'pre-defined' CIM_IndicationFilter instance for changes in the OperationalStatus of LogicalDisk instances.
9.8.25 CIM_IndicationFilter (Storage Pool Creation)	Conditional	Experimental. Conditional requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='3' (Predefined Filters). This is the 'pre-defined' CIM_IndicationFilter instance for the addition of a new StoragePool instance.

Table 152 - CIM Elements for CKD Block Services

Element Name	Requirement	Description
9.8.26 CIM_IndicationFilter (Storage Pool Deletion)	Conditional	Experimental. Conditional requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='3' (Predefined Filters). This is the 'pre-defined' CIM_IndicationFilter instance for the deletion of a StoragePool instance.
9.8.27 CIM_IndicationFilter (Storage Pool TotalManagedSpace)	Conditional	Experimental. Conditional requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='3' (Predefined Filters). This is the 'pre-defined' CIM_IndicationFilter instance for changes in TotalManagedSpace for StoragePool instances.
9.8.28 CIM_IndicationFilter (Storage Volume Creation)	Conditional	Experimental. Conditional requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='3' (Predefined Filters). This is the 'pre-defined' CIM_IndicationFilter instance for the addition of a new StorageVolume instance.
9.8.29 CIM_IndicationFilter (Storage Volume Deletion)	Conditional	Experimental. Conditional requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='3' (Predefined Filters). This is the 'pre-defined' CIM_IndicationFilter instance for the deletion of a StorageVolume instance.
9.8.30 CIM_IndicationFilter (Storage Volume OperationalStatus)	Conditional	Experimental. Conditional requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='3' (Predefined Filters). This is the 'pre-defined' CIM_IndicationFilter instance for changes in the OperationalStatus of StorageVolume instances.
9.8.31 CIM_IndicationFilter (WQL Logical Disk OperationalStatus)	Conditional	Deprecated. Conditional requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='3' (Predefined Filters). This is the 'pre-defined' WQL version of the CIM_IndicationFilter instance for changes in the OperationalStatus of LogicalDisk instances.
9.8.32 CIM_IndicationFilter (WQL Storage Volume OperationalStatus)	Conditional	Deprecated. Conditional requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='3' (Predefined Filters). This is the 'pre-defined' WQL version of the CIM_IndicationFilter instance for changes in the OperationalStatus of StorageVolume instances.

Table 152 - CIM Elements for CKD Block Services

Element Name	Requirement	Description
9.8.33 CIM_LogicalDisk	Conditional	Conditional requirement: Referenced from Volume Management - LogicalDisk is mandatory. A LogicalDisk is allocated from a concrete StoragePool. This is required if the parent profile supports LogicalDisks.
9.8.34 CIM_MemberOfCollection (Block Services Filter Collection to FilterCollection)	Conditional	Experimental. Conditional requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='5' (Predefined Filter Collections). This associates the Block Services predefined FilterCollection to the FilterCollection for the autonomous profile (e.g., the Array FilterCollection).
9.8.35 CIM_MemberOfCollection (Predefined Filter Collection to Block Services Filters)	Conditional	Experimental. Conditional requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='5' (Predefined Filter Collections). This associates the Block Services predefined FilterCollection to the predefined Filters supported by the implementation.
9.8.36 CIM_OwningJobElement	Conditional	Conditional requirement: Support for Job Control profile.
9.8.37 CIM_StorageConfigurationCapabilities	Optional	These Capabilities define the capabilities provided by the CIM_StorageConfigurationService.
9.8.38 CIM_StorageConfigurationCapabilities (Concrete)	Optional	
9.8.39 CIM_StorageConfigurationCapabilities (Global)	Conditional	Conditional requirement: Support for StorageConfigurationService.
9.8.40 CIM_StorageConfigurationCapabilities (Primordial)	Optional	
9.8.41 CIM_StorageConfigurationService	Optional	This service provides method for volume and pool manipulation. There are no enhancements for CKD storage.
9.8.42 CIM_StoragePool	Mandatory	Primordial and Concrete Pools. These are unchanged for CKD storage.
9.8.43 CIM_StoragePool (Concrete)	Mandatory	The concrete StoragePool. A concrete StoragePool shall be allocated from another StoragePool. It shall be used for allocating StorageVolumes and LogicalDisks as well as other concrete StoragePools.
9.8.44 CIM_StoragePool (Empty)	Optional	An empty StoragePool is a special case of a StoragePool (Concrete or Primordial) where the StoragePool contains no capacity.
9.8.45 CIM_StoragePool (Primordial)	Mandatory	The primordial StoragePool. It is created by the provider and cannot be deleted or modified. It cannot be used to allocate any storage element other than concrete StoragePools.
9.8.46 CIM_StorageSettingWithHints	Optional	These are hints that can be added to StorageSetting. There are no enhancements for CKD.
9.8.47 CIM_StorageSettingsAssociatedToCapabilities	Optional	This class associates the StorageCapabilities with the pre-defined setting. There are no enhancements for CKD.

Table 152 - CIM Elements for CKD Block Services

Element Name	Requirement	Description
9.8.48 CIM_StorageSettingsGeneratedFromCapabilities	Optional	This class associates the StorageCapabilities with the StorageSetting generated from it via the CreateSetting method. There are no enhancements for CKD.
9.8.49 CIM_SystemDevice (System to StorageVolume or LogicalDisk)	Mandatory	Associates a top level system to the StorageVolumes or LogicalDisks.
9.8.50 SNIA_StorageCapabilities	Mandatory	These Capabilities define the capabilities provided by a CIM_StoragePool. This includes the capability to support SCSI and/or CKD storage.
9.8.51 SNIA_StorageSetting	Mandatory	The SNIA_StorageSettings define the settings for a given StorageVolume (or LogicalDisk). This includes the Setting for whether or not the volume is SCSI or CKD.
9.8.52 SNIA_StorageVolume	Conditional	Conditional requirement: Referenced from either Array or Storage Virtualizer - StorageVolume is mandatory. A logical unit representing a virtual disk. A StorageVolume is allocated from a concrete StoragePool. The StorageVolume is enhanced for CKD.
9.8.53 SNIA_StorageVolume	Optional	An optional extension of CIM_StorageVolume.
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_StoragePool	Mandatory	Creation/Deletion of StoragePool. See 6.8.23 CIM_IndicationFilter (Storage Pool Creation).
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_StoragePool	Mandatory	Deletion of StoragePool. See 6.8.24 CIM_IndicationFilter (Storage Pool Deletion).
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_StorageVolume	Conditional	Conditional requirement: Referenced from Array - StorageVolume is mandatory or Referenced from Storage Virtualizer - StorageVolume is mandatory or Referenced from Host Hardware RAID Controller - StorageVolume is mandatory. Creation of StorageVolume, if the StorageVolume storage element is implemented. See 6.8.26 CIM_IndicationFilter (Storage Volume Creation).
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_StorageVolume	Conditional	Conditional requirement: Referenced from Array - StorageVolume is mandatory or Referenced from Storage Virtualizer - StorageVolume is mandatory or Referenced from Host Hardware RAID Controller - StorageVolume is mandatory. Deletion of StorageVolume, if the StorageVolume storage element is implemented. See 6.8.27 CIM_IndicationFilter (Storage Volume Deletion).
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_StorageVolume AND SourceInstance.OperationalStatus <> PreviousInstance.OperationalStatus	Conditional	Conditional requirement: Referenced from Array - StorageVolume is mandatory or Referenced from Storage Virtualizer - StorageVolume is mandatory or Referenced from Host Hardware RAID Controller - StorageVolume is mandatory. Deprecated WQL - Change of status of a Storage Volume, if Storage Volume is implemented. See 6.8.30 CIM_IndicationFilter (WQL Storage Volume OperationalStatus).

Table 152 - CIM Elements for CKD Block Services

Element Name	Requirement	Description
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_StorageVolume AND SourceInstance.CIM_StorageVolume::OperationalStatus <> PreviousInstance.CIM_StorageVolume::OperationalStatus	Conditional	Conditional requirement: Referenced from Array - StorageVolume is mandatory or Referenced from Storage Virtualizer - StorageVolume is mandatory or Referenced from Host Hardware RAID Controller - StorageVolume is mandatory. CQL - Change of status of a Storage Volume, if Storage Volume is implemented. See 6.8.28 CIM_IndicationFilter (Storage Volume OperationalStatus).
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_LogicalDisk	Conditional	Conditional requirement: Referenced from Volume Management - LogicalDisk is mandatory. Creation of LogicalDisk, if the LogicalDisk storage element is implemented. See 6.8.20 CIM_IndicationFilter (Logical Disk Creation).
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_LogicalDisk	Conditional	Conditional requirement: Referenced from Volume Management - LogicalDisk is mandatory. Deletion of LogicalDisk, if the LogicalDisk storage element is implemented. See 6.8.21 CIM_IndicationFilter (Logical Disk Deletion).
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_LogicalDisk AND SourceInstance.OperationalStatus <> PreviousInstance.OperationalStatus	Conditional	Conditional requirement: Referenced from Volume Management - LogicalDisk is mandatory. Deprecated WQL - Change of status of LogicalDisk, if LogicalDisk is implemented. See 6.8.29 CIM_IndicationFilter (WQL Logical Disk OperationalStatus).
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_LogicalDisk AND SourceInstance.CIM_LogicalDisk::OperationalStatus <> PreviousInstance.CIM_LogicalDisk::OperationalStatus	Conditional	Conditional requirement: Referenced from Volume Management - LogicalDisk is mandatory. CQL - Change of status of LogicalDisk, if LogicalDisk is implemented. See 6.8.22 CIM_IndicationFilter (Logical Disk OperationalStatus).
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_StoragePool AND SourceInstance.CIM_StoragePool::TotalManagedSpace <> PreviousInstance.CIM_StoragePool::TotalManagedSpace	Mandatory	CQL - Change of TotalManagedSpace. See 6.8.25 CIM_IndicationFilter (Storage Pool TotalManagedSpace).
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_StoragePool	Mandatory	Creation/Deletion of StoragePool.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_StoragePool	Mandatory	Deletion of StoragePool.
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_StorageVolume	Conditional	Conditional requirement: Referenced from either Array or Storage Virtualizer - StorageVolume is mandatory. Creation of StorageVolume, if the StorageVolume storage element is implemented.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_StorageVolume	Conditional	Conditional requirement: Referenced from either Array or Storage Virtualizer - StorageVolume is mandatory. Deletion of StorageVolume, if the StorageVolume storage element is implemented.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_StorageVolume AND SourceInstance.OperationalStatus <> PreviousInstance.OperationalStatus	Conditional	Conditional requirement: Referenced from either Array or Storage Virtualizer - StorageVolume is mandatory. Deprecated WQL - Change of status of a Storage Volume, if Storage Volume is implemented.

Table 152 - CIM Elements for CKD Block Services

Element Name	Requirement	Description
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_StorageVolume AND SourceInstance.CIM_StorageVolume::OperationalStatus <> PreviousInstance.CIM_StorageVolume::OperationalStatus	Conditional	Conditional requirement: Referenced from either Array or Storage Virtualizer - StorageVolume is mandatory. CQL - Change of status of a Storage Volume, if Storage Volume is implemented.
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_LogicalDisk	Conditional	Conditional requirement: Referenced from Volume Management - LogicalDisk is mandatory. Creation of LogicalDisk, if the LogicalDisk storage element is implemented.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_LogicalDisk	Conditional	Conditional requirement: Referenced from Volume Management - LogicalDisk is mandatory. Deletion of LogicalDisk, if the LogicalDisk storage element is implemented.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_LogicalDisk AND SourceInstance.OperationalStatus <> PreviousInstance.OperationalStatus	Conditional	Conditional requirement: Referenced from Volume Management - LogicalDisk is mandatory. Depreciated WQL - Change of status of LogicalDisk, if LogicalDisk is implemented.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_LogicalDisk AND SourceInstance.CIM_LogicalDisk::OperationalStatus <> PreviousInstance.CIM_LogicalDisk::OperationalStatus	Conditional	Conditional requirement: Referenced from Volume Management - LogicalDisk is mandatory. CQL - Change of status of LogicalDisk, if LogicalDisk is implemented.

9.8.2 CIM_AllocatedFromStoragePool

This class is unchanged from the Block Services Package.

Requirement: Mandatory.

9.8.3 CIM_AllocatedFromStoragePool (Pool from Pool)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 153 describes class CIM_AllocatedFromStoragePool (Pool from Pool).

Table 153 - SMI Referenced Properties/Methods for CIM_AllocatedFromStoragePool (Pool from Pool)

Properties	Flags	Requirement	Description and Notes
SpaceConsumed		Mandatory	
Antecedent		Mandatory	Antecedent references the parent pool from which the dependent pool is allocated.
Dependent		Mandatory	

9.8.4 CIM_AllocatedFromStoragePool (Volume or LogicalDisk from Pool)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Referenced from Array - StorageVolume is mandatory or Referenced from Storage Virtualizer - StorageVolume is mandatory or Referenced from Host Hardware RAID Controller - StorageVolume is mandatory.

Table 154 describes class CIM_AllocatedFromStoragePool (Volume or LogicalDisk from Pool).

Table 154 - SMI Referenced Properties/Methods for CIM_AllocatedFromStoragePool (Volume or LogicalDisk from Pool)

Properties	Flags	Requirement	Description and Notes
SpaceConsumed		Mandatory	
Antecedent		Mandatory	
Dependent		Mandatory	

9.8.5 CIM_ElementCapabilities

This class is unchanged from the Block Services Package.

Requirement: Mandatory.

9.8.6 CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StorageVolume or LogicalDisk)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 155 describes class CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StorageVolume or LogicalDisk).

Table 155 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StorageVolume or LogicalDisk)

Properties	Flags	Requirement	Description and Notes
Capabilities		Mandatory	The capabilities object associated with the element.
ManagedElement		Mandatory	A Storage Volume or Logical Disk.

9.8.7 CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StoragePool)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 156 describes class CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StoragePool).

Table 156 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StoragePool)

Properties	Flags	Requirement	Description and Notes
Capabilities		Mandatory	The capabilities object (CIM_EnabledLogicalElementCapabilities) with an ElementName of "StoragePool Enabled Capabilities" that is associated with a storage pool.
ManagedElement		Mandatory	A reference to an instance of a StoragePool.

9.8.8 CIM_ElementCapabilities (StorageCapabilities to StorageConfigurationService)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 157 describes class CIM_ElementCapabilities (StorageCapabilities to StorageConfigurationService).

Table 157 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (StorageCapabilities to StorageConfigurationService)

Properties	Flags	Requirement	Description and Notes
Capabilities		Mandatory	The capabilities object associated with the element.
ManagedElement		Mandatory	The managed element.

9.8.9 CIM_ElementCapabilities (StorageCapabilities to StoragePool)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 158 describes class CIM_ElementCapabilities (StorageCapabilities to StoragePool).

Table 158 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (StorageCapabilities to StoragePool)

Properties	Flags	Requirement	Description and Notes
Capabilities		Mandatory	The capabilities object associated with the element.
ManagedElement		Mandatory	The managed element.

9.8.10 CIM_ElementCapabilities (StorageConfigurationCapabilities to StorageConfigurationService)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 159 describes class CIM_ElementCapabilities (StorageConfigurationCapabilities to StorageConfigurationService).

Table 159 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (StorageConfigurationCapabilities to StorageConfigurationService)

Properties	Flags	Requirement	Description and Notes
Capabilities		Mandatory	The capabilities object associated with the element.
ManagedElement		Mandatory	The managed element.

9.8.11 CIM_ElementCapabilities (StorageConfigurationCapabilities to concrete StoragePool)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 160 describes class CIM_ElementCapabilities (StorageConfigurationCapabilities to concrete StoragePool).

Table 160 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (StorageConfigurationCapabilities to concrete StoragePool)

Properties	Flags	Requirement	Description and Notes
Capabilities		Mandatory	The capabilities object associated with the element.
ManagedElement		Mandatory	The managed element.

9.8.12 CIM_ElementCapabilities (StorageConfigurationCapabilities to primordial StoragePool)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 161 describes class CIM_ElementCapabilities (StorageConfigurationCapabilities to primordial StoragePool).

Table 161 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (StorageConfigurationCapabilities to primordial StoragePool)

Properties	Flags	Requirement	Description and Notes
Capabilities		Mandatory	The capabilities object associated with the element.
ManagedElement		Mandatory	The managed element.

9.8.13 CIM_ElementCapabilities (Used to declare the naming capabilities of the StoragePool)

Associates EnabledLogicalElementCapabilities with StorageConfigurationService. This is for identifying the capability to provide an element name for storage pools.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 162 describes class CIM_ElementCapabilities (Used to declare the naming capabilities of the StoragePool).

**Table 162 - SMI Referenced Properties/Methods for CIM_ElementCapabilities
(Used to declare the naming capabilities of the StoragePool)**

Properties	Flags	Requirement	Description and Notes
Capabilities		Mandatory	The capabilities object (CIM_EnabledLogicalElementCapabilities) with an ElementName of "StoragePool Enabled Capabilities" that is associated with an instance of StorageConfigurationService.
ManagedElement		Mandatory	A reference to an instance of CIM_StorageConfigurationService.

9.8.14 CIM_ElementCapabilities (Used to declare the naming capabilities of the StorageVolume or LogicalDisk)

Associates EnabledLogicalElementCapabilities with StorageConfigurationService. This is for identifying the capability to provide an element name for storage volumes or logical disks.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 163 describes class CIM_ElementCapabilities (Used to declare the naming capabilities of the StorageVolume or LogicalDisk).

**Table 163 - SMI Referenced Properties/Methods for CIM_ElementCapabilities
(Used to declare the naming capabilities of the StorageVolume or LogicalDisk)**

Properties	Flags	Requirement	Description and Notes
Capabilities		Mandatory	The capabilities object (CIM_EnabledLogicalElementCapabilities) with an ElementName of "StorageVolume Enabled Capabilities" or "LogicalDisk Enabled Capabilities" that is associated with an instance of StorageConfigurationService.
ManagedElement		Mandatory	A reference to an instance of CIM_StorageConfigurationService.

9.8.15 CIM_ElementSettingData

This class is unchanged from the Block Services Package.

Requirement: Mandatory.

9.8.16 CIM_EnabledLogicalElementCapabilities (For StorageConfigurationService)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 164 describes class CIM_EnabledLogicalElementCapabilities (For StorageConfigurationService).

Table 164 - SMI Referenced Properties/Methods for CIM_EnabledLogicalElementCapabilities (For StorageConfigurationService)

Properties	Flags	Requirement	Description and Notes
InstanceID		Mandatory	
ElementName		Mandatory	For this usage of the capabilities this should include one of the following three values: StoragePool Enabled Capabilities StorageVolume Enabled Capabilities LogicalDisk Enabled Capabilities.
ElementNameEditSupported		Mandatory	Denotes whether a storage element can be named.
MaxElementNameLen		Mandatory	Specifies the maximum length in glyphs (letters) for the name. See MOF for details.
ElementNameMask		Mandatory	The regular expression that specifies the possible content and format for the element name. See MOF (Annex A) for details.
RequestedStatesSupported		Optional	Expresses the states to which this element may be changed using the RequestStateChange method. If this property is NULL, it may be assumed that the state may not be changed.

9.8.17 CIM_EnabledLogicalElementCapabilities (For StoragePool)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 165 describes class CIM_EnabledLogicalElementCapabilities (For StoragePool).

Table 165 - SMI Referenced Properties/Methods for CIM_EnabledLogicalElementCapabilities (For StoragePool)

Properties	Flags	Requirement	Description and Notes
InstanceID		Mandatory	
ElementName		Mandatory	For this usage of the capabilities this should be 'StoragePool Enabled Capabilities'.
ElementNameEditSupported		Mandatory	Denotes whether a storage element can be named.
MaxElementNameLen		Mandatory	Specifies the maximum length in glyphs (letters) for the name. See MOF for details.
ElementNameMask		Mandatory	The regular expression that specifies the possible content and format for the element name. See MOF for details.
RequestedStatesSupported		Optional	Expresses the states to which this element may be changed using the RequestStateChange method. If this property is NULL, it may be assumed that the state may not be changed.

9.8.18 CIM_FilterCollection (Block Services Predefined FilterCollection)

Experimental. This is a collection of predefined IndicationFilters to which a client may subscribe. A Block Services implementation shall indicate support for predefined FilterCollections by the SNIA_IndicationConfigurationCapabilities.FeaturesSupported = '5' (Predefined Filter Collections).

Requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='5' (Predefined Filter Collections).

Table 166 describes class CIM_FilterCollection (Block Services Predefined FilterCollection).

**Table 166 - SMI Referenced Properties/Methods for CIM_FilterCollection
(Block Services Predefined FilterCollection)**

Properties	Flags	Requirement	Description and Notes
InstanceID		Mandatory	Shall specify the unique identifier for an instance of this class within the Implementation namespace.
CollectionName		Mandatory	The value of CollectionName shall be 'SNIA:Block Services'.

9.8.19 CIM_HostedCollection (System to predefined IndicationFilters)

Experimental.

Requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='5' (Predefined Filter Collections).

Table 167 describes class CIM_HostedCollection (System to predefined IndicationFilters).

**Table 167 - SMI Referenced Properties/Methods for CIM_HostedCollection
(System to predefined IndicationFilters)**

Properties	Flags	Requirement	Description and Notes
Dependent		Mandatory	Reference to the predefined FilterCollection for Block Services.
Antecedent		Mandatory	Reference to the System of the referencing profile.

9.8.20 CIM_HostedService

This class is unchanged from the Block Services Package.

Requirement: Optional.

9.8.21 CIM_HostedStoragePool

Requirement: Mandatory.

Table 168 describes class CIM_HostedStoragePool.

Table 168 - SMI Referenced Properties/Methods for CIM_HostedStoragePool

Properties	Flags	Requirement	Description and Notes
GroupComponent		Mandatory	The reference to the hosting computer system.
PartComponent		Mandatory	The reference to the hosted storage pool.

9.8.22 CIM_IndicationFilter (Logical Disk Creation)

Experimental. This is the 'pre-defined' CIM_IndicationFilter instance for the addition of a new LogicalDisk instance. This would typically occur as a result of an invocation of CreateOrModifyElementFromStoragePool method.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='3' (Predefined Filters).

Table 169 describes class CIM_IndicationFilter (Logical Disk Creation).

Table 169 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Logical Disk Creation)

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
CreationClassName		Mandatory	See the CreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
SystemName		Mandatory	See the SystemName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
Name		Mandatory	This shall be 'SNIA:Block Services:LogicalDiskCreation'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
Query		Mandatory	SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_LogicalDisk.
QueryLanguage		Mandatory	This should be 'DMTF:CQL' for CQL queries, but may be 'WQL' or 'SMI-S V1.0'. WQL and SMI-S V1.0 are deprecated in favor of 'DMTF:CQL'.
ElementName	N	Optional	See the ElementName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).

9.8.23 CIM_IndicationFilter (Logical Disk Deletion)

Experimental. This is the 'pre-defined' CIM_IndicationFilter instance for the deletion of a LogicalDisk instance. This would typically occur as a result of an invocation of ReturnToStoragePool method.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='3' (Predefined Filters).

Table 170 describes class CIM_IndicationFilter (Logical Disk Deletion).

Table 170 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Logical Disk Deletion)

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
CreationClassName		Mandatory	See the CreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
SystemName		Mandatory	See the SystemName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
Name		Mandatory	This shall be 'SNIA:Block Services:LogicalDiskDeletion'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
Query		Mandatory	SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_LogicalDisk.
QueryLanguage		Mandatory	This should be 'DMTF:CQL' for CQL queries, but may be 'WQL' or 'SMI-S V1.0'. WQL and SMI-S V1.0 are deprecated in favor of 'DMTF:CQL'.
ElementName	N	Optional	See the ElementName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).

9.8.24 CIM_IndicationFilter (Logical Disk OperationalStatus)

Experimental. This is the 'pre-defined' CIM_IndicationFilter instance for changes in the OperationalStatus of LogicalDisk instances.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='3' (Predefined Filters).

Table 171 describes class CIM_IndicationFilter (Logical Disk OperationalStatus).

Table 171 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Logical Disk OperationalStatus)

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
CreationClassName		Mandatory	See the CreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
SystemName		Mandatory	See the SystemName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
Name		Mandatory	This shall be 'SNIA:Block Services:LogicalDiskOperationalStatus'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
Query		Mandatory	SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_LogicalDisk AND SourceInstance.CIM_LogicalDisk::OperationalStatus <> PreviousInstance.CIM_LogicalDisk::OperationalStatus.
QueryLanguage		Mandatory	This shall be 'DMTF:CQL'.
ElementName	N	Optional	See the ElementName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).

9.8.25 CIM_IndicationFilter (Storage Pool Creation)

Experimental. This is the 'pre-defined' CIM_IndicationFilter instance for the addition of a new StoragePool instance. This would typically occur as a result of an invocation of CreateOrModifyStoragePool method.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='3' (Predefined Filters).

Table 172 describes class CIM_IndicationFilter (Storage Pool Creation).

Table 172 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Storage Pool Creation)

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
CreationClassName		Mandatory	See the CreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
SystemName		Mandatory	See the SystemName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
Name		Mandatory	This shall be 'SNIA:Block Services:StoragePoolCreation'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
Query		Mandatory	SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_StoragePool.
QueryLanguage		Mandatory	This should be 'DMTF:CQL' for CQL queries, but may be 'WQL' or 'SMI-S V1.0'. WQL and SMI-S V1.0 are deprecated in favor of 'DMTF:CQL'.
ElementName	N	Optional	See the ElementName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).

9.8.26 CIM_IndicationFilter (Storage Pool Deletion)

Experimental. This is the 'pre-defined' CIM_IndicationFilter instance for the deletion of a StoragePool instance. This would typically occur as a result of an invocation of DeleteStoragePool method.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='3' (Predefined Filters).

Table 173 describes class CIM_IndicationFilter (Storage Pool Deletion).

Table 173 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Storage Pool Deletion)

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
CreationClassName		Mandatory	See the CreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
SystemName		Mandatory	See the SystemName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
Name		Mandatory	This shall be 'SNIA:Block Services:StoragePoolDeletion'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
Query		Mandatory	SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_StoragePool.
QueryLanguage		Mandatory	This should be 'DMTF:CQL' for CQL queries, but may be 'WQL' or 'SMI-S V1.0'. WQL and SMI-S V1.0 are deprecated in favor of 'DMTF:CQL'.
ElementName	N	Optional	See the ElementName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).

9.8.27 CIM_IndicationFilter (Storage Pool TotalManagedSpace)

Experimental. This is the 'pre-defined' CIM_IndicationFilter instance for changes in TotalManagedSpace for StoragePool instances. This would typically occur as a result of an invocation of CreateOrModifyStoragePool that expands a StoragePool.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='3' (Predefined Filters).

Table 174 describes class CIM_IndicationFilter (Storage Pool TotalManagedSpace).

**Table 174 - SMI Referenced Properties/Methods for CIM_IndicationFilter
(Storage Pool TotalManagedSpace)**

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
CreationClassName		Mandatory	See the CreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
SystemName		Mandatory	See the SystemName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
Name		Mandatory	This shall be 'SNIA:Block Services:StoragePoolTotalManagedSpace'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
Query		Mandatory	SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_StoragePool AND SourceInstance.CIM_StoragePool::TotalManagedSpace <> PreviousInstance.CIM_StoragePool::TotalManagedSpace.
QueryLanguage		Mandatory	This shall be 'DMTF:CQL'.
ElementName	N	Optional	See the ElementName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).

9.8.28 CIM_IndicationFilter (Storage Volume Creation)

Experimental. This is the 'pre-defined' CIM_IndicationFilter instance for the addition of a new StorageVolume instance. This would typically occur as a result of an invocation of CreateOrModifyElementFromStoragePool method.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='3' (Predefined Filters).

Table 175 describes class CIM_IndicationFilter (Storage Volume Creation).

**Table 175 - SMI Referenced Properties/Methods for CIM_IndicationFilter
(Storage Volume Creation)**

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
CreationClassName		Mandatory	See the CreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
SystemName		Mandatory	See the SystemName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
Name		Mandatory	This shall be 'SNIA:Block Services:StorageVolumeCreation'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
Query		Mandatory	SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_StorageVolume.
QueryLanguage		Mandatory	This should be 'DMTF:CQL' for CQL queries, but may be 'WQL' or 'SMI-S V1.0'. WQL and SMI-S V1.0 are deprecated in favor of 'DMTF:CQL'.
ElementName	N	Optional	See the ElementName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).

9.8.29 CIM_IndicationFilter (Storage Volume Deletion)

Experimental. This is the 'pre-defined' CIM_IndicationFilter instance for the deletion of a StorageVolume instance. This would typically occur as a result of an invocation of ReturnToStoragePool method.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='3' (Predefined Filters).

Table 176 describes class CIM_IndicationFilter (Storage Volume Deletion).

**Table 176 - SMI Referenced Properties/Methods for CIM_IndicationFilter
(Storage Volume Deletion)**

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
CreationClassName		Mandatory	See the CreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
SystemName		Mandatory	See the SystemName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
Name		Mandatory	This shall be 'SNIA:Block Services:StorageVolumeDeletion'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
Query		Mandatory	SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_StorageVolume.
QueryLanguage		Mandatory	This should be 'DMTF:CQL' for CQL queries, but may be 'WQL' or 'SMI-S V1.0'. WQL and SMI-S V1.0 are deprecated in favor of 'DMTF:CQL'.
ElementName	N	Optional	See the ElementName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).

9.8.30 CIM_IndicationFilter (Storage Volume OperationalStatus)

Experimental. This is the 'pre-defined' CIM_IndicationFilter instance for changes in the OperationalStatus of StorageVolume instances.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='3' (Predefined Filters).

Table 177 describes class CIM_IndicationFilter (Storage Volume OperationalStatus).

Table 177 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Storage Volume OperationalStatus)

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
CreationClassName		Mandatory	See the CreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
SystemName		Mandatory	See the SystemName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
Name		Mandatory	This shall be 'SNIA:Block Services:StorageVolumeOperationalStatus'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
Query		Mandatory	SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_StorageVolume AND SourceInstance.CIM_StorageVolume::OperationalStatus <> PreviousInstance.CIM_StorageVolume::OperationalStatus .
QueryLanguage		Mandatory	This shall be 'DMTF:CQL'.
ElementName	N	Optional	See the ElementName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).

9.8.31 CIM_IndicationFilter (WQL Logical Disk OperationalStatus)

Deprecated. This is the 'pre-defined' WQL version of the CIM_IndicationFilter instance for changes in the OperationalStatus of LogicalDisk instances.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='3' (Predefined Filters).

Table 178 describes class CIM_IndicationFilter (WQL Logical Disk OperationalStatus).

**Table 178 - SMI Referenced Properties/Methods for CIM_IndicationFilter
(WQL Logical Disk OperationalStatus)**

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
CreationClassName		Mandatory	See the CreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
SystemName		Mandatory	See the SystemName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
Name		Mandatory	This shall be 'SNIA:Block Services:LogicalDiskOperationalStatusWQL'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
Query		Mandatory	SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_LogicalDisk AND SourceInstance.OperationalStatus <> PreviousInstance.OperationalStatus.
QueryLanguage		Mandatory	This shall be 'WQL' or 'SMI-S V1.0'.
ElementName	N	Optional	See the ElementName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).

9.8.32 CIM_IndicationFilter (WQL Storage Volume OperationalStatus)

Deprecated. This is the 'pre-defined' WQL version of the CIM_IndicationFilter instance for changes in the OperationalStatus of StorageVolume instances.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='3' (Predefined Filters).

Table 179 describes class CIM_IndicationFilter (WQL Storage Volume OperationalStatus).

**Table 179 - SMI Referenced Properties/Methods for CIM_IndicationFilter
(WQL Storage Volume OperationalStatus)**

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
CreationClassName		Mandatory	See the CreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
SystemName		Mandatory	See the SystemName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
Name		Mandatory	This shall be 'SNIA:Block Services:StorageVolumeOperationalStatusWQL'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
Query		Mandatory	SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_StorageVolume AND SourceInstance.OperationalStatus <> PreviousInstance.OperationalStatus.
QueryLanguage		Mandatory	This shall be 'WQL' or 'SMI-S V1.0'.
ElementName	N	Optional	See the ElementName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).

9.8.33 CIM_LogicalDisk

LogicalDisks could be formatted as CKD disks. The Properties that are different from what is specified in the Block Services Package have descriptive text. Properties that are unchanged from the Block Services Package are omitted from the table.

Created By: Static

Modified By: Static

Deleted By: Extrinsic: StorageConfigurationService.ReturnToStoragePool

Requirement: Referenced from Volume Management - LogicalDisk is mandatory.

Table 180 describes class CIM_LogicalDisk.

Table 180 - SMI Referenced Properties/Methods for CIM_LogicalDisk

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	Opaque identifier.
ElementName		Optional	User-friendly name.
Name		Mandatory	OS Device Name.

Table 180 - SMI Referenced Properties/Methods for CIM_LogicalDisk

Properties	Flags	Requirement	Description and Notes
NameFormat		Mandatory	This shall be "12" (OS Device Name).
ExtentStatus		Mandatory	
OperationalStatus		Mandatory	Value shall be 2 3 6 8 15 (OK or Degraded or Error or Starting or Dormant).
BlockSize (overridden)		Mandatory	The BlockSize would report the number of bytes in a cylinder.
NumberOfBlocks (overridden)		Mandatory	The number of blocks would be the number of cylinders.
ConsumableBlocks (overridden)		Mandatory	The number of usable cylinders.
IsBasedOnUnderlyingRedundancy		Mandatory	
NoSinglePointOfFailure		Mandatory	
DataRedundancy		Mandatory	
PackageRedundancy		Mandatory	
DeltaReservation		Mandatory	
Usage		Optional	The specialized usage intended for this element.
OtherUsageDescription		Optional	Set when Usage value is "Other".
ClientSettableUsage		Optional	Lists Usage values that can be set by a client for this element.
Primordial		Mandatory	Shall be false.
ExtentDiscriminator		Mandatory	Experimental. This is an array of values that shall contain 'SNIA:Allocated'.
DataOrganization (added)		Mandatory	Supported value for SMI-S is "4" (Count Key Data). Values that are not "4" are for non-CKD LogicalDisks. CKD LogicalDisks use "4".

9.8.34 CIM_MemberOfCollection (Block Services Filter Collection to FilterCollection)

Experimental. This associates the Block Services predefined FilterCollection to the FilterCollection for the autonomous profile (e.g., the Array FilterCollection).

Requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='5' (Predefined Filter Collections).

Table 181 describes class CIM_MemberOfCollection (Block Services Filter Collection to FilterCollection).

Table 181 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Block Services Filter Collection to FilterCollection)

Properties	Flags	Requirement	Description and Notes
Collection		Mandatory	Reference to the Block Services predefined FilterCollection.
Member		Mandatory	Reference to the Block Services predefined FilterCollection.

9.8.35 CIM_MemberOfCollection (Predefined Filter Collection to Block Services Filters)

Experimental. This associates the Block Services predefined FilterCollection to the predefined Filters supported by the implementation.

Requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='5' (Predefined Filter Collections).

Table 182 describes class CIM_MemberOfCollection (Predefined Filter Collection to Block Services Filters).

Table 182 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Predefined Filter Collection to Block Services Filters)

Properties	Flags	Requirement	Description and Notes
Collection		Mandatory	Reference to the Block Services predefined FilterCollection.
Member		Mandatory	Reference to the predefined IndicationFilters of the Block Services implementation.

9.8.36 CIM_OwningJobElement

Conditional on support for Job Control profile.

Requirement: Support for Job Control profile.

Table 183 describes class CIM_OwningJobElement.

Table 183 - SMI Referenced Properties/Methods for CIM_OwningJobElement

Properties	Flags	Requirement	Description and Notes
OwnedElement		Mandatory	
OwningElement		Mandatory	

9.8.37 CIM_StorageConfigurationCapabilities

The Properties that are different from what is specified in the Block Services Package have descriptive text. Properties that are unchanged from the Block Services Package are omitted from the tables.

Created By: Static

Requirement: Optional.

9.8.38 CIM_StorageConfigurationCapabilities (Concrete)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 184 describes class CIM_StorageConfigurationCapabilities (Concrete).

Table 184 - SMI Referenced Properties/Methods for CIM_StorageConfigurationCapabilities (Concrete)

Properties	Flags	Requirement	Description and Notes
InstanceID		Mandatory	
ElementName		Mandatory	
SupportedStoragePoolFeatures		Optional	Lists what StorageConfigurationService functionalities are implemented. Matches 2 3 5 6 7 (InExtents or Single InPool or Storage Pool QoS Change or Storage Pool Capacity Expansion or Storage Pool Capacity Reduction).
SupportedSynchronousActions		Optional	Lists what actions, invoked through StorageConfigurationService methods, shall not produce Concrete jobs. This International Standard recognizes "2" (Storage Pool Creation), "4" (Storage Pool Modification), "5" (Storage Element Creation), "12" (Storage Element from Element Creation), "13" (Storage Element from Element Modification) or "15" (StoragePool Usage Modification).
SupportedStorageElementTypes		Mandatory	Lists the type of storage elements that are supported by this implementation. This International Standard recognizes '2' (StorageVolume) or '4' (LogicalDisk).
SupportedAsynchronousActions		Optional	Lists what actions, invoked through StorageConfigurationService methods, may produce Concrete jobs. This International Standard recognizes "2" (Storage Pool Creation), "4" (Storage Pool Modification), "5" (Storage Element Creation), "12" (Storage Element from Element Creation), "13" (Storage Element from Element Modification) or "15" (StoragePool Usage Modification).
SupportedStorageElementFeatures		Optional	Lists actions supported through the invocation of StorageServiceService.CreateOrModifyElementFromStoragePool(). Matches 3 8 (StorageVolume Creation or LogicalDisk Creation).
SupportedStorageElementUsage		Optional	Indicates the intended usage or any restrictions that may have been imposed on supported storage elements.
ClientSettableElementUsage		Optional	Indicates the intended usage or any restrictions that may have been imposed on the usage of client-settable elements.
SupportedStoragePoolUsage		Optional	Indicates the intended usage or any restrictions that may have been imposed on storage pools.
ClientSettablePoolUsage		Optional	Indicates the intended usage or any restrictions that may have been imposed on the usage of a client-settable storage pool.

9.8.39 CIM_StorageConfigurationCapabilities (Global)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Support for StorageConfigurationService.

Table 185 describes class CIM_StorageConfigurationCapabilities (Global).

Table 185 - SMI Referenced Properties/Methods for CIM_StorageConfigurationCapabilities (Global)

Properties	Flags	Requirement	Description and Notes
InstanceID		Mandatory	
ElementName		Mandatory	
SupportedStoragePoolFeatures		Optional	Lists what StorageConfigurationService functionalities are implemented. Matches 2 3 5 6 7 (InExtents or Single InPool or Storage Pool QoS Change or Storage Pool Capacity Expansion or Storage Pool Capacity Reduction).
SupportedSynchronousActions		Optional	Lists what actions, invoked through StorageConfigurationService methods, shall not produce Concrete jobs.
SupportedStorageElementTypes		Mandatory	Lists the type of storage elements that are supported by this implementation.
SupportedAsynchronousActions		Optional	Lists what actions, invoked through StorageConfigurationService methods, may produce Concrete jobs.
SupportedStorageElementFeatures		Optional	Lists actions supported through the invocation of StorageServiceService.CreateOrModifyElementFromStoragePool(). Matches 3 5 8 9 11 12 13 (StorageVolume Creation or StorageVolume Modification or LogicalDisk Creation or LogicalDisk Modification or Storage Element QoS Change or Storage Element Capacity Expansion or Storage Element Capacity Reduction).
SupportedStorageElementUsage		Optional	Indicates the intended usage or any restrictions that may have been imposed on supported storage elements.
ClientSettableElementUsage		Optional	Indicates the intended usage or any restrictions that may have been imposed on the usage of client-settable elements.
SupportedStoragePoolUsage		Optional	Indicates the intended usage or any restrictions that may have been imposed on storage pools.
ClientSettablePoolUsage		Optional	Indicates the intended usage or any restrictions that may have been imposed on the usage of a client-settable storage pool.

9.8.40 CIM_StorageConfigurationCapabilities (Primordial)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 186 describes class CIM_StorageConfigurationCapabilities (Primordial).

Table 186 - SMI Referenced Properties/Methods for CIM_StorageConfigurationCapabilities (Primordial)

Properties	Flags	Requirement	Description and Notes
InstanceID		Mandatory	
ElementName		Mandatory	
SupportedStoragePoolFeatures		Optional	Lists what StorageConfigurationService functionalities are implemented. Matches 2 3 (InExtents or Single InPool).

Table 186 - SMI Referenced Properties/Methods for CIM_StorageConfigurationCapabilities (Primordial)

Properties	Flags	Requirement	Description and Notes
SupportedSynchronousActions		Optional	Lists what actions, invoked through StorageConfigurationService methods, shall not produce Concrete jobs. This International Standard recognizes "2" (Storage Pool Creation), "12" (Storage Element from Element Creation) or "15" (StoragePool Usage Modification).
SupportedStorageElementTypes		Optional	Lists the type of storage elements that are supported by this implementation. This International Standard does not recognize any values for this property.
SupportedAsynchronousActions		Optional	Lists what actions, invoked through StorageConfigurationService methods, may produce Concrete jobs. This International Standard recognizes "2" (Storage Pool Creation), "12" (Storage Element from Element Creation) or "15" (StoragePool Usage Modification).
SupportedStorageElementFeatures		Optional	Lists actions supported through the invocation of StorageServiceService.CreateOrModifyElementFromStoragePool(). This International Standard does not recognize any values for this property. For Primordial pools, this shall not contain 3 (StorageVolume Creation), 5 (StorageVolume Modification), 8 (LogicalDisk Creation) or 9 (LogicalDisk Modification).
SupportedStorageElementUsage		Optional	For Primordial StorageConfigurationCapabilities, this shall be NULL.
ClientSettableElementUsage		Optional	For Primordial StorageConfigurationCapabilities, this shall be NULL.
SupportedStoragePoolUsage		Optional	Indicates the intended usage or any restrictions that may have been imposed on storage pools.
ClientSettablePoolUsage		Optional	Indicates the intended usage or any restrictions that may have been imposed on the usage of a client-settable storage pool.

9.8.41 CIM_StorageConfigurationService

This class is unchanged from the Block Services Package. The changes are in the StorageCapabilities and StorageSettings associated with these methods.

Created By: Static

Requirement: Optional.

9.8.42 CIM_StoragePool

This class is unchanged from the Block Services Package. Changes for StoragePools are in the StorageCapabilities associated to the StoragePools.

Requirement: Mandatory.

9.8.43 CIM_StoragePool (Concrete)

Created By: Extrinsic: StorageConfigurationService.CreateOrModifyStoragePool

Modified By: Extrinsic: StorageConfigurationService.CreateOrModifyStoragePool

Deleted By: Extrinsic: StorageConfigurationService.DeleteStoragePool

Requirement: Mandatory.

Table 187 describes class CIM_StoragePool (Concrete).

Table 187 - SMI Referenced Properties/Methods for CIM_StoragePool (Concrete)

Properties	Flags	Requirement	Description and Notes
Primordial		Mandatory	Shall be false.
InstanceID		Mandatory	
ElementName		Optional	
PoolID		Mandatory	A unique name in the context of this system that identifies this Pool.
TotalManagedSpace		Mandatory	
RemainingManagedSpace		Mandatory	
Usage		Optional	The specialized usage intended for this element.
OtherUsageDescription		Optional	Set when Usage value is "Other".
ClientSettableUsage		Optional	Lists Usage values that can be set by a client for this element.
GetSupportedSizes()		Conditional	Conditional requirement: Support for StorageConfigurationService. List the discrete storage element sizes that can be created or expanded from this Pool.
GetSupportedSizeRange()		Conditional	Conditional requirement: Support for StorageConfigurationService. List the size ranges for storage element that can be created or expanded from this Pool.
GetAvailableExtents()		Optional	List the StorageExtents from this Pool that may be used to create or expand a storage element. The StorageExtents may not already be in use as supporting capacity for existing storage element.

9.8.44 CIM_StoragePool (Empty)

An empty StoragePool is a special case of a StoragePool where the StoragePool contains no capacity. All properties are supported as defined for the StoragePool (Concrete or Primordial), except that the empty StoragePool has TotalManagedSpace=0.

Created By: Extrinsic: StorageConfigurationService.CreateOrModifyStoragePool

Modified By: Extrinsic: StorageConfigurationService.CreateOrModifyStoragePool

Deleted By: Extrinsic: StorageConfigurationService.DeleteStoragePool

Requirement: Optional.

Table 188 describes class CIM_StoragePool (Empty).

Table 188 - SMI Referenced Properties/Methods for CIM_StoragePool (Empty)

Properties	Flags	Requirement	Description and Notes
Primordial		Mandatory	This may be either true or false. That is, both concrete and primordial StoragePools may be empty.
InstanceID		Mandatory	
ElementName		Optional	
PoolID		Mandatory	
TotalManagedSpace		Mandatory	This shall be 0 for an empty StoragePool.
RemainingManagedSpace		Mandatory	

Table 188 - SMI Referenced Properties/Methods for CIM_StoragePool (Empty)

Properties	Flags	Requirement	Description and Notes
Usage		Optional	
OtherUsageDescription		Optional	
ClientSettableUsage		Optional	
GetSupportedSizes()		Conditional	Conditional requirement: Support for StorageConfigurationService.
GetSupportedSizeRange()		Conditional	Conditional requirement: Support for StorageConfigurationService.
GetAvailableExtents()		Optional	

9.8.45 CIM_StoragePool (Primordial)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 189 describes class CIM_StoragePool (Primordial).

Table 189 - SMI Referenced Properties/Methods for CIM_StoragePool (Primordial)

Properties	Flags	Requirement	Description and Notes
Primordial		Mandatory	Shall be true.
InstanceID		Mandatory	
ElementName		Optional	
PoolID		Mandatory	A unique name in the context of this system that identifies this Pool.
TotalManagedSpace		Mandatory	
RemainingManagedSpace		Mandatory	
Usage		Optional	The specialized usage intended for this element.
OtherUsageDescription		Optional	Set when Usage value is "Other".
ClientSettableUsage		Optional	Lists Usage values that can be set by a client for this element.
GetSupportedSizes()		Conditional	Conditional requirement: Support for StorageConfigurationService. List the discrete storage element sizes that can be created or expanded from this Pool.
GetSupportedSizeRange()		Conditional	Conditional requirement: Support for StorageConfigurationService. List the size ranges for storage element that can be created or expanded from this Pool.
GetAvailableExtents()		Optional	List the StorageExtents from this Pool that may be used to create or expand a storage element. The StorageExtents may not already be in use as supporting capacity for existing storage element.

9.8.46 CIM_StorageSettingWithHints

This class is unchanged from the Block Services Package.

Requirement: Optional.

9.8.47 CIM_StorageSettingsAssociatedToCapabilities

This class is unchanged from the Block Services Package.

Requirement: Optional.

9.8.48 CIM_StorageSettingsGeneratedFromCapabilities

This class is unchanged from the Block Services Package.

Created By: Extrinsic: CreateSetting

Modified By: ModifyInstance

Deleted By: DeleteInstance

Requirement: Optional.

9.8.49 CIM_SystemDevice (System to StorageVolume or LogicalDisk)

Created By: Static

Modified By: Static

Deleted By: Extrinsic: StorageConfigurationService.ReturnToStoragePool

Requirement: Mandatory.

Table 190 describes class CIM_SystemDevice (System to StorageVolume or LogicalDisk).

**Table 190 - SMI Referenced Properties/Methods for CIM_SystemDevice
(System to StorageVolume or LogicalDisk)**

Properties	Flags	Requirement	Description and Notes
PartComponent		Mandatory	
GroupComponent		Mandatory	

9.8.50 SNIA_StorageCapabilities

The SNIA_StorageCapabilities is subclassed from CIM_StorageCapabilities to add the SupportedDataOrganizations property. The Properties that are different from what is specified in the Block Services Package have descriptive text. Note that SCSI can be coded as NULL or any value other than "4". The class definition specializes the CIM_StorageCapabilities definition in the Block Services profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 191 describes class SNIA_StorageCapabilities.

Table 191 - SMI Referenced Properties/Methods for SNIA_StorageCapabilities

Properties	Flags	Requirement	Description and Notes
InstanceID (overridden)		Mandatory	
ElementName (overridden)		Mandatory	
ElementType (overridden)		Mandatory	

Table 191 - SMI Referenced Properties/Methods for SNIA_StorageCapabilities

Properties	Flags	Requirement	Description and Notes
NoSinglePointOfFailure (overridden)		Mandatory	
NoSinglePointOfFailureDefault (overridden)		Mandatory	
DataRedundancyMin (overridden)		Mandatory	
DataRedundancyMax (overridden)		Mandatory	
DataRedundancyDefault (overridden)		Mandatory	
PackageRedundancyMin (overridden)		Mandatory	
PackageRedundancyMax (overridden)		Mandatory	
PackageRedundancyDefault (overridden)		Mandatory	
ExtentStripeLengthDefault (overridden)		Optional	
ParityLayoutDefault (overridden)		Optional	
UserDataStripeDepthDefault (overridden)		Optional	
SupportedDataOrganizations (added)		Mandatory	Supported values for SMI-S are "4" (Count Key Data) and anything else (including NULL) for non-CKD volumes. CKD Volumes use "4".
SupportedExtentTypes (added)		Mandatory	Supported values for SMI-S are "2" ("Open"), "3" ("Intermediate") and "4" ("Mainframe"). CKD access is supported for either "3" or "4". Open systems access is supported for either "2" or "3".
CreateSetting() (overridden)		Mandatory	
GetSupportedStripeLengths() (overridden)		Optional	
GetSupportedStripeLengthRange() (overridden)		Optional	
GetSupportedParityLayouts() (overridden)		Optional	
GetSupportedStripeDepths() (overridden)		Optional	
GetSupportedStripeDepthRange() (overridden)		Optional	

9.8.51 SNIA_StorageSetting

The SNIA_StorageSetting is subclassed from CIM_StorageSetting and is enhanced to add the DataOrganization, CUIImage, SubsystemID and EmulatedDevice properties. The Properties that are different from what is specified in the Block Services Package have descriptive text. The class definition specializes the CIM_StorageSetting definition in the Block Services profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Extrinsic: StorageCapabilities.CreateSetting

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 192 describes class SNIA_StorageSetting.

Table 192 - SMI Referenced Properties/Methods for SNIA_StorageSetting

Properties	Flags	Requirement	Description and Notes
InstanceID (overridden)		Mandatory	
ElementName (overridden)		Mandatory	
NoSinglePointOfFailure (overridden)		Mandatory	
DataRedundancyMin (overridden)		Mandatory	
DataRedundancyMax (overridden)		Mandatory	
DataRedundancyGoal (overridden)		Mandatory	
PackageRedundancyMin (overridden)		Mandatory	
PackageRedundancyMax (overridden)		Mandatory	
PackageRedundancyGoal (overridden)		Mandatory	
ExtentStripeLength (overridden)		Optional	
ExtentStripeLengthMin (overridden)		Optional	
ExtentStripeLengthMax (overridden)		Optional	
ParityLayout (overridden)		Optional	
UserDataStripeDepth (overridden)		Optional	
UserDataStripeDepthMin (overridden)		Optional	
UserDataStripeDepthMax (overridden)		Optional	
ChangeableType (overridden)		Mandatory	
StorageExtentInitialUsage		Optional	The Usage value to be used when creating a new storage element.
StoragePoolInitialUsage		Optional	The Usage value to be used when creating a new storage pool.
DataOrganization (added)		Mandatory	Supported value for CKD Volumes in SMI-S is "4" (Count Key Data). For non-CKD Volumes the property is either NULL or any value other than "4".
ExtentType (added)		Mandatory	This property specifies extent type for host access. ("1"(=Other), "2"(=Open), "3"(Intermediate), "4"(=Mainframe)).

Table 192 - SMI Referenced Properties/Methods for SNIA_StorageSetting

Properties	Flags	Requirement	Description and Notes
CUIImage (added)		Optional	This property is the Node Element Descriptor of the Control Unit Image (this property is required for CKD StorageVolumes). It is not required for LogicalDisks.
SubsystemID (added)		Optional	This property is the Subsystem ID if the array or virtualizer supports Subsystem IDs. If they are supported they would be required on volume creation.
EmulatedDevice (added)		Optional	This string property specifies the specific device (e.g., 3 380 or 3 390) that is emulated by the volume.

9.8.52 SNIA_StorageVolume

The SNIA_StorageVolume is subclassed from CIM_StorageVolume and enhances that class to add the DataOrganization, CUIImage, SubsystemID and EmulatedDevice properties. Other properties have some unique CKD considerations. The StorageVolume is listed as optional. The Properties that are different from what is specified in the Block Services Package have descriptive text. The class definition specializes the CIM_StorageVolume definition in the Block Services profile. Properties or methods not inherited are marked accordingly as '(overridden)' or '(added)' in the left most column.

Created By: Static

Modified By: Static

Deleted By: Extrinsic: StorageConfigurationService.ReturnToStoragePool

Requirement: Referenced from either Array or Storage Virtualizer - StorageVolume is mandatory.

Table 193 describes class SNIA_StorageVolume.

Table 193 - SMI Referenced Properties/Methods for SNIA_StorageVolume

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName (overridden)		Mandatory	
SystemName (overridden)		Mandatory	
CreationClassName (overridden)		Mandatory	
DeviceID (overridden)		Mandatory	
ElementName (overridden)		Optional	
Name (overridden)	CD	Mandatory	An Identifier for this volume.
OtherIdentifyingInfo (overridden)	CD	Optional	
IdentifyingDescriptions (overridden)		Optional	
NameFormat (overridden)		Mandatory	Format for Name property. For CKD Volumes, this shall be set to "12" (NED).
NameNamespace		Mandatory	The namespace that defines uniqueness for the NameFormat.
ExtentStatus (overridden)		Mandatory	
OperationalStatus (overridden)		Mandatory	

Table 193 - SMI Referenced Properties/Methods for SNIA_StorageVolume

Properties	Flags	Requirement	Description and Notes
BlockSize (overridden)		Mandatory	The BlockSize would report the number of bytes in a cylinder.
NumberOfBlocks (overridden)		Mandatory	The number of blocks would be the number of cylinders.
ConsumableBlocks (overridden)		Mandatory	The number of usable cylinders.
IsBasedOnUnderlyingRedundancy (overridden)		Mandatory	
NoSinglePointOfFailure (overridden)		Mandatory	
DataRedundancy (overridden)		Mandatory	
PackageRedundancy (overridden)		Mandatory	
DeltaReservation (overridden)		Mandatory	
Usage		Optional	The specialized usage intended for this element.
OtherUsageDescription		Optional	Set when Usage value is "Other".
ClientSettableUsage		Optional	Lists Usage values that can be set by a client for this element.
Primordial		Mandatory	Shall be false.
ExtentDiscriminator		Mandatory	Experimental. This is an array of values that shall contain 'SNIA:Allocated'.
DataOrganization (added)		Mandatory	Supported value for CKD Storage Volumes in SMI-S is "4" (Count Key Data). For non-CKD volumes the property is either NULL or any value other than "4".
ExtentType (added)		Mandatory	This property specifies extent type for host access. ("1" (=Other), "2" (=Open), "3" (Intermediate), "4" (=Mainframe)).
CUIImage (added)		Mandatory	This property is the Node Element Descriptor of the Control Unit Image (this property is required for CKD Volumes).
SubsystemID (added)		Optional	This property is the Subsystem ID if the array or virtualizer supports Subsystem IDs. If they are supported they would be required on volume creation.
EmulatedDevice (added)		Optional	This string property specifies the specific device (e.g., 3 380 or 3 390) that is emulated by the volume.

9.8.53 SNIA_StorageVolume

This represents the same instance as CIM_StorageVolume, but is extended to support the CanDelete property.

Created By: Static

Modified By: Static

Deleted By: Extrinsic: StorageConfigurationService.ReturnToStoragePool

Requirement: Optional.

Table 194 describes class SNIA_StorageVolume.

Table 194 - SMI Referenced Properties/Methods for SNIA_StorageVolume

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	
ElementName		Optional	
Name		Mandatory	
OtherIdentifyingInfo		Optional	
IdentifyingDescriptions		Optional	
NameFormat		Mandatory	
NameNamespace		Mandatory	
ExtentStatus		Mandatory	
OperationalStatus		Mandatory	
BlockSize		Mandatory	
NumberOfBlocks		Mandatory	
ConsumableBlocks		Mandatory	
IsBasedOnUnderlyingRedundancy		Mandatory	
NoSinglePointOfFailure		Mandatory	
DataRedundancy		Mandatory	
PackageRedundancy		Mandatory	
DeltaReservation		Mandatory	
Usage		Optional	
OtherUsageDescription		Optional	
ClientSettableUsage		Optional	
Primordial		Mandatory	
ExtentDiscriminator		Mandatory	Experimental. This is an array of values that shall contain 'SNIA:Allocated'.
CanDelete		Optional	Experimental. Indicates if the volume is able to be deleted by a client application.

EXPERIMENTAL

STABLE

10 Copy Services Subprofile

10.1 Description

10.1.1 Synopsis

Profile Name: Copy Services (Component Profile)

Version: 1.5.0

Organization: SNIA

CIM Schema Version: 2.23

Table 195 describes the related profiles for Copy Services.

Table 195 - Related Profiles for Copy Services

Profile Name	Organization	Version	Requirement	Description
Block Services	SNIA	1.5.0	Mandatory	
Job Control	SNIA	1.5.0	Optional	

Central Class: N/A

Scoping Class: ComputerSystem

10.1.2 Overview

The Copy Services Subprofile is an optional subprofile for the Array, Virtualization and Volume Manager Profiles.

The subprofile defines a management interface for local mirror management, local snapshot management and clone management.

The subprofile specification uses terminology consistent with the SNIA dictionary of storage networking except for the term clone. A clone is a fully copied replica the same size as the source element created with the intent of becoming an independent element.

Two types of synchronization views are supported. A replica may be synchronized to the current view of the source element or may be synchronized to a point-in-time view. Snapshots and clones always represent a point-in-time view of the source element. A mirror can represent either a current view or a point-in-time view as indicated by the synchronization state property of the association. A provider maintains a stateful view of a source element as long as the source and replica association is maintained. The synchronization view is modeled with a StorageSynchronized association. A client can determine the type and state of the synchronized view by inspecting properties of the association instance.

EXPERIMENTAL

Two copy operation modes are supported – synchronous and asynchronous. In the synchronous mode, the write operations to the source elements are reflected to the target elements before signalling the host that a write operation is complete. In the asynchronous mode, the host is signaled as soon as the write operations to the source elements are complete. However, the writes to the target elements may take place at a later time.

EXPERIMENTAL

The subprofile supports two types of storage elements. Replicas can be instances of StorageVolume or LogicalDisk. The source and replica elements shall be the same element type. All of the instance diagrams that follow show StorageVolume replicas but apply equally to LogicalDisk replicas.

A copy service for storage elements deploys some type of copy engine. Copy techniques for storage elements include full background copy, copy-on-write and copy-on-read. Most aspects of copy engines are opaque to clients. A provider may allow the client to manage the copy engine for background copy operations. This optional capability is discussed in 10.6.9.

EXPERIMENTAL

The subprofile includes a variable space consumption model that a provider may use for delta replica elements. Most storage elements receive a fixed allocation of space when the element is created and the consumed space is a contiguous block set. Delta replicas may not receive any space allocation when created and, subsequently, consume space one block at a time as the associated source element is updated. The resulting block set for a delta replica is typically scattered throughout a container element such as a storage pool.

Replication Services supports “copying” thinly provisioned elements. Unlike fully provisioned elements, a thinly provisioned element has fewer actual allocated storage blocks than the advertised capacity of the element.

The Replication Service generally relies on the implementation’s copy engine to perform the actual copy operations. However, the profile can expose the “copy methodology” if that information is available.

EXPERIMENTAL

IECNORM.COM : Click to view the full PDF of ISO/IEC 24775-4:2014

10.1.3 Copy Services Discovery

The extrinsic methods invoked to create and manage replicas are defined in the StorageConfigurationService class shown in Figure 46.

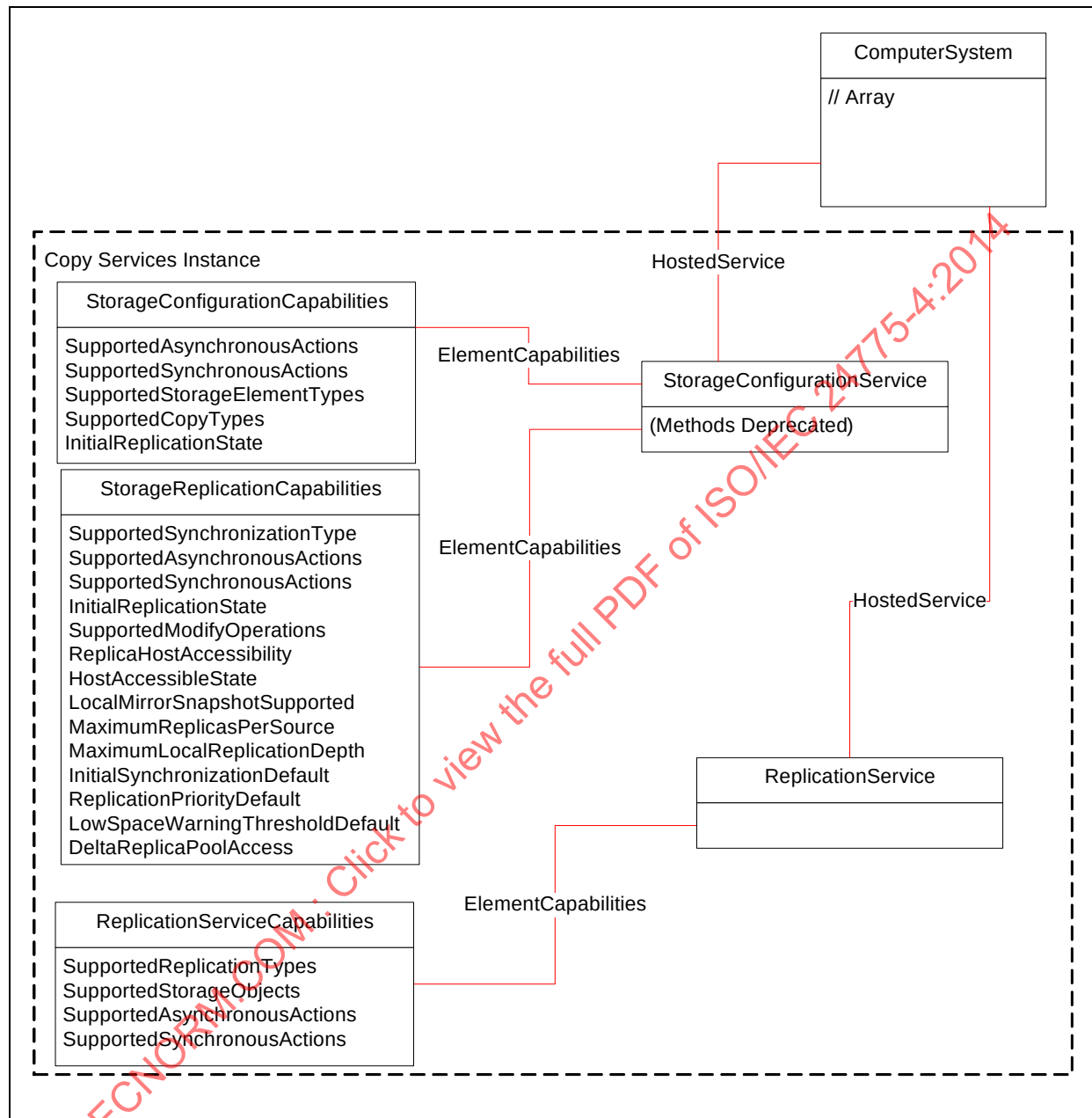


Figure 46 - Copy Services Discovery

EXPERIMENTAL

The single instance of the class ReplicationService and its methods provide the mechanism for creating and managing replicas.

Replication Services relies on the Block Services Package for storage pool manipulations and capacity related indications.

EXPERIMENTAL

10.1.4 Copy Services Capabilities

10.1.4.1 General

The Copy Services Subprofile enables a provider to deploy all of the modeled replication capabilities in a single service instance. For example, one service instance may support local mirrors and delta snapshots. A client discovers and analyzes each of these capabilities as shown in Figure 46: "Copy Services Discovery".

EXPERIMENTAL

The StorageConfigurationService methods for performing copy functions are being deprecated, but the StorageConfigurationCapabilities and ReplicationServiceCapabilities are not being deprecated. The newer methods for performing copy functions are in the ReplicationService, which has its own Capabilities class. Both the StorageConfigurationCapabilities and the ReplicationServiceCapabilities would be associated to the StorageConfigurationService. This subclause discusses both sets of capabilities and how they relate.

EXPERIMENTAL

10.1.4.2 Replication Policy

A provider exposes an instance of StorageReplicationCapabilities for each replication capabilities supported. The CopyType property as defined in CIM_StorageSynchronized describes the replication policies supported by the subprofile.

- **Async:** Create and maintain an asynchronous mirror copy of the source.
- **Sync:** Create and maintain a synchronous mirror copy of the source. Writes done to the source element are reflected to the mirror before signalling the host that the write is complete. Used to maintain a copy requiring guaranteed consistency during a recovery operation.
- **UnSyncAssoc:** Creates an unsynchronized copy associated to the source element. This type of copy is called a "snapshot" and represents a point-in-time image of the source element. Separate instances of StorageReplicationCapabilities may be defined for full size snapshots and delta snapshots corresponding to this CopyType value.
- **UnSyncUnAssoc:** Creates an unsynchronized clone of the source element and does not maintain the source association after completing the copy operation.

EXPERIMENTAL

In addition, an implementation may specify SyncTypes to describe the replication policy supported by the profile. The following SyncTypes are defined:

- **Mirror:** Creates and maintains a synchronized mirror copy of the source. Writes done to the source element are reflected to the target element. The target element remains dependent on the source element.
- **Snapshot:** Creates a point-in-time, virtual image of the source element. The target element remains dependent on the source element. Snapshots are commonly known as delta replicas and contain incrementally changed data as well as the pointers to the unchanged source element data.
- **Clone:** Creates a point-in-time, independent, copy of the source element.

Synchronized replication indicates that updates to a source element are reflected to the target element. The mode determines whether the target element is updated immediately, in the case of synchronous mode, or some time later, in the case of asynchronous mode.

Table 196 compares the SyncTypes and the relationships between the source and target elements. It is a quick reference for the clients to determine the appropriate SyncType for the intended target results.

Table 196 - Comparing SyncTypes

SyncType	Relation of Target to Source	Updates to Source Reflected to Target	Target is Point-In-Time Copy	Target is self-contained	Target is Virtual copy of Source	Target's space consumption
Mirror	Dependent	Yes	No	Yes-after Split/Detach	No	Same as source
Snapshot	Dependent	No	Yes	No	Yes	Much less than source
Clone	Independent	No	Yes	Yes	No	Same as source

With respect to "Relation of Target to Source," Dependent indicates the target element shall remain associated with the source element; Independent indicates the target element can exist without the source element.

10.1.4.3 Modes

The mode controls when the write operations are performed. The following modes are defined:

- Synchronous: The writer waits until the write operations are committed to both the source and target elements, or to both the source element and a target related entity, such as pointer tables.
- Asynchronous: The writer waits until the write operations are committed to the source elements only. In this mode, there can be a delay before the write operations are committed to the target elements.

10.1.4.4 Alignment of SupportedSynchronizationType and SupportedReplicationType

The values for SupportedSynchronizationType (in StorageReplicationCapabilities) and SupportedReplicationType (in ReplicationServiceCapabilities) should be aligned with each other. Table 197 the alignment of these properties.

Table 197 - Alignment of SupportedSynchronizationType and SupportedReplicationType

Supported ReplicationType	Supported Synchronization Type	Notes
Synchronous Mirror Local	Sync	If an implementation supports the "Sync" SupportedSynchronizationType, then it should report that it supports a "Synchronous Mirror Local" SupportedReplicationType.
Asynchronous Mirror Local	Async	If an implementation supports the "Async" SupportedSynchronizationType, then it should report that it supports a "Asynchronous Mirror Local" SupportedReplicationType.
Synchronous Snapshot Local	UnsyncAssoc - Full	If an implementation supports the "UnsyncAssoc - Full" SupportedSynchronizationType, then it may report that it supports a "Synchronous Snapshot Local" SupportedReplicationType.
	UnsyncAssoc - Delta	If an implementation supports the "UnsyncAssoc - Delta" SupportedSynchronizationType, then it may report that it supports a "Synchronous Snapshot Local" SupportedReplicationType.
Asynchronous Snapshot Local	UnsyncAssoc - Full	If an implementation supports the "UnsyncAssoc - Full" SupportedSynchronizationType, then it may report that it supports a "Asynchronous Snapshot Local" SupportedReplicationType.
	UnsyncAssoc - Delta	If an implementation supports the "UnsyncAssoc - Delta" SupportedSynchronizationType, then it may report that it supports a "Asynchronous Snapshot Local" SupportedReplicationType.
Synchronous Clone Local	UnsyncUnassoc	If an implementation supports the "UnsyncUnassoc" SupportedSynchronizationType, then it may report that it supports a "Synchronous Clone Local" SupportedReplicationType.
Asynchronous Clone Local		If an implementation supports the "UnsyncUnassoc" SupportedSynchronizationType, then it may report that it supports a "Asynchronous Clone Local" SupportedReplicationType.

EXPERIMENTAL**10.1.4.5 Other Capabilities**

The StorageReplicationCapabilities class defines informational properties with un-modifiable values that guide a client using the various capabilities of the service. For example:

- Instance 1 defines the capability to create local mirrors. SupportedSynchronizationType is set to a value of "Sync" and the AttachReplica method is the only method supported for mirror creation. The InitialReplicationState is "Synchronized".
- Instance 2 defines the capability to create snapshots. SupportedSynchronizationType is set to a value of "UnSyncAssoc - Delta" and the CreateReplica method is the only method supported for snapshot creation. The InitialReplicationState is "Idle".

Further details concerning discovery and the use of capability properties are included in 10.6 "Client Considerations and Recipes".

10.1.5 Replication modeling

10.1.5.1 General

Figure 47: "Local Replica" shows the basic model of a local replica.

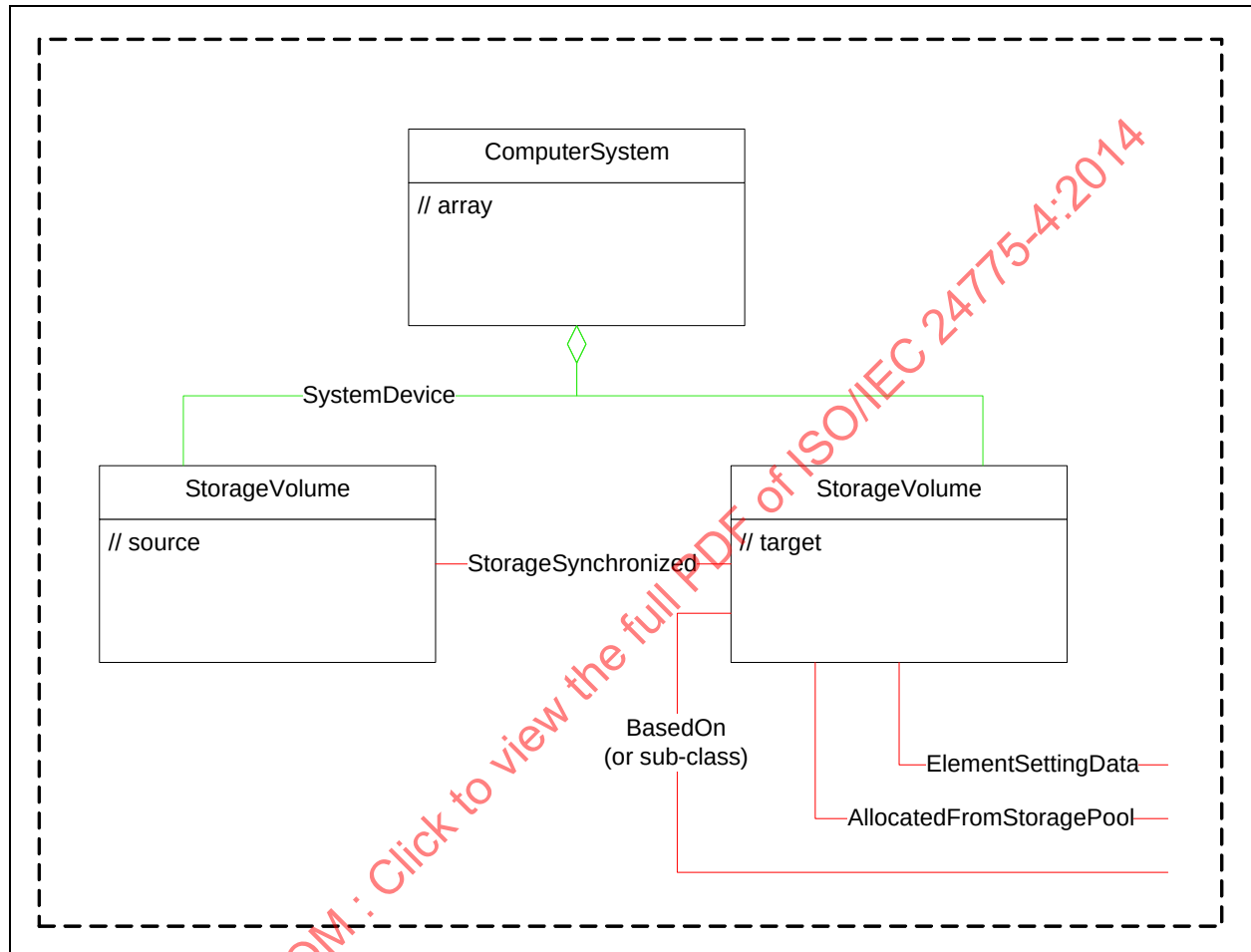


Figure 47 - Local Replica

A local replica is created by invoking either the CreateReplica or the AttachReplica extrinsic methods. CreateReplica creates a new storage element in a storage pool. AttachReplica transforms an existing, independent storage element into a replica. The new replica is the same element type as the source element. Several associations are implicitly created for all replica elements. A StorageSynchronized association shall be created if the new replica remains associated with its source element. A SystemDevice association shall be created or shall already exist. An AllocatedFromStoragePool association shall be created or shall already exist. An ElementSettingData association with an instance of StorageSetting is created or shall already exist for the replica element. An optional BasedOn association may exist if AttachReplica is invoked to transform an existing element into an associated replica.

EXPERIMENTAL

The CreateReplica method allows a client to delegate the selection of a target element location and settings to the invoked provider. The client selects a source element for the replication operation and may optionally choose to supply a storage pool location and storage settings or to let the provider make the choices. The AttachReplica method allows a client to completely

manage the source/target replication pairing. The client creates a new target element or selects an existing element to be used as the target. Once the target element is prepared, the client invokes the AttachReplica method and the provider pairs the source and target elements selected by the client. All providers shall support at least one of these two methods.

EXPERIMENTAL

10.1.5.2 Multiple Replicas

The subprofile supports both multiple replicas per associated source element and multi-level replication. Properties in StorageReplicationCapabilities allow the provider to indicate the maximum number of replicas for one source element and the maximum depth for multi-level replication. Figure 48: "Multi-Level Local Replication" show the basic model for local multi-level replication.

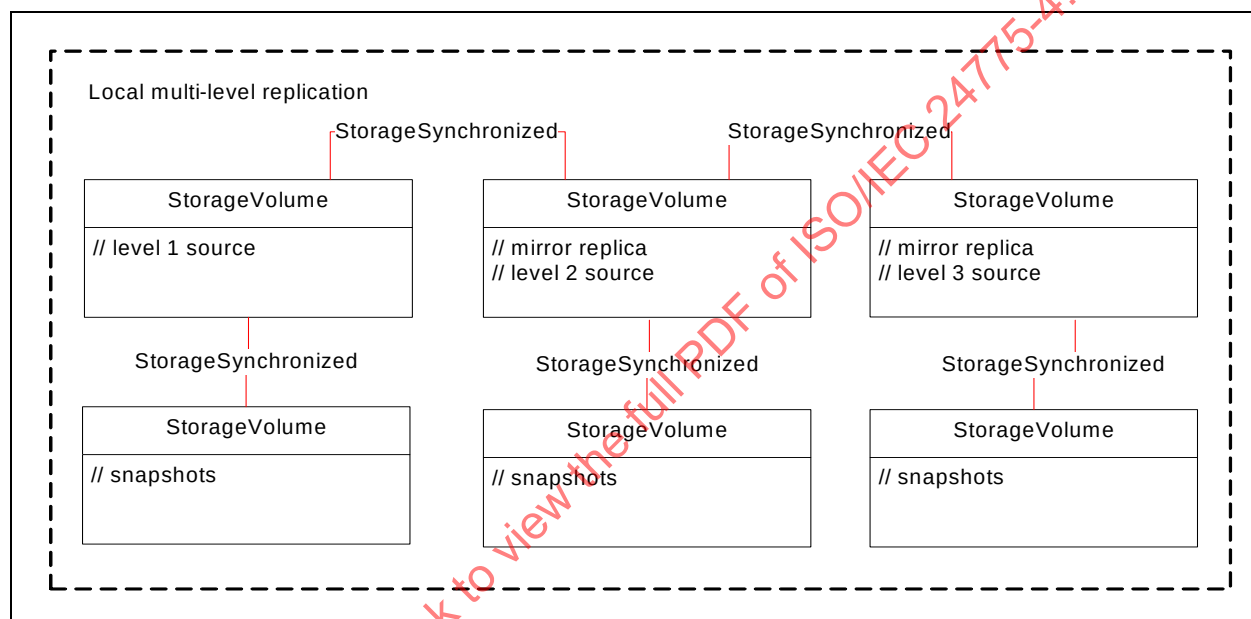


Figure 48 - Multi-Level Local Replication

EXPERIMENTAL

If an implementation supports multi-hop replication, the supported features (obtained via the GetSupportedFeatures method) will indicate "Multi-hop element replication". Furthermore, the implementation may need to know that the client is planning to add additional hops in subsequent operations. In this case, the replication capabilities would indicate "Multi-hop requires advance notice". In response to this capability, the client in creating the first replica, shall set the property ReplicationSettingData.Multihop appropriately. See 10.7 "CIM Elements" for details on Multihop specification. The capabilities method GetSupportedMaximum indicates the maximum number of hops supported by the implementation.

EXPERIMENTAL

10.1.5.3 Snapshots

Snapshots are created using CopyType "UnSyncAssoc" when either the CreateReplica or AttachReplica extrinsic method is invoked. Snapshots may be created as full replicas or delta replicas. A provider supporting delta replicas may enable several optional capabilities used with the variable space consumption model described in 10.6 "Client Considerations and Recipes". A client uses these capabilities to ensure sufficient but not excessive availability of

space for groups of delta replicas. Action can be taken by a client to prevent failure of delta replica elements caused by lack of consumable space.

Figure 49: "Multiple Snapshots Per Source Element" shows the basic model of snapshots created as delta replicas.

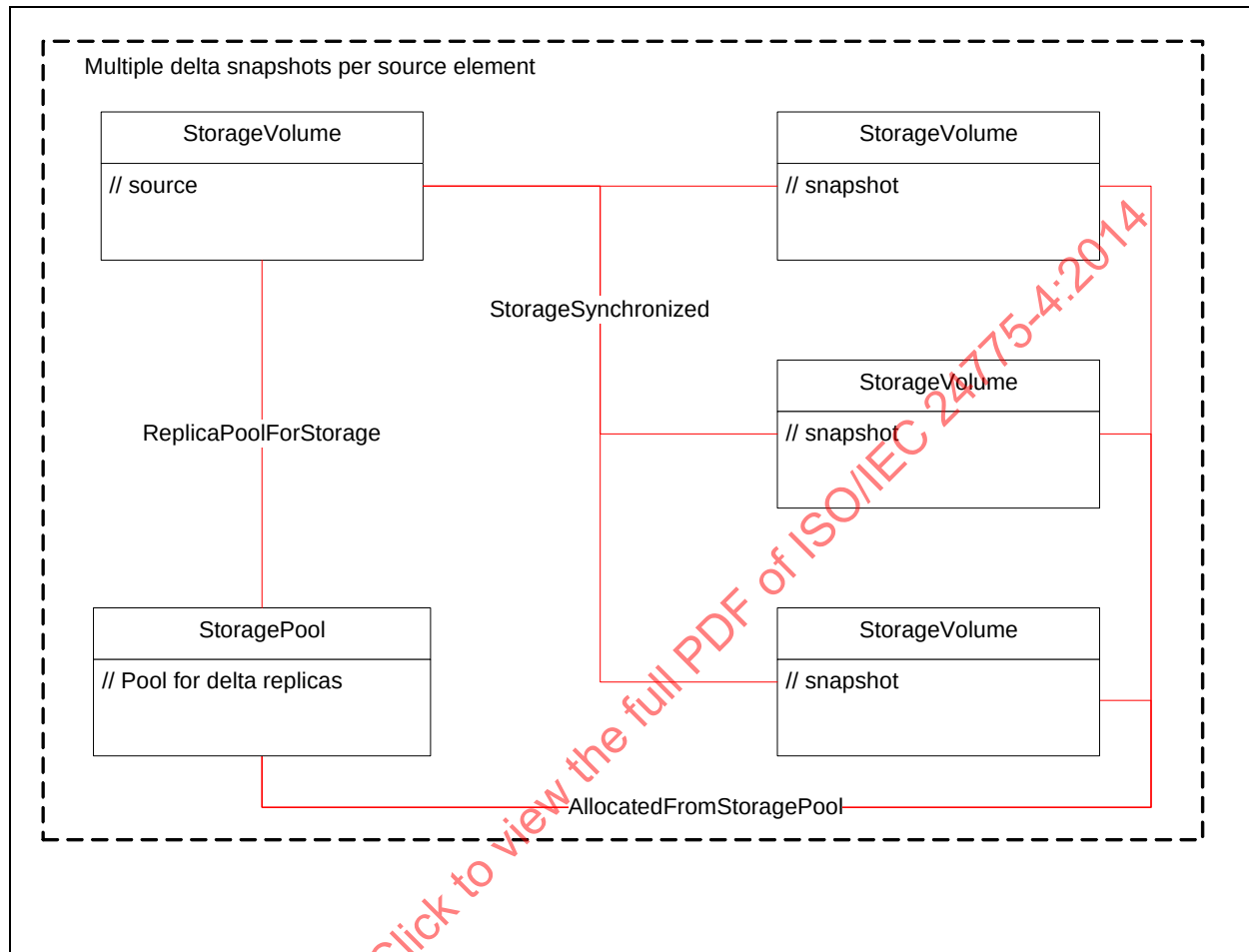


Figure 49 - Multiple Snapshots Per Source Element

10.1.6 Associations

10.1.6.1 General

Copy Services utilizes associations.

10.1.6.2 StorageSynchronized Association

10.1.6.2.1 General

This association relates the individual source and target elements. The association's property SyncState indicates the current state of the association. Some possible values of SyncState are Initialized or Synchronized.

In addition to the SyncState, there are a number of other properties on the StorageSynchronized Association. These include:

- WhenSynced: This is the date/time of the creation of a point in time copy.
- SyncMaintained: This indicates whether synchronization is maintained.
- CopyType: This defines the type of (copy) association between source and target.

- **ReplicaType:** This is an informational property describing the type of replication.

EXPERIMENTAL

- **CopyPriority:** Priority of copy engine I/O relative to host I/O.

In addition, there are a number of other properties that are being added to the StorageSynchronized Association. These include:

- **WhenEstablished:** Specifies when the association was established.
- **WhenActivated:** Specifies when the association was activated.
- **WhenSuspended:** Specifies when the association was suspended.
- **SyncType:** Type of association between source and target elements.
- **Mode:** Specifies when target elements are updated.
- **RequestedCopyState:** Indicates the last requested or desired state for the association.
- **CopyState:** indicates the current state of the association.
- **ProgressStatus:** Status of association between source and target groups.
- **PercentSynced:** Specifies the percent of the work completed to reach synchronization.

10.1.6.2.2 Alignment of StorageSynchronized Properties

The SyncType and mode properties and the CopyType property are related and their values should be aligned as shown in Table 198.

Table 198 - Alignment of SyncType/Mode and CopyType

SyncType / Mode	CopyType	Notes
Mirror / Asynchronous	Async	If an implementation reports SyncType="Mirror" and Mode="Asynchronous", then it should report CopyType="Async".
Mirror / Synchronous	Sync	If an implementation reports SyncType="Mirror" and Mode="Synchronous", then it should report CopyType="Sync".
Snapshot / Synchronous	UnsyncAssoc	If an implementation reports SyncType="Snapshot" and Mode="Synchronous" or Mode="Asynchronous", then it should report CopyType="UnsyncAssoc".
Snapshot / Asynchronous		
Clone / Synchronous	UnsyncUnAssoc	If an implementation reports SyncType="Clone" and Mode="Synchronous" or Mode="Asynchronous", then it should report CopyType="UnsyncUnAssoc".
Clone / Asynchronous		

The CopyState and ProgressStatus and SyncState properties are related and their values should be aligned as shown in Table 199.

Table 199 - Alignment of CopyState and SyncState

CopyState / ProgressStatus	SyncState	Notes
Initialized / Completed	Initialized	If an implementation reports CopyState="Initialized" and ProgressStatus="Completed", then it should report SyncState="Initialized".

Table 199 - Alignment of CopyState and SyncState

CopyState / ProgressStatus	SyncState	Notes
Initialized / Preparing	Prepare In Progress	If an implementation reports CopyState="Initialized" and ProgressStatus="Preparing", then it should report SyncState="Prepare In Progress".
Prepared / Completed	Prepared	If an implementation reports CopyState="Prepared" and ProgressStatus="Completed", then it should report SyncState="Prepared".
Unsynchronized / Synchronizing	ResyncInProgress	If an implementation reports CopyState="Unsynchronized" and ProgressStatus="Synchronizing", then it should report SyncState="ResyncInProgress".
Synchronized / Completed	Synchronized or Frozen	If an implementation reports CopyState="Synchronized" and ProgressStatus="Completed", then it should report SyncState="Synchronized" or SyncState="Frozen".
Initialized / Completed	PrepareInProgress	If an implementation reports CopyState="Initialized" and ProgressStatus="Completed", then it should report SyncState="PrepareInProgress".
Prepared / Completed	Prepared	If an implementation reports CopyState="Prepared" and ProgressStatus="Completed", then it should report SyncState="Prepared".
Prepared / Synchronizing	ResyncInProgress	If an implementation reports CopyState="Prepared" and ProgressStatus="Synchronizing", then it should report SyncState="ResyncInProgress".
Unsynchronized / Suspending	Quiesce In Progress	If an implementation reports CopyState="Unsynchronized" and ProgressStatus="Suspending", then it should report SyncState="Quiesce In Progress".
Unsynchronized / Dormant	Quiesce In Progress	If an implementation reports CopyState="Unsynchronized" and ProgressStatus="Dormant", then it should report SyncState="Quiesce In Progress".
Synchronized / Completed	Synchronized	For mirrors, if an implementation reports CopyState="Synchronized" and ProgressStatus="Completed", then it should report SyncState="Synchronized".
Synchronized / Completed	Idle	For snapshots, if an implementation reports CopyState="Synchronized" and ProgressStatus="Completed", then it should report SyncState="Idle".
Synchronized / Suspending	Quiesce In Progress	If an implementation reports CopyState="Synchronized" and ProgressStatus="Suspending", then it should report SyncState="Quiesce In Progress".
Synchronized / Fracturing	Fracture In Progress	If an implementation reports CopyState="Synchronized" and ProgressStatus="Fracturing", then it should report SyncState="Fracture In Progress".
Synchronized / Splitting	Fracture In Progress	If an implementation reports CopyState="Synchronized" and ProgressStatus="Splitting", then it should report SyncState="Fracture In Progress".
Synchronized / Failing Over	RestoreInProgress	If an implementation reports CopyState="Synchronized" and ProgressStatus="Failing Over", then it should report SyncState="RestoreInProgress".
Synchronized / Dormant	Quiesce In Progress	If an implementation reports CopyState="Synchronized" and ProgressStatus="Dormant", then it should report SyncState="Quiesce In Progress".
Synchronized / Initializing	Initialized	If an implementation reports CopyState="Synchronized" and ProgressStatus="Initializing", then it should report SyncState="Initialized".

Table 199 - Alignment of CopyState and SyncState

CopyState / ProgressStatus	SyncState	Notes
Fractured / Completed	Fractured	If an implementation reports CopyState="Fractured" and ProgressStatus="Completed", then it should report SyncState="Fractured".
Fractured / Resyncing	ResyncInProgress	If an implementation reports CopyState="Fractured" and ProgressStatus="Resyncing", then it should report SyncState="ResyncInProgress".
Split / Completed	Fractured	If an implementation reports CopyState="Split" and ProgressStatus="Completed", then it should report SyncState="Fractured".
Split / Resyncing	ResyncInProgress	If an implementation reports CopyState="Split" and ProgressStatus="Resyncing", then it should report SyncState="ResyncInProgress".
Suspended / Completed	Quiesced	If an implementation reports CopyState="Initialized" and ProgressStatus="Completed", then it should report SyncState="Quiesced".
Suspended / Resyncing	ResyncInProgress	If an implementation reports CopyState="Suspended" and ProgressStatus="Resyncing", then it should report SyncState="ResyncInProgress".
Broken / Not Applicable	Broken	If an implementation reports CopyState="Broken" and ProgressStatus="Not Applicable", then it should report SyncState="Broken".
Inactive / Completed	Quiesced	For mirrors, if an implementation reports CopyState="Inactive" and ProgressStatus="Completed", then it should report SyncState="Quiesced".
Inactive / Completed	Idle	For snapshots, if an implementation reports CopyState="Inactive" and ProgressStatus="Completed", then it should report SyncState="Idle".
Inactive / Resyncing	ResyncInProgress	If an implementation reports CopyState="Inactive" and ProgressStatus="Resyncing", then it should report SyncState="ResyncInProgress".
Aborted / Completed	Quiesced	For mirrors, if an implementation reports CopyState="Aborted" and ProgressStatus="Completed", then it should report SyncState="Quiesced".
Aborted / Completed	Idle	For snapshots, if an implementation reports CopyState="Aborted" and ProgressStatus="Completed", then it should report SyncState="Idle".
Failedover / Completed	Fractured	For mirrors, if an implementation reports CopyState="Failedover" and ProgressStatus="Completed", then it should report SyncState="Fractured".
Failedover / Completed	Frozen	For snapshots, if an implementation reports CopyState="Failedover" and ProgressStatus="Completed", then it should report SyncState="Frozen".
Synchronized / Failing back	RestoreInProgress	If an implementation reports CopyState="Synchronized" and ProgressStatus="Failing back", then it should report SyncState="RestoreInProgress".
Skewed / Completed	Initialized	If an implementation reports CopyState="Skewed" and ProgressStatus="Completed", then it should report SyncState="Initialized".
Skewed / Resyncing	ResyncInProgress	If an implementation reports CopyState="Skewed" and ProgressStatus="Resyncing", then it should report SyncState="ResyncInProgress".

10.1.6.3 SettingsDefineState Association

The SettingsDefineState associates an element (e.g., a StorageVolume) to a SynchronizationAspect. An instance of SynchronizationAspect includes properties for the date and time of the point-in-time copy and a reference to the source element (see Figure 50). The association is particularly useful for Clones (targets) and Snapshots (source) that do not have a StorageSynchronized association to another storage element. In the case of Clones, the StorageSynchronized association is removed (generally, following the provider's restart) after the copy operation completes. As for Snapshots, it is possible to create a point-in-time snapshot copy of an element, or a group of elements, without having a target element (using the method CreateSynchronizationAspect). In this mode, the target elements are added at a later time (using the method ModifySettingsDefineState).

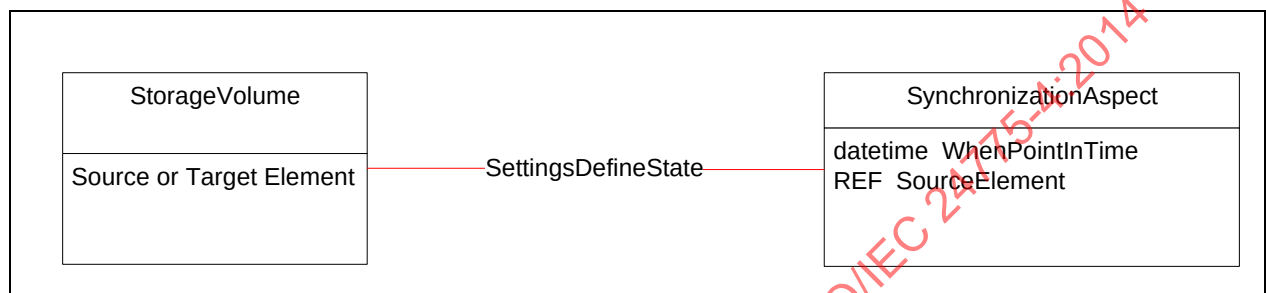


Figure 50 - SettingsDefineState Association

SettingsDefineState may also be applied to Mirror targets; as such, the property SynchronizationAspect.WhenPointInTime would have the date and time of when the mirror relationship was fractured (or split).

In all cases, the SettingsDefineState association may not persist across the provider's restarts. Furthermore, an instance of a SynchronizationAspect shall be removed if the SourceElement is deleted.

Figure 51 is an instance diagram for a clone target element and its associated SynchronizationAspect instance. Once the clone target element becomes synchronized, the StorageSynchronized association is removed and the property SynchronizationAspect.SyncState has a value of "Operation Completed".

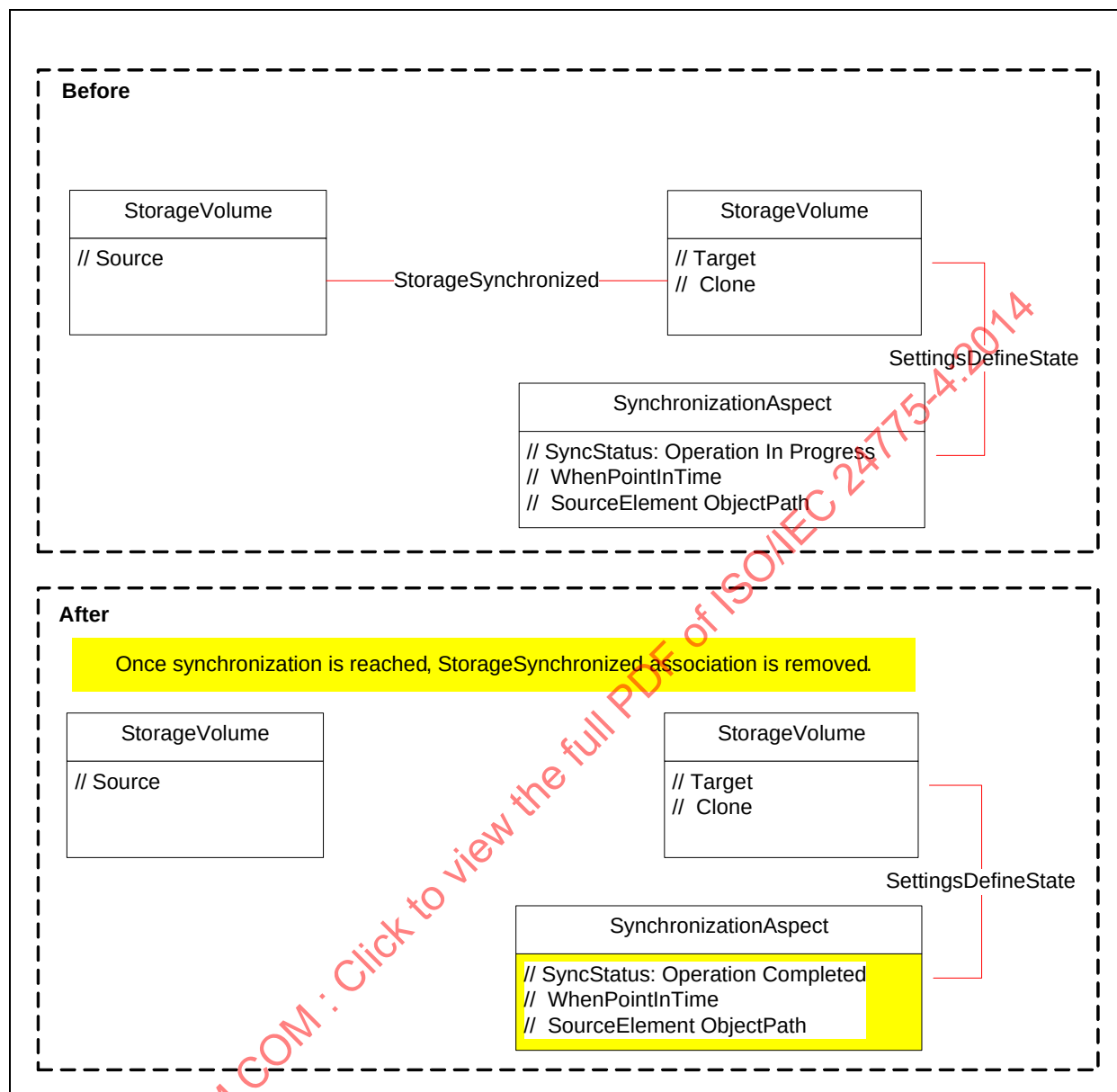


Figure 51 - SynchronizationAspect Instance

EXPERIMENTAL**10.1.7 Durable Names and Correlatable IDs of the Profile**

This is not applicable to local copy services. Normal Block Services Correlatable IDs apply for volumes (or logical disks) managed by Copy Services.

10.1.8 Accessibility to Created Elements**DEPRECATED****10.1.8.1 Using StorageConfigurationService Methods**

The subprofile recommends that method providers for replica creation methods make all replica elements and associations accessible when the method response is returned to the client. This includes the case when the provider returns "job started" to the client. This allows

the client to immediately monitor and manage the replica, new associations to the replica and new associated elements.

If the provider returns “job completed”, all new elements and associations shall be accessible. If “job started” is returned, new elements may not be immediately accessible. There are two cases the provider should consider:

Case 1: a new element and new associations are created (CreateReplica).

If the provider returns a reference to the new element as a method output parameter, all new associations shall also be accessible and AffectedJobElement shall now reference the new element for the returned job reference. No instance creation indications need to be generated. If the provider does not return a reference to the new element, an instance creation indication shall be generated when the new element is accessible. When the job completes successfully, AffectedJobElement shall reference the new element. The new element and all new associations shall be accessible when the instance creation indication is generated or the job completes successfully, whichever occurs first. Instance creation indications are not generated for new associations.

Case 2: a new association is created for an existing element (AttachReplica).

If the provider returns “job started”, AffectedJobElement already references the existing element and the client may attempt to access the new StorageSynchronized association. If the new association is not accessible, an instance creation indication for StorageSynchronized shall be generated when the association is accessible. The new association shall be accessible when the instance creation indication is generated or the job completes successfully, whichever occurs first.

For both cases, at the time an element or association is accessible to the client, all manageable element and association properties have valid values.

DEPRECATED

EXPERIMENTAL

10.1.8.2 Using ReplicationService Methods

Not defined in this version of the International Standard.

EXPERIMENTAL

10.1.9 Completion of Long Operations

DEPRECATED

10.1.9.1 Using StorageConfigurationService Methods

The subprofile supports three ways of indicating the completion of long running operations when a replica element is created or modified. This does not apply to a detach operation.

Option 1:

- 1) Provider returns “job completed” status.
- 2) SyncState value set to “... In Progress”.
- 3) Instance modification or instance deletion indication when SyncState value changes to final, steady state.

Option 2:

- 1) Provider returns "job started" status and REF to replica element.
- 2) SyncState value set to "... In Progress".
- 3) Instance modification or instance deletion indication when SyncState value changes to final, steady state.
- 4) Instance modification when ConcreteJob ends.

Option 3:

- 1) Provider returns "job started" status but no REF to replica element.
- 2) Instance creation indication for StorageSynchronized when element is available. May indicate "... In Progress" state or final state.
- 3) Instance modification or instance deletion indication when SyncState value changes to final, steady state.
- 4) Instance modification when ConcreteJob ends.

Options 2 and 3 based on job control allow a provider to indicate "percent complete" for long operations and report job failure information with an instance of Error.

Any option may be selected for un-associated replicas if the provider creates a temporary instance of StorageSynchronized that is implicitly deleted when the replica is finished. If a temporary instance is not created, then only options 2 and 3 may be selected and steps 2 and 3 are bypassed.

The ModifySynchronization detach operation and the ReturnToStoragePool method cause element and association deletion. There are two ways to indicate completion of long delete operations.

Option 1:

Provider returns "job completed". All affected elements and associations are no longer accessible. No instance deletion indications should be generated.

Option 2:

- 1) Provider returns "job started" status. Client assumes elements and associations are no longer accessible.
- 2) An instance deletion indication is generated for StorageSynchronized for a detach operation or for a replica element for a ReturnToStoragePool invocation. The element is successfully deleted when either job completion occurs or the instance deletion indication is generated, whichever occurs first.

DEPRECATED

EXPERIMENTAL

10.1.9.2 Using ReplicationService Methods

There are two ways of indicating the completion of long running operations when a replica element is created or modified:

Option 1: Generally, the long running operations are performed under the control of a job. The client can monitor the progress of the job by polling the job's status and percent complete, or by subscribing to job related indications.

Option 2: Subscribe to receive indications when the CopyState of StorageSynchronized changes.

Clients may utilize both options simultaneously. To avoid receiving many indications, it is recommended for the clients to utilize indication queries that are constrained by the object path of the appropriate replication association.

If replication operation was specified with a WaitForCopyState parameter, the job “waits” until at least the CopyState is reached, at which point the job considers the operation complete. However, depending on the specified WaitForCopyState, the copy engine may continue until a steady state is achieved. For example, in the Figure 54, Inactive and Synchronized states are considered steady states, whereas Initialized and Unsynchronized are transient states.

EXPERIMENTAL

10.1.10 State Management For Associated Replicas

Both mirror and snapshot replicas maintain stateful associations with source elements. The SyncState property of a StorageSynchronized association identifies the state. All providers shall support the deprecated ModifySynchronization extrinsic method that allows a client to manage the synchronization state of an associated replica unless a provider only allows unassociated replicas. All of the modify operations supported by the subprofile are classified as mandatory, optional or not supported by type of replica. Mirror replicas are the only type of replica created for CopyType values “Sync” and “Async”. Snapshot replicas are the only type of replica created for CopyType value “UnSyncAssoc”. Table 200 shows the classification.

Table 200 - Synchronization Operation Support Requirements

ModifySynchronization Operation	Mirror Replicas	Snapshot Replicas
Detach	Mandatory	Optional
Resync	Mandatory	Mandatory
Fracture	Mandatory	Not supported
Quiesce	Optional	Optional
Unquiesce	Optional	Not supported
Prepare	Optional	Optional
Unprepare	Optional	Optional
Restore	Optional	Optional
Start Copy	Not supported	Optional
Stop Copy	Not Supported	Optional
Reset To Sync	Optional	Not supported
Reset To Async	Optional	Not supported

All instances of StorageReplicationCapabilities shall indicate all mandatory operations plus all supported optional operations in the value list assigned to the SupportedModifyOperations[] property. Undeployed optional operations should be implemented as a stubbed “no operation” to ensure backward compatibility with earlier versions of the subprofile. Modify operations perform the following actions:

Resync: Causes a fractured mirror replica to change from a point-in-time (PIT) view to a synchronized mirror replica representing the current view of the source element. The provider can execute a full or incremental copy as needed to realize a synchronized state. Causes a snapshot to be restarted as a new PIT image with a new value assigned to WhenSynced. May release all space previously consumed by the snapshot.

Fracture: Splits a synchronized mirror replica from its source element, changing the replica from a current view of the source element to a PIT view.

Restore: Copies a fractured mirror or a snapshot to the source element. At the completion of the restore operation, the source and replica represent the same PIT view. The Restore operation for each supported CopyType can be implemented as an incremental restore or a full restore based on the capabilities of the provider.

Detach: Removes the association between the source and replica elements. The StorageSynchronized association is deleted. If the replica is still a valid PIT image, the provider sets OperationalStatus to "OK". If not a valid image but the storage element can be reused, the provider sets OperationalStatus to "Error". A Detach operation does not delete the replica element. A client should invoke ReturnToStoragePool if the element is to be deleted following the Detach operation.

Start Copy: Starts a background copy operation for a snapshot replica. At the completion of the copy operation, the snapshot enters "Frozen" state.

Stop Copy: Stops a background copy operation for a snapshot replica. The snapshot state changes from "Copy In Progress" to "Idle".

Quiesce/Unquiesce: This operation has optional, vendor-specific behavior for mirror replicas that is opaque to clients. The Quiesce operation stops the copy engine for snapshots and the snapshot no longer consumes space. A snapshot is no longer a valid PIT image if the source element is updated after the snapshot enters "Quiesced" state.

Prepare/Unprepare: This operation has optional, vendor-specific behavior for all replica types that may also depend on the entry state. A prepare operation typically starts a copy engine if entered from "Initialized" state.

Reset To Sync: Changes the CopyType value of a mirror replica from "Async" to "Sync".

Reset To Async: Changes the CopyType value of a mirror replica from "Sync" to "Async".

This information is summarized in Table 201.

Table 201 - SyncState Values

Synchronization State (SyncState value)	Mirror Replicas	Snapshot Replicas	Required ModifySynchronization Operations For Optional States
Initialized	Optional	Optional	Prepare
Prepare In Progress	Optional	Optional	
Prepared	Optional	Optional	Unprepare
Resync In Progress	Mandatory	Mandatory	
Synchronized	Mandatory	Not specified	
Idle	Not specified	Mandatory	
Quiesce In Progress	Optional	Optional	Quiesce
Quiesced	Optional	Optional	Quiesce
Fracture In Progress	Mandatory	Not specified	
Fractured	Mandatory	Not specified	
Copy In Progress	Not specified	Optional	Start Copy

Table 201 - SyncState Values (Continued)

Synchronization State (SyncState value)	Mirror Replicas	Snapshot Replicas	Required ModifySynchronization Operations For Optional States
Frozen	Not specified	Mandatory	
Restore In Progress	Optional	Optional	Restore
Broken	Optional	Optional	

EXPERIMENTAL

In addition, an implementation may maintain CopyState and ProgressStatus for a StorageSynchronized relationship.

The CopyState property of the StorageSynchronized association identifies the state, while the ProgressStatus property of the same association indicates the “status” of the copy operation to reach the requested CopyState, which is indicated in the property RequestedSyncState. For example, CopyState might have a value of “UnSynchronized”, while ProgressStatus might have a value of “Synchronizing”, also known as “sync-in-progress”. In all cases, when creating a replica element, the desired SyncState is Synchronized, which indicates the replica element has the same data as the source element. The RequestedSyncState property will contain “Not Applicable” once the requested SyncState is achieved.

Use the method ReplicationServiceCapabilities.GetSupportedCopyStates to determine the possible CopyStates. The CopyStates have been normalized in such a way that they may apply to all SyncTypes.

IECNORM.COM : Click to view the full PDF of ISO/IEC 24775-4:2014

Table 202 describes the supported CopyStates.

Table 202 - CopyStates Values

CopyState value	Description
Initialized	The source and target elements are associated. The copy engine has not started – no dataflow.
Prepared	Initialization is completed, the copy engine has started, however, the data flow has not started.
Synchronized	The “copy operation” is complete. The target element is an “exact replica” of the source element.
Unsynchronized	Not all the source element data has been “copied” to the target element.
Fractured	The target element was abruptly split from its source element – consistency is not guaranteed.
Split	The target element was gracefully (or systematically) split from its source element – consistency is guaranteed.
Suspended	Data flow between the source and target elements has stopped. Writes to source element are held until the association is Resumed.
Broken	Replica is not a valid view of the source element. OperationalStatus of replica may indicate an Error condition. This state generally indicates an error condition such as broken connection.
Aborted	The copy operation is aborted with the Abort operation. Use the Resync Replica operation to restart the copy operation.
Failedover	Reads and writes to/from the target element. Source element is not “reachable”.
Inactive	Copy engine has stopped, writes to source element will not be sent to target element.
Skewed	The target has been modified and is no longer synchronized with the source element or the point-in-time view.
Mixed	Applies to the SyncState of GroupSynchronized. It indicates the StorageSynchronized associations of the elements in the groups have different SyncState values.

EXPERIMENTAL

10.1.11 Reporting Time of Synchronization

All providers shall have access to a time service that allows the provider to assign a date/time value to the WhenSynced property of StorageSynchronized at the time a replica becomes a valid PIT view of its source element. The WhenSynced value for mirror replicas shall be non-null for the “Fractured” and “Restore In Progress” synchronization states. The WhenSynced value for snapshot replicas shall be non-null for any synchronization state allowing host access to the replica.

10.1.12 State Transition Rules

A provider shall enforce state transition rules for associated replicas. If a client initiates a ModifySynchronization operation that causes a state transition violation, the provider returns an error response of “Invalid State Transition”. The provider shall allow a client to bypass certain transitions related to operations not supported by the provider. For example, a snapshot transition from “Idle” to “Resync In Progress” is allowed if the provider does not support Quiesce and Prepare operations.

Synchronization states have the following behavior:

Initialized: A source element and replica element are associated and all implicitly created associations are accessible. The copy engine has not started.

Synchronized: A mirror replica is fully copied and represents the current view of the source element.

Idle: A snapshot is accessible but not copied and represents a PIT view of the source element. A copy engine is actively executing copy-on-write operations.

Fractured: A mirror element is split from its source element and is now a PIT view.

Frozen: A snapshot is accessible and fully copied and represents a PIT view of the source element. The copy engine is stopped.

Broken: A replica is not a valid view of the source element and OperationalStatus of the replica element may have a value of "Error" if a repair action is necessary. The provider may allow access to a replica in this state if indicated in HostAccessibleState[] of StorageReplicationCapabilities. The subprofile currently does not specify how to recover from "Broken" state. A ModifySynchronization Detach operation may be invoked to a replica in this state.

Values of the SyncMaintained and WhenSynced properties in a StorageSynchronized association are maintained as shown in the Table 203. The table does not apply to CopyType "UnSyncUnAssoc".

Table 203 - SyncMaintained and WhenSynced Properties

Synchronization State	SyncMaintained		WhenSynced	
	Sync/Async	UnSyncAssoc	Sync/Async	UnSyncAssoc
Initialized	True or False	True or False	Null	Date/Time frozen
Prepare In Progress	True or False	True or False	Null	Date/Time frozen
Prepared	True or False	True or False	Null	Date/Time frozen
Resync In Progress	True or False	True or False	Null	Date/Time frozen
Synchronized	True	Not specified	Null or D/T copy done	Null
Idle	Not specified	True or False	Null	Date/Time frozen
Quiesce In Progress	True or False	False	Null or D/T copy done	Null
Quiesced	True or False	False	Null or D/T copy done	Null
Fracture In Progress	True or False	Not specified	Null or D/T copy done	Null
Fractured	False	Not specified	Date/Time frozen	Null
Copy In Progress	Not specified	True or False	Null	Date/Time frozen
Frozen	Not specified	False	Null	Date/Time frozen
Restore In Progress	False	False	Date/Time frozen	Date/Time frozen
Broken	False	False	Null	Null

SyncMaintained "True" means that a copy engine is actively copying updated blocks from the source element to the target element. "False" means either the copy engine is stopped or copying the target to the source during "Restore In Progress" state. WhenSynced can contain two forms of a Date/Time value. A non-null value indicates either the date/time a frozen image is created or the date/time that the source element is completely copied to the target mirror

element. The Fracture, Resync and Restore operations for ModifySynchronization may cause the WhenSynced value to change.

10.1.13 State Transitions

10.1.13.1 General

Figure 52: "State Transitions for Mirrors and Clones" shows state transitions for mirrors and clones:

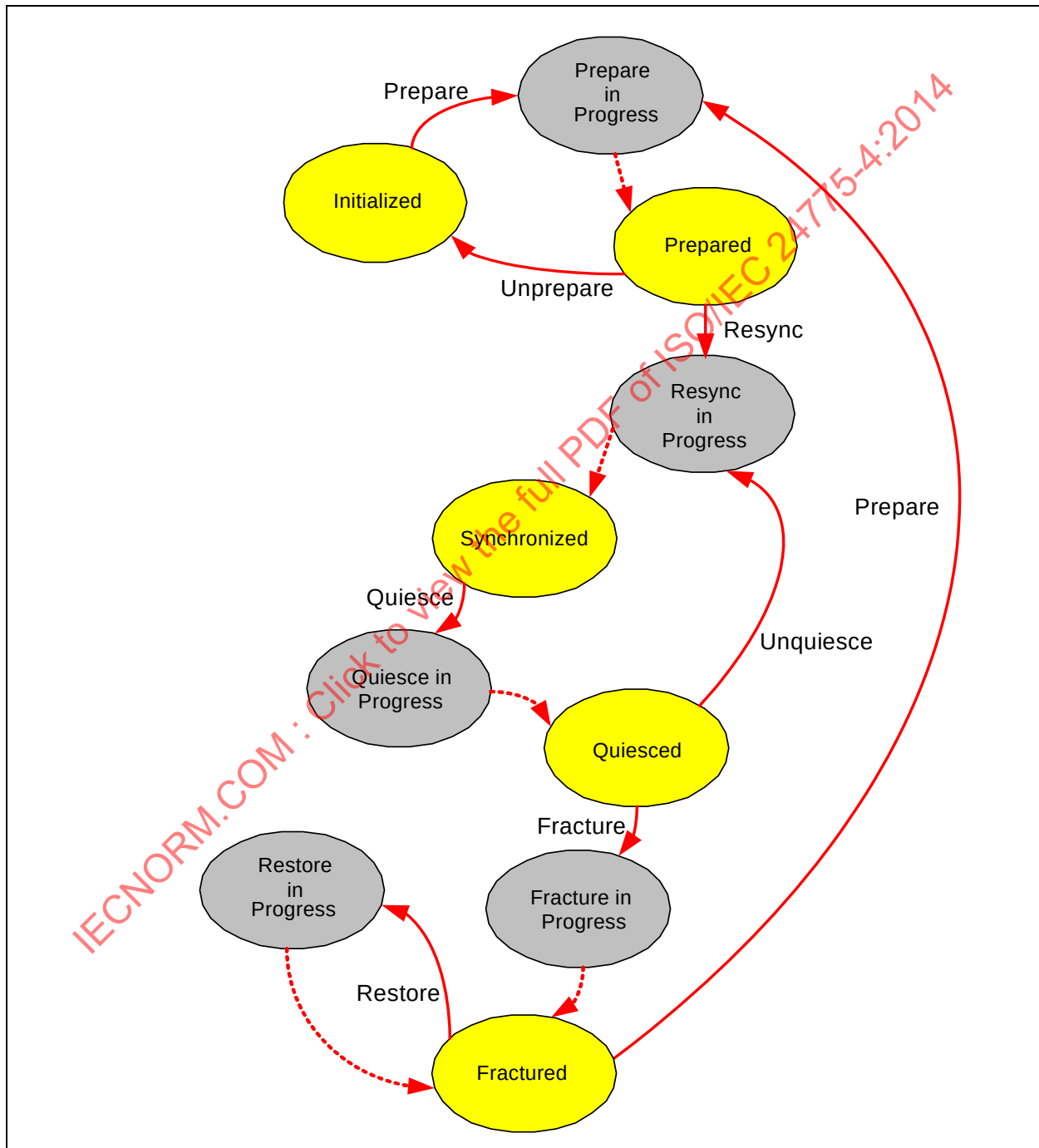


Figure 52 - State Transitions for Mirrors and Clones

Figure 53: "State Transitions for Snapshots and Migration" shows state transitions for snapshots:

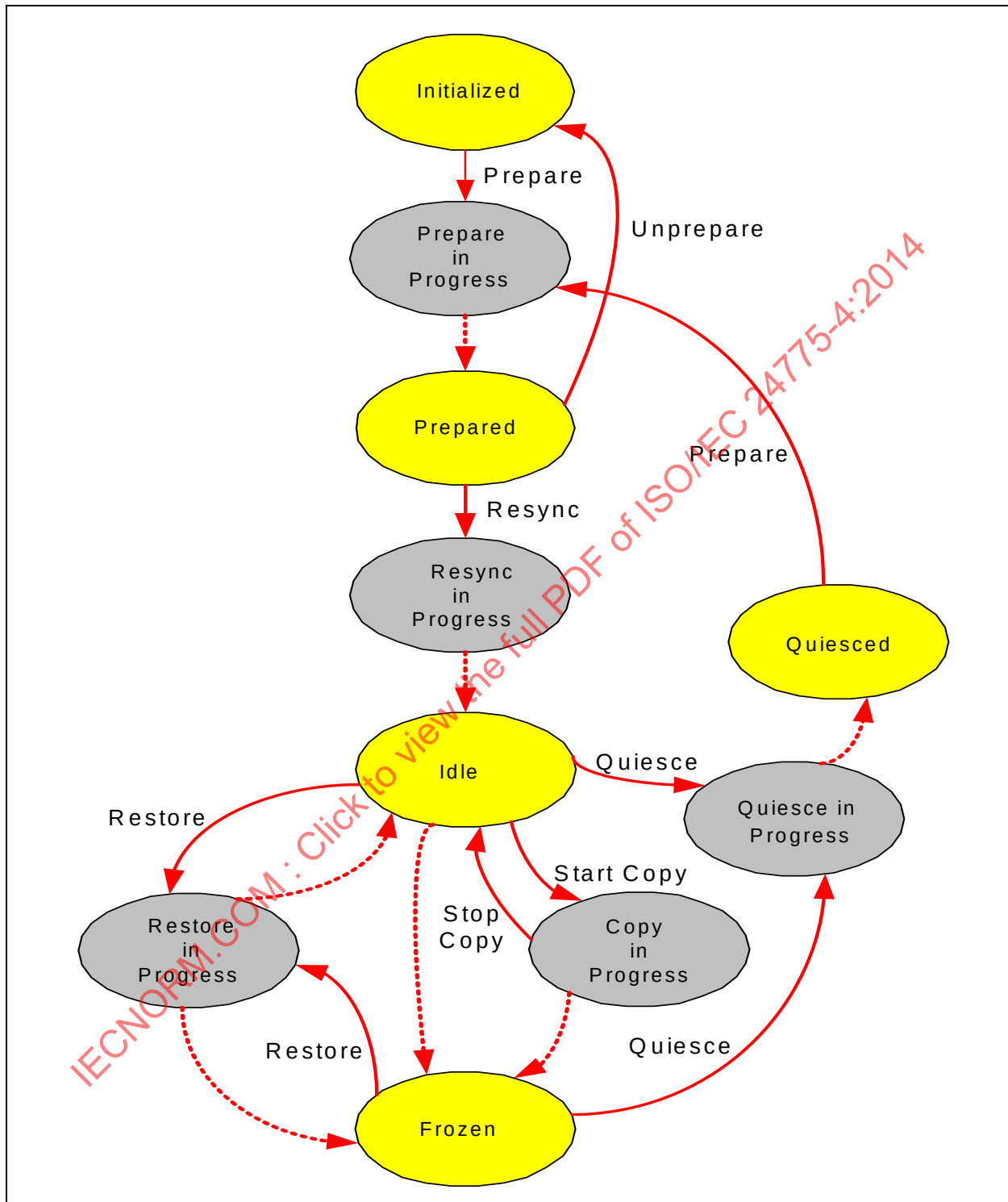


Figure 53 - State Transitions for Snapshots and Migration

The preceding state diagrams for mirrors and snapshots use the following conventions.

- The state diagram is entered when any of the three replica creation methods is invoked. Exit occurs when a ModifySynchronization Detach operation is invoked.
- A transition from a steady state to an in progress state is shown by a solid arrow line and is initiated by a ModifySynchronization operation other than Detach.

- An automatic transition from an in progress state to a steady state is shown by a dashed arrow line.
- Automatic exit occurs from an in progress state when cloning and migration operations have completed.

EXPERIMENTAL

Figure 54 shows the CopyState transitions. The dashed arrow lines represent automatic transitions. They transition unconditionally when the target element is ready to move to the next state. The solid arrow lines represent the transitions as the result of a requested operation (using, for example, ModifyReplicaSynchronization). The label of the solid arrow line indicates the requested operation.

The “create” methods normally start with the Initialized state. However, it is possible to use the WaitForCopyState parameter of the create method to force the CopyState to the Inactive or Prepared state after the initialization is complete. In this case, CopyState will remain in Inactive or Prepared state until a Modify method is used to Activate the synchronization.

IECNORM.COM : Click to view the full PDF of ISO/IEC 24775-4:2014

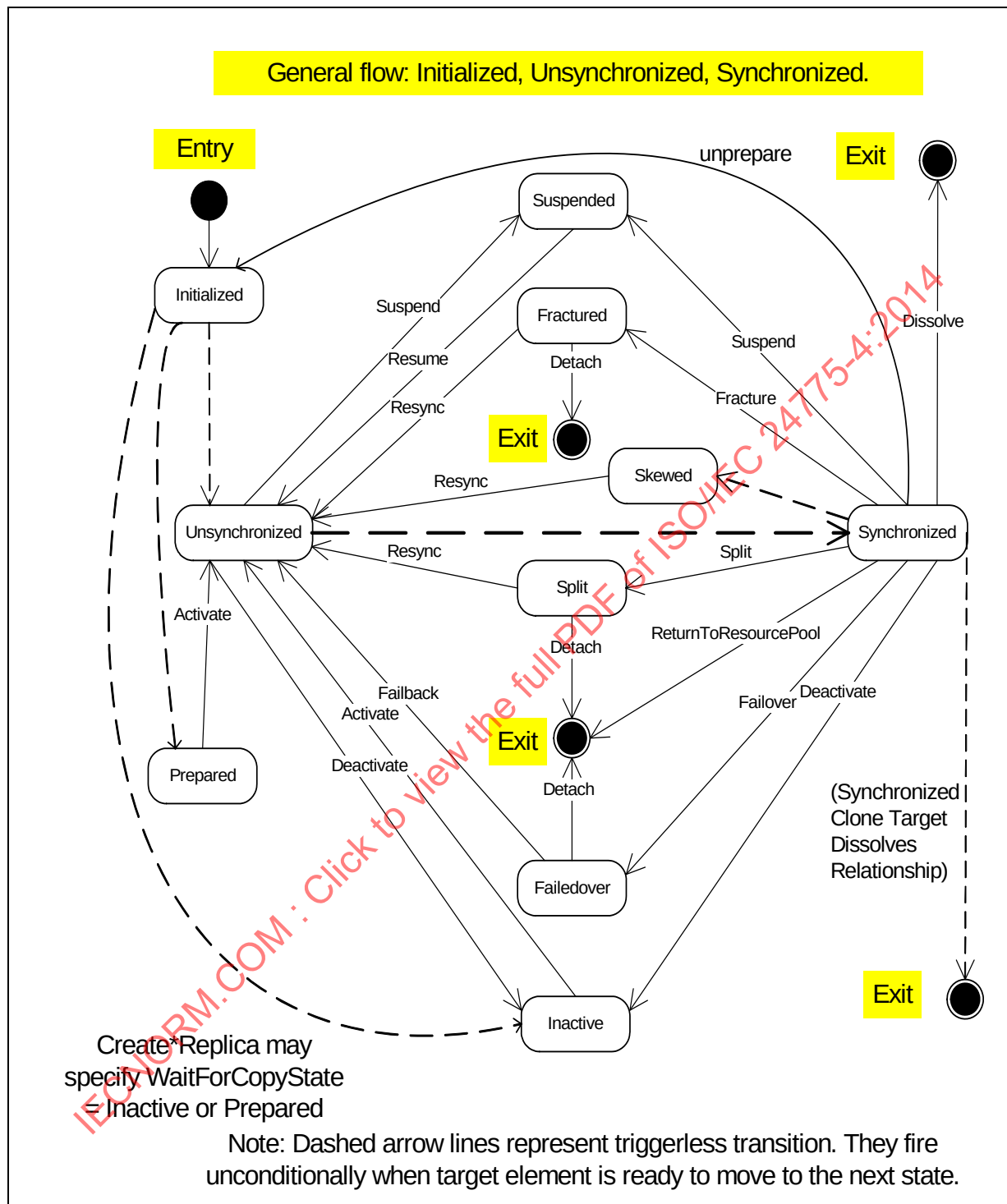


Figure 54 - CopyState Transitions

10.1.13.2 Alignment of State Transitions

Both SyncState and the combination of CopyState and ProgressStatus should be reported and the values need to be aligned. Table 199 addresses the basic alignment. This subclause provides more detail on the state transitions and how they would be coded for both SyncState and CopyState.

- CopyState="Initialized", ProgressStatus="Completed" (SyncState="PrepareInProgress")
If the InitialReplicationState="Initialized", then this state will exist. When the Initial state can be

Initialized, this is the state of a StorageSynchronized after it is created (or Unprepared). The association exists, but nothing is going on (WhenSynced=NULL). With ModifyReplicaSynchronization an Initialized association is automatically Prepared.

Note that it is also possible to get to the Initialized state by doing a ModifyReplicaSynchronization Unprepare operation. This puts the association back in the Initialized state (which is then automatically progressed to the next state).

From the Initialized state, the no ModifyReplicaSynchronization operations are supported.

- CopyState="Prepared", ProgressStatus="Completed" (SyncState="Prepared")
If the InitialReplicationState="Prepared" or an Initialized association has been successfully Prepared, then this state will exist. The association exists, but nothing is going on (WhenSynced=NULL), but it is enabled for a Resync operation.
From the "Prepared" state the only operation supported is Activate. This is represented by:
 - CopyState="Prepared" and ProgressStatus="Synchronizing" (SyncState="ResyncInProgress")
- CopyState="Unsynchronized", ProgressStatus="Synchronizing" (SyncState="ResyncInProgress")
This CopyState is equivalent to a SyncState of "ResyncInProgress". From the "Synchronized" state the only operations supported are Suspend and Deactivate. How this gets reported as SyncState depends on how the CopyState was achieved.
 - With Suspend: When a client uses ModifyReplicaSynchronization with an Operation of "Suspend" the association changes to CopyState="Unsynchronized" with ProgressStatus="Suspending". The SyncState should be set to "QuiesceInProgress".
 - With Deactivate: When a client uses ModifyReplicaSynchronization with an Operation of "Deactivate" the association changes to CopyState="Unsynchronized" with ProgressStatus="Dormant". The SyncState should be set to "QuiesceInProgress".
- CopyState="Synchronized", ProgressStatus="Completed" (SyncState="Synchronized" or "Idle")
The CopyState of "Synchronized" is an automatic transition from the Unsynchronized state. For mirrors, then an implementation should report SyncState="Synchronized". For snapshots, the implementation should report SyncState="Idle". From the "Synchronized" state the operations supported are: Suspend, Fracture, Split, Failover, Deactivate, Unprepare and Dissolve.
 - With Suspend: When a client uses ModifyReplicaSynchronization with an Operation of "Suspend" the association changes to CopyState="Synchronized" with ProgressStatus="Suspending". The SyncState should be set to "QuiesceInProgress".
 - With Fracture: When a client uses ModifyReplicaSynchronization with an Operation of "Fracture" the association changes to CopyState="Synchronized" with ProgressStatus="Fracturing". The SyncState should be set to "Fracture In Progress".
 - With Split: When a client uses ModifyReplicaSynchronization with an Operation of "Split" the association changes to CopyState="Synchronized" with ProgressStatus="Splitting". The SyncState should be set to "Fracture In Progress".
 - With Failover: When a client uses ModifyReplicaSynchronization with an Operation of "Failover" the association changes to CopyState="Synchronized" with ProgressStatus="Failing over". The SyncState should be set to "Restore In Progress".
 - With Deactivate: When a client uses ModifyReplicaSynchronization with an Operation of "Deactivate" the association changes to CopyState="Synchronized" with ProgressStatus="Dormant". The SyncState should be set to "QuiesceInProgress".
 - With Unprepare: When a client uses ModifyReplicaSynchronization with an Operation of "Unprepare" the association changes to CopyState="Synchronized" with ProgressStatus="Initializing". The SyncState should be set to "Initialized".
 - With Dissolve: The StorageSynchronized is deleted.
- CopyState="Fractured", ProgressStatus="Completed" (SyncState="Fractured")
This CopyState is equivalent to a SyncState of "Fractured". From the "Fractured" state the only operations supported are: Resync and Detach.

- With Detach: The StorageSynchronized is deleted.
- With Resync: When a client uses ModifyReplicaSynchronization with an Operation of "Resync" the association changes to CopyState="Fractured" with ProgressStatus="Resyncing". The SyncState should be set to "ResyncInProgress".
- CopyState="Split", ProgressStatus="Completed" (SyncState="Fractured")
This CopyState is equivalent to a SyncState of "Fractured". From the "Split" state the only operations supported are: Resync and Detach.
 - With Detach: The StorageSynchronized is deleted.
 - With Resync: When a client uses ModifyReplicaSynchronization with an Operation of "Resync" the association changes to CopyState="Split" with ProgressStatus="Resyncing". The SyncState should be set to "ResyncInProgress".
- CopyState="Suspended", ProgressStatus="Completed" (SyncState="Quiesced")
This CopyState is equivalent to a SyncState of "Quiesced". From the "Suspended" state the only operation supported is: Resume.
 - With Resume: When a client uses ModifyReplicaSynchronization with an Operation of "Resume" the association changes to CopyState="Suspended" with ProgressStatus="Resyncing". The SyncState should be set to "ResyncInProgress".
- CopyState="Broken", ProgressStatus="Not Applicable" (SyncState="Broken")
This CopyState is equivalent to a SyncState of "Broken". From the "Broken" state the only operation supported is Activate. Repair work shall be done. When this is done, the association is put in the "Inactive" state.
 - With Activate: When a client uses ModifyReplicaSynchronization with an Operation of "Activate" the association changes to CopyState="Inactive" with ProgressStatus="Resyncing". The SyncState should be set to "ResyncInProgress".
- CopyState="Aborted", ProgressStatus="Completed" (SyncState="Idle" for snapshots and "Quiesced" for mirrors)
This CopyState is equivalent to a SyncState of "Idle" for snapshots and "Quiesced" for mirrors. From the "Aborted" state the only operation supported is Activate.
 - With Activate: When a client uses ModifyReplicaSynchronization with an Operation of "Activate" the association changes to CopyState="Aborted" with ProgressStatus="Resyncing". The SyncState should be set to "ResyncInProgress".
- CopyState="Failedover", ProgressStatus="Completed" (SyncState="Frozen" for snapshots and "Fractured" for mirrors)
This CopyState is equivalent to a SyncState of "Frozen" for snapshots and "Fractured" for mirrors. From the "Failedover" state the only operations supported are: Failback and Detach.
 - With Failback: When a client uses ModifyReplicaSynchronization with an Operation of "Failback" the association changes to CopyState="Synchronized" with ProgressStatus="Failing back". The SyncState should be set to "Restore In Progress".
 - With Detach: The association is deleted.
- CopyState="Inactive", ProgressStatus="Completed" (SyncState="Idle" for snapshots and "Quiesced" for mirrors)
This CopyState is equivalent to a SyncState of "Idle" for snapshots and "Quiesced" for mirrors. From the "Inactive" state the only operation supported is: Activate.
 - With Activate: When a client uses ModifyReplicaSynchronization with an Operation of "Activate" the association changes to CopyState="Inactive" with ProgressStatus="Resyncing". The SyncState should be set to "ResyncInProgress".
- CopyState="Skewed", ProgressStatus="Completed" (SyncState="Initialized")
This CopyState is equivalent to a SyncState of "Initialized". That is, the association exists, but

nothing else can be said about it. From the "Skewed" state the only operation supported is: Resync.

- With Resync: When a client uses ModifyReplicaSynchronization with an Operation of "Resync" the association changes to CopyState="Skewed" with ProgressStatus="Resyncing". The SyncState should be set to "ResyncInProgress". Note that with ModifyReplicaSynchronization, Prepare is automatic.
- CopyState="Mixed", ProgressStatus="Completed"
The mixed state only applies to group operations and should never show up on single source-target pairs.

Using the deprecated method ModifySynchronization, the SyncStates that are effected also need to be reported in the CopyState and ProgressStatus properties. This is summarized by the following bullets:

- SyncState="Initialized" (CopyState="Initialized", ProgressStatus="Completed")
This state would only exist if InitialReplicationState="Initialized" or an ModifySynchronization Unprepare operation is issued. The only ModifySynchronization operation supported is Prepare.
- With Prepare: When a client uses ModifySynchronization with an Operation of "Prepare" the association changes to SyncState="PrepareInProgress". This should be reported as CopyState="Initialized" with ProgressStatus="Preparing".
- SyncState="Prepared" (CopyState="Prepared", ProgressStatus="Completed")
The only ModifySynchronization operations supported are Resync or Unprepare.
- With Resync: When a client uses ModifySynchronization with an Operation of "Resync" the association changes to SyncState="ResyncInProgress". This should be reported as CopyState="Prepared" with ProgressStatus="Synchronizing".
- With Unprepare: When a client uses ModifySynchronization with an Operation of "Unprepare" the association changes to SyncState="Initialized". This should be reported as CopyState="Initialized" with ProgressStatus="Completed".
- SyncState="Synchronized" (CopyState="Synchronized", ProgressStatus="Completed")
This state only applies to mirrors. The only ModifySynchronization operation supported is Quiesce.
- With Quiesce: When a client uses ModifySynchronization with an Operation of "Quiesce" the association changes to SyncState="QuiesceInProgress". This should be reported as CopyState="Synchronized" with ProgressStatus="Dormant".
- SyncState="Quiesced" (CopyState="Suspended", ProgressStatus="Completed")
The only ModifySynchronization operations supported are Fracture and Unquiesce for mirrors and Prepare for snapshots.
- With Fracture: When a client uses ModifySynchronization with an Operation of "Fracture" the association changes to SyncState="FractureInProgress". This should be reported as CopyState="Suspended" with ProgressStatus="Fracturing".
- With Unquiesce: When a client uses ModifySynchronization with an Operation of "Unquiesce" the association changes to SyncState="ResyncInProgress". This should be reported as CopyState="Suspended" with ProgressStatus="Resyncing".
- With Prepare: When a client uses ModifySynchronization with an Operation of "Prepare" the association changes to SyncState="PrepareInProgress". This should be reported as CopyState="Suspended" with ProgressStatus="Preparing".
- SyncState="Restore In Progress" (CopyState="Synchronized", ProgressStatus="Failing over")
- SyncState="Idle" (CopyState="Inactive", ProgressStatus="Completed")
This state only applies to snapshots. The only ModifySynchronization operations supported are Quiesce, Start Copy and Restore.

- With Quiesce: When a client uses ModifySynchronization with an Operation of "Quiesce" the association changes to SyncState="QuiesceInProgress". This should be reported as CopyState="Inactive" with ProgressStatus="Dormant".
- With Start Copy: When a client uses ModifySynchronization with an Operation of "Start Copy" the association changes to SyncState="Copy In Progress". This should be reported as CopyState="Inactive" with ProgressStatus="Synchronizing". Note that this is a background copy.
- With Restore: When a client uses ModifySynchronization with an Operation of "Restore" the association changes to SyncState="Restore In Progress". This should be reported as CopyState="Inactive" with ProgressStatus="Failing over".
- SyncState="Broken" (CopyState="Broken", ProgressStatus="Completed")
A broken association needs to be repaired. After the relationship is repaired, the association goes into its InitialReplicationState.
- SyncState="Fractured" (CopyState="Fractured", ProgressStatus="Completed")
This state only applies to mirrors. The only ModifySynchronization operations supported are Prepare and Restore.
 - With Prepare: When a client uses ModifySynchronization with an Operation of "Prepare" the association changes to SyncState="PrepareInProgress". This should be reported as CopyState="Fractured" with ProgressStatus="Preparing".
 - With Restore: When a client uses ModifySynchronization with an Operation of "Restore" the association changes to SyncState="Restore In Progress". This should be reported as CopyState="Fractured" with ProgressStatus="Failing over".
- SyncState="Frozen" (CopyState="Synchronized", ProgressStatus="Completed")
This state only applies to snapshots. The only ModifySynchronization operations supported are Quiesce and Restore.
 - With Quiesce: When a client uses ModifySynchronization with an Operation of "Quiesce" the association changes to SyncState="QuiesceInProgress". This should be reported as CopyState="Synchronized" with ProgressStatus="Dormant".
 - With Restore: When a client uses ModifySynchronization with an Operation of "Restore" the association changes to SyncState="Restore In Progress". This should be reported as CopyState="Synchronized" with ProgressStatus="Failing over".

10.1.13.3 Synchronized SyncState

Synchronized state for the Mirror and Clone SyncTypes indicates all data has been copied from the source element to the target element. For the Snapshot SyncType, because the target element is a virtual point-in-time view of the source element, the Synchronized CopyState indicates all the metadata (pointers) for the snapshot have been created. Synchronization for the snapshots is achieved relatively quickly.

Figure 55 shows a sampling of the CopyState transitions and the corresponding ProgressStatus changes. In a steady state condition, for example, the CopyState has a value of "Synchronized", and at the same time the ProgressStatus has a value of "Completed".

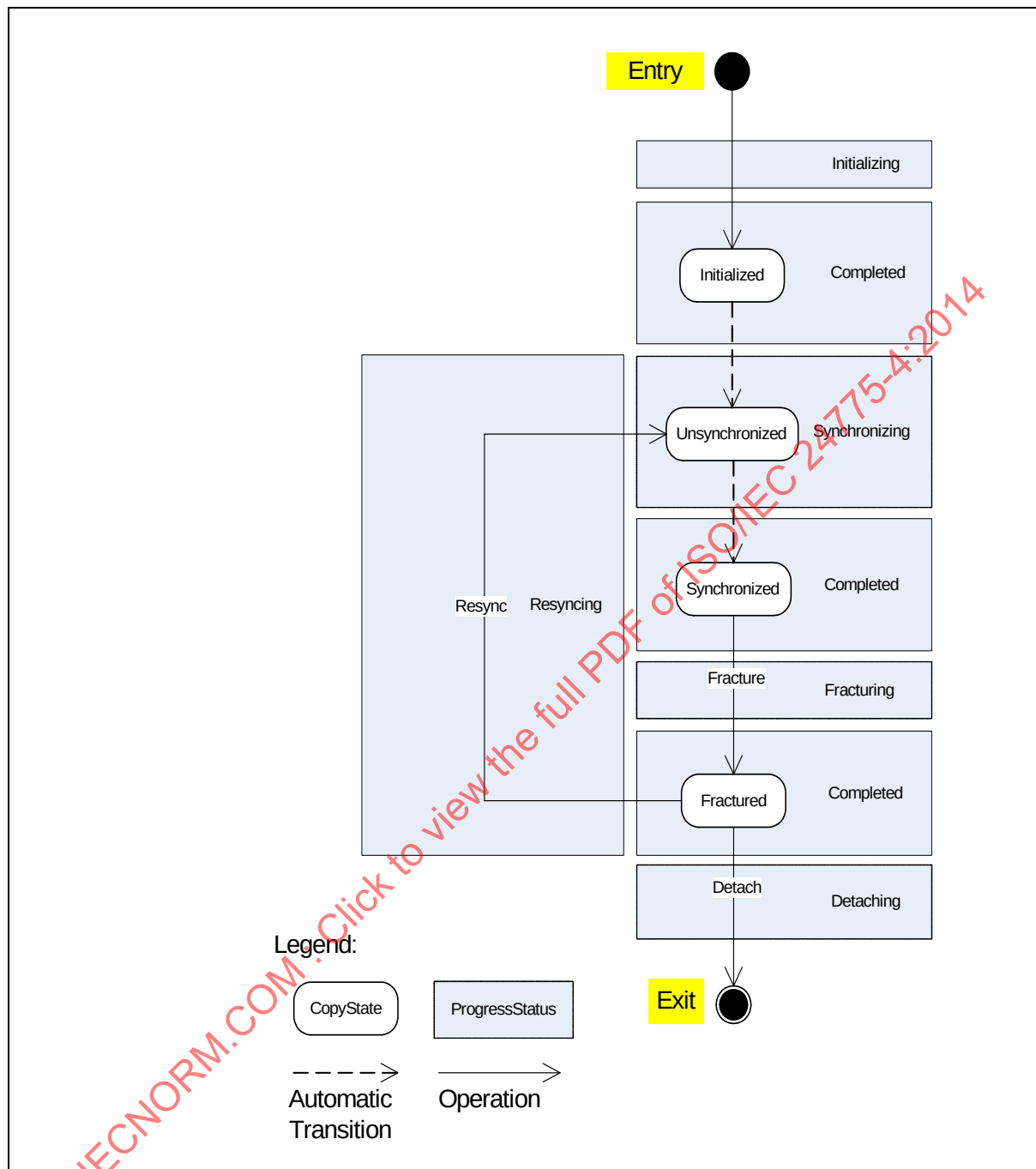


Figure 55 - Sample CopyState and ProgressStatus Transitions

10.1.14 Accessibility to Associations and Elements

There are two cases that should be considered:

Case 1: The method completes successfully without returning a job. The created replication association (StorageSynchronized for Mirror and Snapshot copy types) and the newly created target element shall be accessible. The StorageSynchronized association between source and target elements for the Clone copy type may not be accessible after synchronization is achieved. However, there will be a SettingsDefineState association (if supported) between the newly copied target element and a SynchronizationAspect instance.

Case 2: The method returns the status of "Job Started". The AffectedJobElement association associates the concrete job to the target element, unless there is no target element such as

CreateSynchronizationAspect or when the target element is deleted (ReturnToStoragePool). In this case, the AffectedJobElement points to the source element. To ensure the replication association is accessible, the CopyState of the association has to have at least reached the Initialized state. To guarantee accessibility to associations and elements, specify the WaitForCopyState when issuing the method CreateElementReplica.

EXPERIMENTAL

10.1.15 Host Access Restrictions

The Copy Services Subprofile does not provide any services for managing access to replicas. However, replication services often restrict access to replicas for the following reasons:

- 1) Replicas have the same volume signature as their source element. Exposing both the source and replica to the same host may cause problems with a duplicate volume signature.
- 2) Delta replicas created by embedded software elements such as a volume manager may be unavailable for export to a secondary host.

The subprofile uses two properties in StorageReplicationCapabilities to indicate host access restrictions:

- 1) ReplicaHostAccessibility
- 2) HostAccessibleState[]

A provider may set values for these two properties indicating any host access restrictions imposed on replicas. These restrictions apply to all replicas created with the same CopyType value. Access control for a specific replica by a specific host is normally managed using services described in Clause 19 Masking and Mapping Subprofile.

EXPERIMENTAL

Generally, exposing both the source and replica to the same host may cause problems due to a duplicate volume signature. At a minimum, the signature of a replica shall be changed before the replica is exposed to the same host as the source element.

Managing host access to source and target elements can be managed by using services described in Clause 19 Masking and Mapping Subprofile.

The method ReplicationServiceCapabilities.GetSupportedCopyStates for each CopyState additionally returns information as to whether a replica is host accessible (boolean) for the given CopyState.

EXPERIMENTAL

EXPERIMENTAL

10.1.16 Settings, Specialized Elements and Pools for Replicas

A copy services provider shall support StorageSetting with the additional properties defined to manage replica elements and replication operations. These properties are listed in the definition of StorageSetting in this subprofile. This definition extends the basic list of required StorageSetting properties listed in the Block Services Package. The CreateSetting method should return a REF to a StorageSetting instance with all of the replication properties initialized to values consistent with the capabilities indicated in StorageReplicationCapabilities. Many replication properties allow an initial value of "not applicable" if the provider does not use the property. The provider sets the value lists for the SupportedStorageElementUsage[] and

SupportedStoragePoolUsage[] properties in StorageConfigurationCapabilities to indicate which values of StorageSetting.StorageExtentInitialUsage and StorageSetting.StoragePoolInitialUsage are supported by the provider.

A provider may require specialized pools to contain delta replicas, specialized elements as replica targets and specialized elements as concrete components for delta replica pools. The provider may require the client to manage creation of these specialized elements – this is explained in detail in 10.6 "Client Considerations and Recipes". Alternatively, the provider may automatically create specialized pools and elements and make them available for discovery by clients. In either case, the StorageExtentInitialUsage and StoragePoolInitialUsage properties in StorageSetting shall be supported by the provider as part of the goal parameter for pool/element creation methods.

Elements and pools specialized for Copy Services are located using the GetElementsBasedOnUsage method described in Clause 6 Block Services Package.

When StorageExtentInitialUsage or StoragePoolInitialUsage is set in the goal parameter for an element or pool creation method, the value acts as an additional parameter indicating a specialized element. The provider ensures that the required element type is created and the Usage property value is set in the new replica element or pool. Certain types of specialized replica elements can be provided by changing existing elements using the RequestUsageChange method. The ClientSettableElementUsage[] value list indicates the allowable modifications for a storage element and the ClientSettablePoolUsage[] value list indicates the allowable modifications for a storage pool.

EXPERIMENTAL

10.1.17 Backward Compatibility

A copy services provider can maintain backward compatibility with a 1.0 copy services client. The following conditions are necessary for backward compatibility:

- 1) The instance of StorageConfigurationCapabilities should set replication capability property values in the same way indicated for a 1.0 copy services provider. A newer copy services client should ignore these properties and use StorageReplicationCapabilities instead.

EXPERIMENTAL

- 2) The provider should treat AttachReplica as an alias for CreateElementReplica.

EXPERIMENTAL

- 3) The provider should treat StorageSynchronized.SyncState values "Synchronized" and "Idle" as equivalent for CopyType "UnSyncAssoc".

10.1.18 Mutually Exclusive Capabilities

Both StorageReplicationCapabilities and StorageConfigurationCapabilities contain the SupportedSynchronousActions[] and SupportedAsynchronousActions[] properties. The provider shall not include the value corresponding to an action in both properties. An action can run synchronously or asynchronously but not both. An action indicated in one of the StorageConfigurationCapabilities properties shall also be indicated in a corresponding instance of StorageReplicationCapabilities.

EXPERIMENTAL

10.1.19 Deleting the Target Elements

Mirror, Clone, and Snapshot target elements that are no longer in a synchronization association are deleted using the `StorageConfigurationService.ReturnToStoragePool` method. However, the Snapshot target elements that are in a synchronization association are deleted using the `ReplicationService.ModifyReplicaSynchronization` (or `ModifySynchronization`) method with the "Return To ResourcePool" operation parameter, which also removes the synchronization association.

10.1.20 Using StorageSettings for Replicas

The `StorageSetting` class has several properties used to create and manage replicas. Instances of this class are used as the goal parameter for the methods of this profile. The extrinsic method `CIM_StorageCapabilities.CreateSetting` is used to create a setting and the intrinsic method `ModifyInstance` is used to adjust the properties of a created `StorageSetting`. See Clause 6 "Block Services Package" for the details of creating and modifying a storage setting.

10.1.21 Finding and Creating Target Elements

The extrinsic method `ReplicationService.GetAvailableTargetElements` is used to locate the available target elements for a given source and copy type. The implementation may also support creating target elements if the appropriate target elements are not supplied and/or are not available. The implementation may require the client to create specialized elements to be used as a target of a copy operation. The specialized elements have a specific values in their `Usage` property. Certain types of specialized elements can be provided by changing the `Usage` property of existing elements. Refer to Clause 6 "Block Services Package" for creating (specialized) elements and modifying the `Usage` value of existing elements.

Refer to 10.5.2.4.8 "GetDefaultReplicationSettingData" and 10.5.2.4.4 "GetSupportedFeatures" to determine if the implementation automatically creates target elements, and if specialized elements are required for the desired `SyncType`.

10.1.22 Using StoragePools for Replicas

10.1.22.1 General

Replicas are allocated from storage pools. The implementation may require specialized storage pools to contain delta replicas (changed tracks of snapshots) or the "write intent log" files. The specialized storage pools have a specific value in their `Usage` property, for example, "Reserved as a Delta Replica Container", "Reserved for Local Replication Services", or "Reserved for Remote Replication Services".

10.1.22.2 Delta Replica StoragePools

Depending on the implementation, the Snapshot targets may require a fixed space consumption or variable space consumption. Refer to 10.5.2.4.4 "GetSupportedFeatures" to determine if specialized storage pool is required.

There are three types of delta replica pool access:

- "Any" - specialized storage pools are not required for delta replicas. The implementation creates delta replicas based on the fixed space consumption model and the client can select any storage pool as a container.
- "Shared" - a single shared storage pool is the container for all delta replicas. This type of storage pool is always preexisting and may be located with the `GetElementBasedOnUsage` method. The client may need to add space to this type of storage pool.

- “Exclusive” - each source element requires an exclusive, special storage pool for associated delta replicas. If the storage pool already exists, it is associated to the source element with a `ReplicaPoolForStorage` association. If the storage pool does not exist, the client creates the storage pool.

“Multiple” - “multiple specialized, exclusive pools may exist or may be created.”

Figure 56 and Figure 57 show the fixed and variable space consumption for the Snapshot targets, respectively. If the implementation supports fixed space consumption, the `DeltaReservation` properties are set by the client to the appropriate values for a new snapshot. The values are set in the associated `StorageSetting` element to be passed as a goal parameter to the `CreateElementReplica` method (or `CreateSynchronizationAspect` method). For variable space consumption, there are no special properties to set by the client.

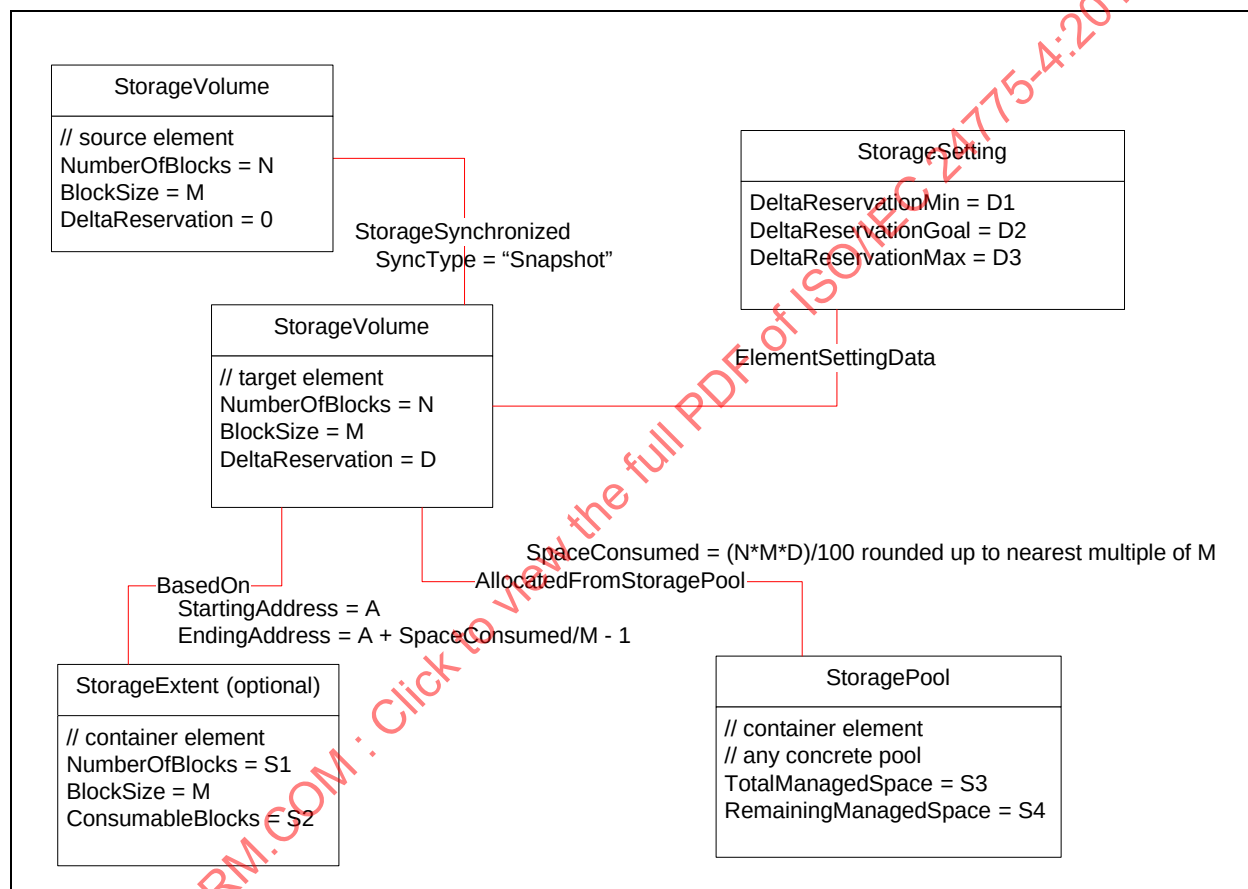


Figure 56 - Fixed Space Consumption

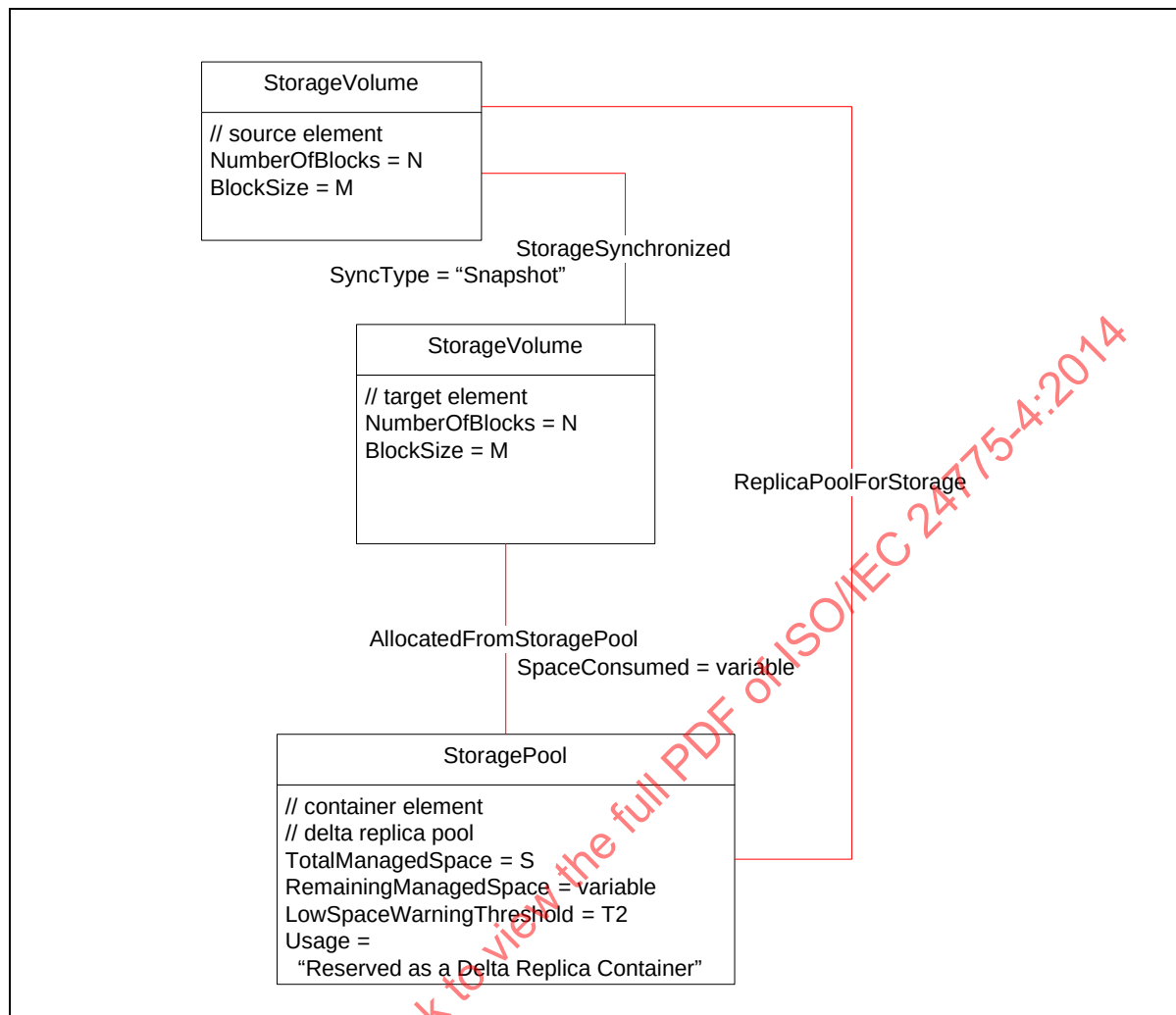


Figure 57 - Variable Space Consumption

10.1.23 Thinly Provisioned Elements

Replication Services supports “copying” thinly provisioned elements. Depending on the underlying implementation, it is possible to copy a thinly provisioned source element to a thinly provisioned target element or alternatively to a fully provisioned target element. Other combinations may be advertised in the capabilities.

If an implementation supports more than one combination of source and target provisioning, clients may use the ReplicationSettingData parameter of the CreateElementReplica to request a specific combination.

Refer to the capabilities for the allowable combinations supported by the implementation. See 10.5.2.4.7, 10.7.17, and 10.5.2.4.8.

10.1.24 Indication Events

10.1.24.1 General

Depending on the implementation, the Copy Services Profile generates a number of different alert and life cycle indications, shown in Table 204. Clients decide what indications they wish to receive by subscribing to the appropriate indications.

Table 204 - Indications

Indication	Source Of
CIM_InstCreation	• New Job Creation • New Target Element Creation • New StorageSynchronized Association Creation
CIM_InstDeletion	• Job Deletion • Target Element Deletion (e.g. Snapshot) • StorageSynchronized Association Deletion
CIM_InstModification	• Job Progress and Status Changes • Source and Target Elements Status Changes • SyncState Changes • ProgressStatus Changes
CIM_AlertIndication	• StoragePool space consumption Alerts (especially by Snapshot targets). • Error conditions, such as: • StorageSynchronized State set to <i>Broken</i>.

EXPERIMENTAL

10.1.24.2 InstCreation on StorageSynchronized

This indication is triggered by any event that causes a StorageSynchronized association to be created. This includes use of methods such as CreateElementReplica. But it may also be triggered by other (external) events.

This indication is required of any conforming implementation of Copy Services.

10.1.24.3 InstDeletion on StorageSynchronized

This indication is triggered by any event that causes a StorageSynchronized association to be deleted. This includes use of methods such as ModifyReplicaSynchronization with the “Detach” operation. But it may also be triggered by other (external) events.

This indication is required of any conforming implementation of Copy Services.

DEPRECATED

10.1.24.4 InstModification on SyncState

This indication is triggered by any event that causes a SyncState change in any StorageSynchronized association. This includes use of methods such as ModifyReplicaSynchronization. But it may also be triggered by other (external) events.

This indication is required of any conforming implementation of Copy Services.

This Indication is being deprecated in favor of the “qualified” InstModification on Copy State (see 10.1.24.5).

DEPRECATED

EXPERIMENTAL

10.1.24.5 Qualified InstDeletion on StorageSynchronized

This indication is triggered by any event that causes a specific client defined StorageSynchronized association to be deleted. This includes use of methods such as ModifyReplicaSynchronization with the “Detach” operation. But it may also be triggered by other (external) events.

This indication may be supported by any conforming implementation of Copy Services.

10.1.24.6 Qualified InstModification on CopyState

This indication is triggered by any event that causes a CopyState change in a specific client defined StorageSynchronized association. This includes use of methods such as ModifyReplicaSynchronization. But it may also be triggered by other (external) events.

This indication may be supported by any conforming implementation of Copy Services.

10.1.24.7 Qualified InstModification on ProgressStatus

This indication is triggered by any event that causes a ProgressStatus change in a specific client defined StorageSynchronized association. This includes use of methods such as ModifyReplicaSynchronization. But it may also be triggered by other (external) events.

This indication may be supported by any conforming implementation of Copy Services.

10.1.24.8 InstModification on ProgressStatus

This indication is triggered by any event that causes a ProgressStatus change in any StorageSynchronized association. This includes use of methods such as ModifyReplicaSynchronization. But it may also be triggered by other (external) events.

This indication may be supported by any conforming implementation of Copy Services.

10.1.24.9 AlertIndication on StorageSynchronized

This indication is triggered by any event that causes a CopyState change to “broken” in any StorageSynchronized association. This is typically triggered by an external event.

This indication may be supported by any conforming implementation of Copy Services.

10.1.24.10 AlertIndication on StoragePool

This indication is triggered by any event that causes the remaining space in any StoragePool to dip below its warning threshold. This could be triggered by any one of a number of events.

This indication may be supported by any conforming implementation of Copy Services.

EXPERIMENTAL

10.2 Health and Fault Management Considerations

10.2.1 Health Indications

Certain capabilities of the subprofile use alert, instance modification and instance deletion indications for health and fault management. In general, instance modification indications when the OperationalStatus values of a replica element change may indicate a fault. Instance

modification indications when `StorageSynchronized.SyncState` automatically changes from any other value to "Broken" indicates a fault. If delta replica pools are supported with warning thresholds, alert indications may be generated by the provider when remaining space in a pool falls below a warning threshold or is completely consumed. The information in the alert indications is described in Table 205, "Copy Services Alert Indications".

EXPERIMENTAL

The Copy Services Subprofile generates alert indications, shown in Table 205, that allow monitoring of dynamic space consumption by delta replica elements. All of the alert indications indicate an `AlertType` value of "Device Alert" and an `OwingEntity` value of "SNIA". Alerts are generated for `CIM_StoragePool` elements to indicate that remaining consumable space is below a warning threshold percentage of total space or that all space in the pool has been consumed. The `LowSpaceWarningThreshold`, `TotalManagedSpace` and `RemainingManagedSpace` properties can be analyzed to determine an appropriate response.

Table 205 - Copy Services Alert Indications

AlertingManaged Element	PerceivedSeverity	ProbableCause	ProbableCauseDescription
Storage pool	Minor (4)	Threshold Crossed (52)	Pool at low space warning threshold: $\text{RemainingManagedSpace} / \text{TotalManagedSpace}$
Storage pool	Major (5)	Out of Memory (33)	No remaining space in storage pool

EXPERIMENTAL

EXPERIMENTAL

The profile uses indications to report health and fault management. In general, instance modification indications are sent when changes in `OperationalStatus` and `HealthState` values of the following instances indicate a fault condition:

- Source and Replica elements

In response to a fault indication, clients can follow the `RelatedElementCausingError` association between the instance reporting the error and the faulted component.

The profile also generates alert indications when the `CopyState` of a replication association transitions to the Broken state.

The profile generates alert indications that allow monitoring of storage pool consumption by the replica elements.

EXPERIMENTAL

10.2.2 Replication Error Messages

DEPRECATED

10.2.2.1 Storage Configuration Service Method Messages

The Copy Services Subprofile returns the error responses listed in Table 206 for the extrinsic methods supported by the subprofile. The subprofile uses MessageID values defined in the common error registry and the storage error registry.

Table 206 - Copy Services Error Responses

MessageID	Message Name
MP2	Operation Not Supported
MP3	Property Not Found
MP5	Parameter Error
MP11	Too Busy To Respond
MP17	Invalid Property Combination During Instance Modification
DRM20	Invalid Extent Passed
DRM24	Invalid State Transition
DRM25	Invalid SAP For Method
DRM26	Resource Not Available
DRM27	Resource Limit Exceeded

DEPRECATED

EXPERIMENTAL

10.2.2.2 Replication Service Method Messages

Not defined in this International Standard.

EXPERIMENTAL

10.3 Cascading Considerations

Not defined in this International Standard.

10.4 Supported Subprofiles and Packages

See 10.1.1 "Synopsis".

The Block Services Subprofile is a mandatory prerequisite for the Copy Services Subprofile. Clients require methods and recipes from block services for the following purposes:

- Identify replica target candidates
- Identify extents and pools to be used as replica containers
- Create and delete replica container elements
- Create and delete replica target elements
- Create generated setting objects with additional properties required by the copy services subprofile.

Many classes and methods defined in Block Services are used in Copy Services without extensions or additional properties. In this case, the classes and methods are not redefined in Copy Services.

The Job Control Subprofile is required if any of the copy services extrinsic methods run asynchronously with created job elements.

Copy services defines instance indications and alert indications using required and optional properties described in clause 43 Indication Profile of ISO/IEC 24775-3:2014.

10.5 Methods of the Profile

10.5.1 Intrinsic Methods of the Profile

The subprofile requires the provider to support the CreateInstance, GetInstance, ModifyInstance and DeleteInstance intrinsic methods for certain optional capabilities of the subprofile.

10.5.2 Extrinsic Methods of the Profile

EXPERIMENTAL

10.5.2.1 Block Services Package

The profile is dependent on other extrinsic methods provided by the Block Services Package for storage pool and storage element manipulations.

EXPERIMENTAL

DEPRECATED

10.5.2.2 StorageConfigurationService Methods

10.5.2.2.1 General

The Copy Services Subprofile is dependent on many of the extrinsic methods provided by block services. The ReturnToStoragePool extrinsic method defined by block services is used to delete a replica element. ReturnToStoragePool may receive an MP3 (property not found) error response for replica elements that are implicitly deleted by a ModifySynchronization Detach operation.

All of the subprofile methods return one of three status codes or return an error response. The supported status codes are:

- 0: Job completed with no error
- 1: Method not supported
- 0x1000: Job started

Table 207 summarizes the extrinsic methods for replica creation and management in the StorageConfigurationService.

Table 207 - Extrinsic Methods of StorageConfigurationService

Method	Described in
ModifySynchronization()	Table 208, "ModifySynchronization"
CreateReplica()	Table 209, "CreateReplica Method"
AttachReplica()	Not documented

10.5.2.2.2 ModifySynchronization Method

Table 208 lists and describes the ModifySynchronization Method.

Table 208 - ModifySynchronization

Method: ModifySynchronization			
Errors: DRM24, MP2, DRM25			
Parameters:			
Qualifiers	Name	Type	Description/Values
IN, REQ	Operation	uint16	Type of operation to modify the replica: 2: Detach 3: Fracture 4: Resync 5: Restore 6: Prepare 7: Unprepare 8: Quiesce 9: Unquiesce 10: Reset to Sync 11: Reset to Async 12: Start Copy 13: Stop Copy
OUT	Job	ConcreteJob REF	Returned if job started.
IN, REQ	Synchronization	StorageSynchronized REF	Association to replica that is modified

"Detach" operation deletes the StorageSynchronized association. An instance deletion indication is generated for this operation.

All ModifySynchronization operations are described in 10.1.8 Accessibility to Created Elements. If "job completed" is returned and the replica association indicates an "... in progress" SyncState value, an instance modification indication should follow when the replica enters its final, expected state. If "job started" is returned, the replica association indicates an "... in progress" SyncState value. In this case, two instance modification indications may follow. One should indicate the final SyncState value of the replica association when the job completes with no error. The other should indicate job completion for the instance of ConcreteJob.

StorageReplicationCapabilities.SupportedModifyOperations[] allows a client to verify that a specific operation is supported by a provider.

10.5.2.2.3 CreateReplica Method

Table 209 describes the CreateReplica Method.

Table 209 - CreateReplica Method

Method: CreateReplica			
Errors: DRM26, DRM27, DRM25, MP5			
Parameters:			
Qualifiers	Name	Type	Description/Values
IN	ElementName	string	Client-assigned, friendly name
OUT	Job	ConcreteJob REF	
IN, REQ	SourceElement	LogicalElement REF	
OUT	TargetElement	LogicalElement REF	
IN	TargetSettingGoal	StorageSetting REF	
IN	TargetPool	StoragePool REF	
IN, REQ	CopyType	uint16	Copy type created: 2: Async 3: Sync 4: UnSyncAssoc 5: UnSyncUnAssoc

Method notes:

- Creates a storage element of the same type as the source element.
- Creates a StorageSynchronized association".
- Creates a SystemDevice association.
- Creates an AllocatedFromStoragePool association.
- Creates a StorageSetting instance with an ElementSettingData association.
- May create a BasedOn association.
- May create a ReplicaPoolForStorage association.
- All CopyType values may be supported.

If TargetPool is not supplied by the client, the provider response is implementation specific. For all operations not using specialized delta replica pools, the behavior of the client follows these rules:

- 1) Provider may return MP5 message indicating that TargetPool is an invalid parameter. In this case, the client should select a pool and retry the operation.
- 2) The provider will select a pool and proceed with the operation.

If the TargetPool is supplied, the provider uses the requested pool except for the next special case. For CopyType "UnSyncAssoc" creating a delta replica and DeltaReplicaPoolAccess

values of “Shared” or “Exclusive” are indicated by the provider, TargetPool should be managed by the client as shown in Table 210.

Table 210 - TargetPool Parameter for Delta Replicas

DeltaReplicaPoolAccessvalue	TargetPool supplied	TargetPool not supplied
Shared	Error with an MP5 message. The specialized pool pre-exists and is always supplied by the provider.	Always the correct client action. The provider locates the specialized pool.
Exclusive	If the method invocation is creating the first delta replica for the specified source element, TargetPool is supplied by the client. The pool is used by the provider and a ReplicaPoolForStorage association is created as a side effect. If delta replicas already exist for the source element, an error with an MP5 message will be returned.	If the specified source element has a ReplicaPoolForStorage association, the provider uses this pool as the container for a new delta replica. If this association does not exist, an error with an MP5 message is returned.

If TargetSettingGoal is not supplied by the client, the provider generates a default StorageSetting element for the replica. If TargetSettingGoal is supplied by the client, the provider will return an MP5 error message if the goal is incompatible with the corresponding target pool. If “job started” is returned, a Target Element reference may or may not be returned by the provider. 10.1.8 Accessibility to Created Elements explains when a reference to the new replica element is available to the client.

10.5.2.2.4 AttachReplica

This method 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,

10.5.2.2.5 Additional Notes on StorageConfigurationService Methods

CreateReplica shall be provided if local replicas are supported. Replica target elements are deleted using the ReturnToStoragePool method in block services. All associations and associated setting elements are automatically deleted at the same time the element is deleted.

TargetElement candidates cannot have an existing SyncedElement role to a StorageSynchronized association. The provider returns a DRM26 error message if the candidate is already in use as a replica target element. Source elements may generally be associated with multiple replica targets. The provider may return a DRM26 error in some cases if an element cannot serve as a replica source. The provider may return a DRM27 error if the client attempts to create replication targets exceeding the provider specified limits.

If the method returns "job completed", the new StorageSynchronized association is accessible to the client. If the method returns "job started", the association may not be accessible. In this case, an instance creation indication should be generated by the provider when the association is accessible.

If the provider supports replica modification, a Goal parameter may be passed by the client to change the value of modifiable setting properties. The provider may ignore properties not relevant to replication operations. The properties that may be supplied by the client include UseReplicationBuffer, InitialSynchronization and ReplicationPriority.

DEPRECATED

EXPERIMENTAL

10.5.2.3 ReplicationService Methods

10.5.2.3.1 General

The ReplicationService has a number of extrinsic methods for replication management.

All of the ReplicationService extrinsic methods return one of the following status codes. Depending on the error condition, a method may return additional error codes and/or throw an appropriate exception to indicate the error encountered.

0: (Job) Completed with no error

1: Method not supported

4: Failed

5: Invalid Parameter

4096: Method Parameters Checked - Job Started

For the input/output parameter values, refer to the appropriate MOF files and the value maps.

Table 211 summarizes the extrinsic methods for replica creation and management in the ReplicationService.

Table 211 - Extrinsic Methods of ReplicationService

Method	Described in
CreateElementReplica	Subclause 10.5.2.3.2
CreateSynchronizationAspect	Subclause 10.5.2.3.3
ModifyReplicaSynchronization	Subclause 10.5.2.3.4
ModifyListSynchronization	Subclause 10.5.2.3.5
ModifySettingsDefineState	Subclause 10.5.2.3.6
GetAvailableTargetElements	Subclause 10.5.2.3.7
GetReplicationRelationships	Subclause 10.5.2.3.8

10.5.2.3.2 CreateElementReplica

```
uint32 ReplicationService.CreateElementReplica(
    [IN] string ElementName,
    [IN, Required] uint16 SyncType,
    [IN, Required] uint16 Mode,
    [IN, Required] CIM_LogicalElement REF SourceElement,
    [IN, OUT] CIM_LogicalElement REF TargetElement,
    [IN, EmbeddedInstance("CIM_ReplicationSettingData")]
    string ReplicationSettingData,
    [OUT] CIM_ConcreteJob REF Job,
    [OUT] CIM_Synchronized REF Synchronization,
    [IN] CIM_SettingData REF TargetSettingGoal,
    [IN] CIM_ResourcePool REF TargetPool,
    [IN] uint16 WaitForCopyState);
```

This method allows a client to create (or start a job to create) a new storage object which is a replica of the specified source storage object (SourceElement). The parameters are as follows:

- **ElementName:** A end user relevant name for the element being created. If NULL, then a system supplied name is used. The value will be stored in the 'ElementName' property for the created element.
- **SyncType:** Describes the type of copy that will be made. For example, Mirror, Snapshot, and Clone.
- **Mode:** Describes whether the target elements will be updated synchronously or asynchronously.
- **SourceElement:** The source storage object which may be a StorageVolume or storage object.
- **TargetElement:**
 - As an input, refers to a target element to use. If a target element is not supplied, the implementation may locate or create a suitable target element. See 10.5.2.4.8.
 - As an output, refers to the created target storage element (i.e., the replica). If a job is created, the target element may not be available immediately.
- **ReplicationSettingData:** If provided, it overrides the default replication setting data for the given SyncType. If not provided, the implementation uses the default replication setting data.
- **Job:** If a Job is created as a side-effect of the execution of the method, then a reference to that Job is returned through this parameter (may be NULL if job is completed).

- **Synchronization:** Refers to the created association between the source and the target element. If a job is created, this parameter may be NULL, unless the association is actually formed.
- **TargetSettingGoal:** The definition for the StorageSetting to be maintained by the target storage object (the replica). If a target element is supplied, this parameter shall be NULL.
- **TargetPool:** The underlying storage for the target element (the replica) will be drawn from TargetPool if specified, otherwise the allocation is implementation specific. If a target element is supplied, this parameter shall be NULL.
- **WaitForCopyState:** Before returning, the method shall wait until this CopyState is reached. For example, CopyState of Initialized means associations have been established, but there is no data flow. CopyState of Synchronized indicates the replica is an exact copy of the source element. CopyState of UnSynchronized means copy operation is in progress (see Table 202, "CopyStates Values," for the CopyStates).

Method Notes:

- Creates a storage element of the same type as the source element.
- Creates a StorageSynchronized association.
- Creates SystemDevice, AllocatedFromStoragePool, and ElementSettingData associations to the newly created target element.
- May create BasedOn and ReplicaPoolForStorage associations.

10.5.2.3.3 CreateSynchronizationAspect

```
uint32 ReplicationService.CreateSynchronizationAspect(
    [IN]    string ElementName,
    [IN, Required]    uint16 SyncType,
    [IN, Required]    uint16 Mode,
    [IN]    CIM_ReplicationGroup REF SourceGroup,
    [IN]    CIM_LogicalElement REF SourceElement,
    [IN]    uint16 Consistency,
    [IN, EmbeddedInstance ( "CIM_ReplicationSettingData" )]
        string ReplicationSettingData,
    [OUT]    CIM_ConcreteJob REF Job,
    [OUT]    CIM_SettingsDefineState REF SettingsState );
```

This method allows a client to create (or start a job to create) new instances of SynchronizationAspect that are associated to the source element via the SettingsDefineState associations. This representation may be of a form of pointers or a series of checkpoints that keep track of the source element data for the created point-in-time.

This method does not include a target element, however, a target element can be added subsequently using the ModifySettingsDefineState method.

The method creates individual associations between the source elements and the instances of SynchronizationAspect.

The parameters are as follows:

- **ElementName:** A end user relevant name. If NULL, then a system supplied default name can be used. The value will be stored in the ElementName property of the created SynchronizationAspect.
- **SyncType:** See 10.5.2.3.2: CreateElementReplica's parameters.
- **Mode:** See 10.5.2.3.2: CreateElementReplica's parameters.

- SourceGroup: This should be null for ungrouped copies.
- SourceElement: See 10.5.2.3.2: CreateElementReplica's parameters.
- Consistency: This should be null for ungrouped copies.
- ReplicationSettingData: See 10.5.2.3.2: CreateElementReplica's parameters.
- Job: See 10.5.2.3.2: CreateElementReplica's parameters.
- SettingsState: Refers to the created association between the source element or group and the instance of the SynchronizationAspect. If a job is created, this parameter may be NULL, unless the association is actually formed.

Method Notes:

- May create an instance of SynchronizationAspect if an appropriate one does not exist already.
- May create ReplicaPoolForStorage associations.

10.5.2.3.4 ModifyReplicaSynchronization

```
uint32 ReplicationService.ModifyReplicaSynchronization(
    [IN, Required] uint16 Operation,
    [IN, Required] CIM_Synchronized REF Synchronization,
    [IN, EmbeddedInstance ( "CIM_ReplicationSettingData" )]
    string ReplicationSettingData,
    [IN] CIM_StorageSynchronized REF SyncPair[],
    [OUT] CIM_ConcreteJob REF Job,
    [IN] boolean Force,
    [OUT] CIM_SettingsDefineState REF SettingsState,
    [IN] uint16 WaitForCopyState);
```

This method allows a client to modify (or start a job to modify) the synchronization association between two storage objects. The parameters are as follows:

- Operation: This parameter describes the type of modification to be made to the replica and/or to the related associations, for example, Split.
- Synchronization: The reference to the replication association describing the elements relationship that is to be modified.
- ReplicationSettingData: See 10.5.2.3.2: CreateElementReplica's parameters.
- SyncPair[]: For operations on ungrouped elements, this parameter should be NULL.
- Job: See 10.5.2.3.2: CreateElementReplica's parameters.
- SettingsState: Reference to the association between the source element and an instance of SynchronizationAspect. This parameters applies to operations such as Dissolve, which dissolves the Synchronized relationship, but causes the SettingsDefineState association to be created. Depending on the implementation, Deactivate may also return a SettingsState.
- Force: Some operations may cause an inconsistency among the target elements. If true, the client is not warned and the operation is performed if possible.
- WaitForCopyState: See 10.5.2.3.2: CreateElementReplica's parameters.

10.5.2.3.5 ModifyListSynchronization

```
uint32 ReplicationService.ModifyListSynchronization(
    [IN, Required] uint16 Operation,
```

```

[IN, Required] CIM_Synchronized REF Synchronization[],
[IN, EmbeddedInstance ( "CIM_ReplicationSettingData" )]
    string ReplicationSettingData,
[OUT] CIM_ConcreteJob REF Job,
[IN] boolean Force,
[IN] uint16 WaitForCopyState);

```

This method allows a client to modify (or start a job to modify) a list of synchronization associations between two storage objects. The parameters are as follows:

- **Operation:** This parameter describes the type of modification to be made to the replica and/or to the related associations, for example, Split.
- **Synchronization:** An array of references to the replication association describing the elements relationship that is to be modified. All elements of the this array shall of the same concrete class, i.e., StorageSynchronized, and shall have the same SyncType, the same Mode, and the Operation shall be valid for the ReplicationType – SyncType, Mode.
- **ReplicationSettingData:** See 10.5.2.3.2: CreateElementReplica's parameters.
- **Job:** See 10.5.2.3.2: CreateElementReplica's parameters.
- **Force:** Some operations may cause an inconsistency among the target elements. If true, the client is not warned and the operation is performed if possible.
- **WaitForCopyState:** See 10.5.2.3.2: CreateElementReplica's parameters. All the supplied synchronization associations shall reach at least the specified CopyState before the method returns.

10.5.2.3.6 ModifySettingsDefineState

```

uint32 ReplicationService.ModifySettingsDefineState(
    [IN, Required] uint16 Operation,
    [IN, Required] CIM_SettingsDefineState REF SettingsState,
    [IN, OUT] CIM_LogicalElement REF TargetElement,
    [IN, OUT] CIM_ReplicationGroup REF TargetGroup,
    [IN] uint64 TargetElementCount,
    [OUT] CIM_Synchronized REF Synchronization,
    [IN, EmbeddedInstance ( "CIM_ReplicationSettingData" )]
        string ReplicationSettingData,
    [OUT] CIM_ConcreteJob REF Job,
    [IN] CIM_SettingData REF TargetSettingGoal,
    [IN] CIM_ResourcePool REF TargetPool,
    [IN] uint16 WaitForCopyState);

```

This method allows a client to modify (or start a job to modify) the SettingsDefineState association between the storage objects and SynchronizationAspect. The modification could range from introducing the target elements, which creates new StorageSynchronized associations, to dissolving the SettingsDefineState associations all together.

With the Copy To Target operation, the supplied SettingsState is deleted since an “active” Synchronization is created to associate the source and the target elements.

The parameters are as follows:

- **Operation:** This parameter describes the type of modification to be made to the related associations, for example, Copy To Target, which initiates the copy operation from the point-in-time view to the supplied targets.

- **SettingsState:** Refers to the associations between the source elements and the SynchronizationAspect instances.
- **TargetElement:** If TargetElement is supplied, TargetGroup and TargetCount shall be NULL.
 - As an input, if the point-in-time has only one source element, this parameter supplies the target element.
 - As an output, refers to the created target storage element (i.e., the replica). If a job is created, the target element may not be available immediately.
- **TargetGroup:** For ungrouped elements, this shall be NULL.
- **Synchronization:** The reference to the replication association describing the element relationship.
- **ReplicationSettingData:** See CreateElementReplica's parameters (10.5.2.3.2).
- **Job:** See CreateElementReplica's parameters (10.5.2.3.2).
- **TargetSettingGoal:** See CreateElementReplica's parameters (10.5.2.3.2).
- **TargetPool:** See CreateElementReplica's parameters (10.5.2.3.2).
- **WaitForCopyState:** See CreateElementReplica's parameters (10.5.2.3.2).

10.5.2.3.7 GetAvailableTargetElements Method

Since the rules for determining potential target volumes for a copy operation are not always straightforward, due to vendor-specific conditions, e.g. RAID level, the number of extents which consist of the StorageVolume, the type of storage array, and so on, it can be difficult for the client to know which volumes can be used as copy targets for a given source volume. This makes it difficult for the user to create a copy pair with the AttachReplica because she/he must know which volumes can be used for target volume for a particular source volume, otherwise the request may fail. The GetAvailableTargetElements method can be used to identify the potential target volumes for a copy operation. GetAvailableTargetElements method takes the source volume and list of candidate pools and returns the list of candidate target volumes for that source volume.

Table 212 describes the GetAvailableTargetElements Method.

Table 212 - GetAvailableTargetElements Method

Method: GetAvailableTargetElements			
Errors: DRM25, DRM27, MP5, MP11			
Parameters:			
Qualifiers	Name	Type	Description/Values
IN, REQ	SourceElement	LogicalElement REF	The original source volume for the pair.

Table 212 - GetAvailableTargetElements Method

Method: GetAvailableTargetElements			
IN	TargetPool[]	StoragePool REF	The arrays of the pools to search for target volumes. The method finds candidate target volumes from the available volumes in the specified TargetPools. This does include volumes with a Usage property value of reserved for copy target.
IN, REQ	CopyType	uint16	Copy type: 2: Async 3: Sync 4: UnSyncAssoc 5: UnSyncUnAssoc 6: Migrate
OUT	Candidates[]	LogicalElement REF	The list of candidate target volumes.

```

uint32 ReplicationService.GetAvailableTargetElements(
    [IN, Required] CIM_LogicalElement REF SourceElement,
    [IN, Required] uint16 CopyType,
    [IN, Required] uint16 Mode,
    [IN, EmbeddedInstance ( "CIM_ReplicationSettingData" )]
    string ReplicationSettingData,
    [IN] CIM_ComputerSystem REF Systems[],
    [IN] CIM_SettingData REF TargetSettingGoal,
    [IN] CIM_ResourcePool REF TargetPools[],
    [OUT] CIM_ConcreteJob REF Job,
    [OUT] CIM_LogicalElement REF Candidates[] );

```

This method allows a client to get (or start a job to get) all of the candidate target elements for the supplied source element. If a job is started, once the job completes, examine the AffectedJobElement associations for candidate targets. The parameters are as follows:

- SourceElement: The source storage object which may be a StorageVolume or storage object.
- CopyType: See CreateElementReplica's parameters (10.5.2.3.2).
- Mode: See CreateElementReplica's parameters (10.5.2.3.2).
- ReplicationSettingData: See CreateElementReplica's parameters (10.5.2.3.2). The parameter is useful for requesting a specific combination of thinly and fully provisioned elements.
- Systems[]: For local copies this parameter should be NULL.
- TargetSettingGoal: Desired target StorageSetting. If NULL, settings of the source elements shall be used.
- TargetPools[]: The storage pools for the target elements. If NULL, all storage pools are examined.
- Job: See CreateElementReplica's parameters (10.5.2.3.2).
- Candidates[]: The list of the candidate target elements found.

10.5.2.3.8 GetReplicationRelationships

```

uint32 ReplicationService.GetReplicationRelationships(

```

```
[IN]    uint16 Type,
[IN]    uint16 CopyType,
[IN]    uint16 Mode,
[IN]    uint16 SyncState,
[OUT]   CIM_ConcreteJob REF Job,
[OUT]   CIM_Synchronized REF Synchronizations[] );
```

This method allows a client to get (or start a job to get) all of the synchronization relationships known to the processing replication service. If a job is started, once the job completes, examine the AffectedJobElement associations for the synchronization relationships. The parameters are as follows:

- **Type:** The type of synchronization relationships, for example, StorageSynchronized. If this parameter is not supplied, all such relationships are retrieved.
- **SyncType:** See CreateElementReplica's parameters (10.5.2.3.2). If this parameter is not supplied, all SyncTypes are retrieved.
- **Mode:** See CreateElementReplica's parameters (10.5.2.3.2). If this parameter is not supplied, all Modes are retrieved.
- **CopyState:** Only retrieve synchronization relationships that currently this CopyState (see Table 202, "CopyStates Values,"). If this parameter is not supplied, relationships are retrieved regardless of their current CopyState.
- **Job:** See CreateElementReplica's parameters (10.5.2.3.2).
- **Synchronizations[]:** An array of elements found.

10.5.2.4 ReplicationServiceCapabilities Methods

There are a number of extrinsic methods in the ReplicationServiceCapabilities that advertise the implemented replication services capabilities.

All of the Profile extrinsic methods return one of the following status codes. Depending on the error condition, a method may return additional error codes and/or throw an appropriate exception to indicate the error encountered.

0: (Job) Completed with no error

1: Method not supported

4: Failed

5: Invalid Parameter

4096: Method Parameters Checked - Job Started

For the input/output parameter values, refer to the appropriate MOF files and the value maps.

Table 213 summarizes the extrinsic methods for replica creation and management in the ReplicationService.

Table 213 - Extrinsic Methods of ReplicationServiceCapabilities

Method	Described in
ConvertSyncTypeToReplicationType	Subclause 10.5.2.4.1
ConvertReplicationTypeToSyncType	Subclause 10.5.2.4.2
GetSupportedCopyStates	Subclause 10.5.2.4.3
GetSupportedFeatures	Subclause 10.5.2.4.4
GetSupportedOperations	Subclause 10.5.2.4.5

Table 213 - Extrinsic Methods of ReplicationServiceCapabilities (Continued)

Method	Described in
GetSupportedSettingsDefineStateOperations	Subclause 10.5.2.4.6
GetSupportedThinProvisioningFeatures	Subclause 10.5.2.4.7
GetSupportedMaximum	Subclause
GetDefaultReplicationSettingData	Subclause 10.5.2.4.8
GetSupportedReplicationSettingData	Subclause 10.5.2.4.9

10.5.2.4.1 ConvertSyncTypeToReplicationType

```

uint32 ReplicationServiceCapabilities.ConvertSyncTypeToReplicationType(
    [IN]    uint16 SyncType,
    [IN]    uint16 Mode,
    [IN]    uint16 LocalOrRemote,
    [OUT]   uint16 SupportedReplicationTypes );

```

The majority of the methods in this class accept ReplicationType which represents a combination of SyncType, Mode, and Local/Remote. This method accepts the supplied information and returns the corresponding ReplicationType, which can be passed to other methods to get the additional capabilities.

Table 214, Table 215, Table 216, and Table 217 show the values for the CovertSyncTypeToReplicationType parameters. These values also appear in the value maps in the appropriate MOF files.

Table 214 - SyncTypes

SyncType	Value
Mirror	6
Snapshot	7
Clone	8

Table 215 - Modes

Mode	Value
Synchronous	2
Asynchronous	3

Table 216 - Local or Remote

LocalOrRemote	Value
Local	2
Remote	3

Table 217 - ReplicationTypes

SupportedReplicationType	Value
Synchronous Mirror Local	2
Asynchronous Mirror Local	3
Synchronous Mirror Remote	4
Asynchronous Mirror Remote	5
Synchronous Snapshot Local	6
Asynchronous Snapshot Local	7
Synchronous Snapshot Remote	8
Asynchronous Snapshot Remote	9
Synchronous Clone Local	10
Asynchronous Clone Local	11
Synchronous Clone Remote	12
Asynchronous Clone Remote	13

10.5.2.4.2 ConvertReplicationTypeToSyncType

```
uint32 ReplicationServiceCapabilities.ConvertReplicationTypeToSyncType(
    [IN] uint16 ReplicationType,
    [OUT] uint16 CopyType,
    [OUT] uint16 Mode,
    [OUT] uint16 LocalOrRemote );
```

This method does the opposite of the method ConvertSyncTypeToReplicationType. This method translates ReplicationType to the corresponding SyncType, Mode, and Local/Remote.

10.5.2.4.3 GetSupportedCopyStates

```
uint32 ReplicationServiceCapabilities.GetSupportedCopyStates(
    [IN] uint16 ReplicationType,
    [OUT] uint16 SupportedCopyStates[],
    [OUT] boolean HostAccessible[] );
```

For a given ReplicationType, this method returns the supported CopyStates (Table 202) and a parallel array to indicate whether for a given CopyState the target element is host accessible or not (true or false).

10.5.2.4.4 GetSupportedFeatures

```
uint32 ReplicationServiceCapabilities.GetSupportedFeatures(
    [IN] uint16 ReplicationType,
    [OUT] uint16 Features[] );
```

For a given ReplicationType, this method returns the supported features listed in Table 218.

Table 218 - Features

Feature	Description
"Replication Groups"	Elements in a replication group are supported in a replication operation.
"Number of hops in multi-hop replication"	Maximum number of hops in multi-hop replication the service can manage.

Table 218 - Features

Feature	Description
"Service suspends source I/O when necessary"	Provider is able to suspend I/O to source elements before splitting the target elements. Otherwise, the client needs to quiesce the application before issuing the split command.
"Targets allocated from Any storage pool"	Specialized storage pools are not required for the target elements, as long as the pool is not reserved for special activities.
"Targets allocated from Shared storage pool"	Targets are allocated from storage pools reserved for Copy Services.
"Targets allocated from Exclusive storage pool"	Targets are allocated from exclusive storage pools.
"Targets allocated from Multiple storage pools"	Targets are allocated from multiple specialized, exclusive pools.
"Targets require reserved elements"	The target elements shall have a specific Usage value. For example, reserved for "Local Replica Target" (mirror), reserved for "Delta Replica Target" (Snapshot), etc.
"Target is associated to SynchronizationAspect"	The target element is associated to SynchronizationAspect via SettingsDefineState. SynchronizationAspect contains the point-in-time timestamp and the source element reference used to copy to the target element.
"Source is associated to SynchronizationAspect"	The source element is associated to SynchronizationAspect via the SettingsDefineState association. SynchronizationAspect contains the point-in-time information of the source data.
"Error recovery from Broken state Automatic"	For example, if the connection between the source and target elements is broken (CopyState = <i>Broken</i>), once the connection is restored, the copy operation continues automatically. If the error recovery is not automatic, it requires manual intervention to restart the copy operation. Use ModifyReplicaSynchronization, with Operation set to Resume.

10.5.2.4.5 GetSupportedOperations

```

uint32 GetSupportedOperations(
    [IN]  uint16 ReplicationType,
    [OUT] uint16 SupportedOperations[] );

```

For a given ReplicationType this method returns the supported Operations on a StorageSynchronized association that can be supplied to the ModifyReplicaSynchronization method, as shown in Table 219.

Refer to Figure 54, "CopyState Transitions" for additional information.

Table 219 - Operations

Operation	Description	Special Consideration
"Abort"	Abort the copy operation if it is possible.	
"Activate Consistency"	Enable consistency.	
"Activate"	Activate an inactive StorageSynchronized association.	

Table 219 - Operations

Operation	Description	Special Consideration
"AddSyncPair"	Add source and target elements of a StorageSynchronized association to the source and target replication groups. The SyncType of the associations shall be the same.	
"Deactivate Consistency"	Disable consistency.	
"Deactivate"	Stop the copy engine. Writes to source element are allowed.	Snapshot: Writes to target element after point-in-time is created are lost (pointers removed).
"Detach"	Remove the association between the source and target elements. Detach does not delete the target element.	
"Dissolve"	Dissolve the synchronization association between two storage objects, however, the target element continues to exist.	Snapshot: This operation also creates a SettingsDefineState association between the source element and an instance of SynchronizationAspect if the ReplicationType supports it.
"Failover"	Enable the read and write operations from the host to the target element. This operation useful for situations when the source element is unavailable.	
"Failback"	Switch the read/write activities from the host back to source element. Update source element from target element with writes to target during the failover period.	
"Fracture"	Separate the target element from the source element.	
"Resync Replica"	Resynchronize a fractured target element.	
"Restore from Replica"	Copy a fractured target element to the source element.	
"Resume"	Continue the copy operation of a suspended (or <i>Broken</i>) relationship.	To continue from the <i>Broken</i> state, the problem should be corrected first before requesting to resume.
"Reset To Sync"	Change Mode to Synchronous.	
"Reset To Async"	Change Mode to Asynchronous.	
"Return To StoragePool"	Delete a Snapshot target.	
"Reverse Roles"	Switch the source and the target elements' roles.	
"Split"	Separate the source and the target elements in a <i>consistent</i> manner.	
"Suspend"	Stop the copy engine in such a way that it can be resumed.	

Table 220 compares the action of similar Operations.

Table 220 - Comparison of Similar Operations

Operations	Description
Activate versus Resume	Activate: Activates a StorageSynchronizes association that has a CopyState of "Inactive." Resume: Resumes a StorageSynchronized association that has a CopyState of "Suspended".
Deactivate versus Suspend	Deactivate: Stops the copy engine. In the case of Snapshots, all writes to target element are deleted (pointers to changed data are removed). While inactive, writes to source element will not be committed to target element once activated. Suspend: Stops the copy engine. All writes to target element are preserved. Once resumed, pending writes to target element are committed.
Fracture versus Split	Fracture: Source and target elements are separated "abruptly." Split: Source and target elements are separated in an orderly fashion. Consistency of target elements is maintained.
Detach versus Dissolve	Detach: The association between the source and target element shall be first Fractured/Split before it can be Detached. Dissolve: The association can have a CopyState of Synchronized. Additionally, Dissolve can create a SettingsDefineState association based on GetSupportedFeatures (see 10.5.2.4.4) Capabilities.

10.5.2.4.6 GetSupportedSettingsDefineStateOperations

uint32

ReplicationServiceCapabilities.GetSupportedSettingsDefineStateOperations(

[IN] uint16 ReplicationType,

[OUT] uint16 SupportedOperations[]);

For a given ReplicationType this method returns the supported operations on a SettingsDefineState association that can be supplied to the ModifySettingsDefineState method, shown in Table 221.

Table 221 - SettingsDefineState Operations

SettingsDefineState Operation	Description	Special Consideration
"Activate Consistency"	Enable consistency	
"Deactivate Consistency"	Disable consistency	
"Delete"	Remove the SettingsDefineState association. Instance of SynchronizationAspect may also be deleted if it is not shared with other elements.	
"Copy To Target"	Introduces the target elements and forms the necessary associations between the source and the target elements (i.e., StorageSynchronized).	

10.5.2.4.7 GetSupportedThinProvisioningFeatures

uint32

```
ReplicationServiceCapabilities.GetSupportedThinProvisioningFeatures(
    [IN]   uint16 ReplicationType,
    [OUT]  uint16 SupportedThinProvisioningFeatures[] );
```

For a given ReplicationType this method returns the supported features related to thin provisioning.

A client can request a specific thin provisioning policy in the ReplicationSettingData parameter of the appropriate method call. Thin provisioning features are specified in Table 222.

Table 222 - Thin Provisioning Features

Feature	Description
"Thin provisioning is not supported"	The replication service does not distinguish between thinly and fully provisioned elements. The service treats all elements as fully provisioned elements.
"Zeros written in unused allocated blocks of target"	Applies to copying from a thinly provisioned element to a fully provisioned element. The implementation needs to allocate "real" storage blocks on the target side for the corresponding blocks of the source element that are unused.
"Unused allocated blocks of target are not initialized"	Applies to copying from a thinly provisioned element to a fully provisioned element. The implementation needs to allocate "real" storage blocks on the target side for the corresponding blocks of the source element that are unused.

```
uint32 ReplicationServiceCapabilities.GetSupportedMaximum(
    [IN]   uint16 ReplicationType,
    [IN]   uint16 Component,
    [OUT]  uint64 MaxValue );
```

This method accepts a ReplicationType and a component, it then returns a static numeric value representing the maximum number of the specified component that the service supports. A value of 0 indicates unlimited components of the given type. In all cases the maximum value is bounded by the availability of resources on the computer system. If the information is not known, the method returns 7 which indicates "Information is not available".

Effectively, this method informs clients of the edge conditions.

Table 223 shows the list of components that can be specified.

Table 223 - Components

Component	Description
"Number of target elements per source element"	Maximum number of target elements per source element.
"Number of total source elements"	Maximum number of total source elements supported by the service.
"Number of total target elements"	Maximum number of total target elements supported by the source.
"Number of hops in multi-hop replication"	Maximum number of hops in multi-hop replication the service can manage.

10.5.2.4.8 GetDefaultReplicationSettingData

```
uint32 ReplicationServiceCapabilities.GetDefaultReplicationSettingData(
    [IN]    uint16 ReplicationType,
    [OUT, EmbeddedObject]
    string DefaultInstance );
```

This method for a given ReplicationType returns the default ReplicationSettingData as an instance.

10.5.2.4.9 GetSupportedReplicationSettingData

Not defined in this International Standard.

EXPERIMENTAL**10.6 Client Considerations and Recipes****10.6.1 Discovery of Copy support and Capabilities**

A single instance of a Copy Services provider may support mirrors, snapshots and clones. A client follows these steps to fully discover and understand all capabilities of the provider:

- Locate the hosted instance of StorageConfigurationService.
- Enumerate and get all of the informational capability objects associated with StorageConfigurationService

Block services shall be supported by the provider. The Copy Services Subprofile shall be registered by the provider. The provider shall host one instance of StorageConfigurationService.

The properties of StorageConfigurationCapabilities and StorageReplicationCapabilities indicate precisely how the provider supports each copy service feature. The client should find one instance of StorageReplicationCapabilities for each SupportedSynchronizationType value supported by the provider. StorageReplicationCapabilities can be specialized as shown in Table 224.

Table 224 - Replica Specialization by CopyType

SupportedSynchronizationType value	CopyType value	Specialization
Async (2)	Async (2)	Asynchronous local mirror replication
Sync (3)	Sync (3)	Synchronous local mirror replication
UnSyncAssoc-Full (4)	UnSyncAssoc (4)	Full snapshots
UnSyncAssoc-Delta (5)	UnSyncAssoc (4)	Delta snapshots
UnSyncUnAssoc (6)	UnSyncUnAssoc (5)	Clone replication

Each instance shows the client:

- Replica type supported (full or delta)
- Methods supported and ModifySynchronization operations supported
- Any restrictions on host access to replicas

- Upper limits such as maximum replicas for one source element
- Specialized features by CopyType

Most of the properties in StorageReplicationCapabilities are optional. The client first analyzes SupportedSynchronousActions[], SupportedAsynchronousActions[], SupportedModifyOperations[] and SupportedSpecializedElements[]. Support for the remaining optional properties is conditional on the values indicated for these properties.

EXPERIMENTAL

If the CIM_ReplicationService has been implemented, another set of methods and capabilities will also exist – the CIM_ReplicationServiceCapabilities. The client should find one instance of ReplicationServiceCapabilities for each instance of hosted ReplicationService. ReplicationServiceCapabilities can be specialized as shown in Table 225.

Table 225 - Replica Specialization by SyncType/Mode

SupportedReplicationType value	SyncType/Mode value	Specialization
Synchronous Mirror Local (2)	Mirror (6) / Synchronous (2)	Synchronous mirror
Asynchronous Mirror Local (3)	Mirror (6) / Asynchronous (3)	Asynchronous mirror
Synchronous Snapshot Local (6)	Snapshot (7) / Synchronous (2)	Synchronous Snapshot
Asynchronous Snapshot Local (7)	Snapshot (7) / Asynchronous (3)	Asynchronous Snapshot
Synchronous Clone Local (10)	Clone (8) / Synchronous (2)	Synchronous Clone
Asynchronous Clone Local (11)	Clone (8) / Asynchronous (3)	Asynchronous Clone

An instance of ReplicationServiceCapabilities shows the client:

- Methods supported and ModifyReplicaSynchronization operations supported, and
- Storage Objects (e.g., Volumes or LogicalDisks) supported

The client first analyzes SupportedSynchronousActions[], SupportedAsynchronousActions[] and SupportedStorageObjects[]. Other features can be determined from the GetSupportedFeatures method of the class.

EXPERIMENTAL

EXPERIMENTAL

10.6.2 Creating and Managing Replicas

In general, creating and managing replicas involves the following steps:

- Decide on the SyncType of replica (Mirror, Snapshot, Clone) and Mode (Synchronous, Asynchronous). See 10.1.4.2.
- Locate the hosted instance of ReplicationService. See 10.1.3.
- Locate the instance of ReplicationServiceCapabilities. Utilize its properties and methods to determine the applicable capabilities offered by the implementation for the desired ReplicationType (includes SyncType and Mode). See 10.1.4.
- Use the method ReplicationService.GetAvailableTargetElements to locate appropriate target elements. Depending on the implementation, it is also possible to allow the service to locate target elements. See 10.5.2.3.7.

- Verify StoragePools have sufficient free capacity for the target elements. See 10.1.22.
- Invoke the appropriate extrinsic method of the ReplicationService to create a replica. See 10.5.2.3.2.
- Monitor the copy operation's progress by examining the replication associations properties, or subscribe to the appropriate indications – including storage pool low space alert indications. See 10.1.6 and 10.1.24.
- Invoke the method ReplicationService.ModifyReplicaSynchronization to modify a replica. For example, “split” a replica from its source element. See 10.5.2.3.4.

EXPERIMENTAL

10.6.3 Using StorageSetting for Replicas

The StorageSetting class has several properties used to create and manage replicas. Instances of this class are used as goal parameters for many of the methods used by the subprofile. These instances are serially reusable for a short sequence of operations ending with creation of a pool or an element. The client should follow these steps:

- 1) Invoke CreateSetting with SettingType value “Goal” for a selected storage pool.
- 2) Set values for all of the properties used to create and manage replicas. These properties are listed in the definition of StorageSetting in this subprofile. Property values can be changed by the ModifyInstance intrinsic method. The SupportedStorageElementUsage[] and SupportedStoragePoolUsage[] properties in StorageConfigurationCapabilities indicates which values of StorageExtentInitialUsage and StoragePoolInitialUsage are supported. Other replication properties may have been returned to the client with an initial value of “not applicable”. The client should not modify the value of any property with a value of “not applicable”.
- 3) The generated setting may initially be used one or more times as a goal parameter for the GetSupportedSizes and GetSupportedSizeRange methods. The setting may then be used once as a goal parameter for a pool or element creation method.
- 4) When the client no longer needs the generated setting instance, invoke the DeleteInstance intrinsic method.

10.6.4 Finding and Creating Target Elements

If a provider supports the AttachReplica method, the client finds or creates target elements eligible to become replicas. A provider may restrict replica targets to a specialized set of elements if element usage restrictions are supported as indicated in StorageConfigurationCapabilities. The client should follow these steps:

Case1: If the instrumentation does not support GetAvailableTargetElements method.

- 1) Determine the required size of the target element. Use the size of the source element unless a delta replica is created. If a delta replica is created, the size may be smaller than the associated source element.
- 2) Create a goal setting instance. Set StorageExtentInitialUsage to the correct value for the type of specialized element needed by the client. Set other replication setting property values as desired. Refer to 10.6.8 Creating and Managing Snapshots for guidelines on using delta reservation properties. Use this goal instance in all the remaining steps.
- 3) Search for existing StorageVolume instances that can be used as replica targets. A client can invoke the GetElementsBasedOnUsage method to locate available targets from existing elements. The client is responsible for screening the candidates for the required size and settings values. The search is always initiated on the system that will host the target element.

- 4) If no candidates exist, follow block services client considerations and recipes to create a new element as the replica target. Target elements may be created in pools or from element types that a provider supports as a component. As in step 2, set `StorageExtentInitialUsage` and all of the other replication setting properties to the required values before creating a new element. If a virtual element is created in a special delta replica pool (described in subsequent subclauses), the `Size` parameter value should be omitted when the element is created.

EXPERIMENTAL

Case2: If the instrumentation supports `GetAvailableTargetElements` method.

- 1) Select the original volume.
- 2) Get the copy target candidates by using `GetAvailableTargetElements`.
- 3) Select one of the candidates.
- 4) Create pair by `CreateElementReplica`.

EXPERIMENTAL

EXPERIMENTAL

10.6.5 Creating and Managing Pools for Delta Replicas

A provider may require specialized pools as containers for delta replicas. Such a pool only contains delta replicas based on the variable space consumption model explained below. The client should inspect the values of `StorageReplicationCapabilities.DeltaReplicaPoolAccess`. Values are:

- “Any” – Specialized pools not required for delta replicas. The provider creates delta replicas based on the fixed space consumption model and the client can select any pool as a container.
- “Shared” – a single shared pool is the container for all delta replicas. This type of pool is always preexisting and may be located with the `GetElementBasedOnUsage` method. The client may need to add space to this type of pool.
- “Exclusive” – each source element requires an exclusive, special pool for associated delta replicas. If the pool already exists, it is associated to the source element with a `ReplicaPoolForStorage` association. If the pool does not exist, the client creates the pool.

Delta replica pools are commonly created from or extended with component elements supplied by the `InExtents[]` parameter of the `CreateOrModifyStoragePool` method. The provider consumes all of the space in the supplied elements for this type of pool. All of the supplied elements should come from a single pool. Preexisting component elements may be located using the `GetElementsBasedOnUsage` method with the `Usage` parameter set to “Element Component”. New component elements may be created using a goal parameter with `StorageExtentInitialUsage` set to “Element Component”. The component element type shall be a type supported by the provider as indicated in `SupportedStorageElementTypes[]`.

A client may increase the size of a preexisting shared pool by adding component elements. A common practice would be to use multiple small elements of equal size. Selected component elements are passed to the `CreateOrModifyStoragePool` method using the `InExtents[]` parameter. The new elements are combined with any existing elements to increase the pool size.

A client may create new exclusive pools or increase the size of an existing exclusive pool. A new exclusive pool is commonly created by supplying one component element that supplies the required pool size. Later, the exclusive pool size is increased by supplying a `Size`

parameter value indicating the required new size of the pool. The provider determines how to increase the size. An exclusive delta replica pool is automatically associated to a source element by the provider. A `ReplicaPoolForStorage` association to the source element is created during the first `CreateReplica` operation that refers to the pool.

If warning threshold alerts are supported, the client may invoke `ModifyInstance` to modify the value of `StoragePool.LowSpaceWarningThreshold`. The pool size can be increased following a low space alert indication.

If the provider requires a shared pool and only supports “Replica Attachment” as the method for creating delta snapshots, then the shared pool shall be provisioned with virtual devices to be used as target elements. The client should ensure that enough virtual devices exist to create the expected maximum number of delta replicas. Some number of virtual devices may preexist. If the client creates virtual devices, create a goal element for each virtual device with `StorageExtentInitialUsage` set to “Delta Replica Target” and omit the `Size` parameter when invoking the element creation method. This type of virtual device always has an initial `SpaceConsumed` value of zero and does not have a `StorageSynchronized` association until `AttachOrModifyReplica` is subsequently invoked by the client.

Capacity management for a delta replica pool adheres to the capacity relationship formula specified in Block Services, Extent Mapping and Extent Conservation. The standard capacity relationship is:

$$\text{TotalManagedSpace} = \text{RemainingManagedSpace} + \text{SUM}(\text{SpaceConsumed})$$

where `SpaceConsumed` is a sum for all elements created in the pool. `RemainingManagedSpace` and `SpaceConsumed` properties may have volatile values for a delta replica pool and the elements in the pool. The provider shall maintain values for these properties that satisfy the formula. However, a client may receive stale values when instance properties are retrieved in multiple operations. The stale values may result in an unequal comparison when the capacity management relationship is checked. A client should not expect to determine exactly how much space is consumed by a delta replica in a shared or exclusive pool. If a snapshot service provider allows multiple snapshots to share a consumed block, only one snapshot will count the block in its `SpaceConsumed` value. The most important capacity management role for the client is to correctly size the delta replica pool. The sizing should be based on the maximum number of snapshots retained in the pool and the expected space consumption per snapshot.

If the provider supports low space warning threshold alerts, the client should subscribe to these alert indications. The client should maintain adequate pool capacity by either increasing the pool size or deleting the oldest snapshots when an alert is received.

Extent mapping and extent conservation are not supported for elements created in a specialized delta replica pool.

EXPERIMENTAL

10.6.6 Creating and Managing Mirrors

A mirror replica is the same size as the associated source element and is fully copied from the source element. A provider may allow the mirror element to be a larger size than the source element. A full background copy is normally initiated by the provider when a mirror replica is created. If the provider defers the background copy, the client may need to initiate the copy at a later time.

A provider normally runs a copy engine that maintains a mirror as the current image of the associated source element. The copy engine may operate in either synchronous or asynchronous mode. If the client requests `CopyType “Sync”` when the replica is created, the copy engine runs in synchronous mode and any write I/O operation to the source does not receive ending status until the write operation is also completed for the mirror. If the client

requests CopyType "Async", the copy engine runs in asynchronous mode and write I/O operations receive ending status when the operation completes for the source element.

A mirror may be changed from a current image of the source element to a point-in-time image using a fracture operation. A mirror in the "Fractured" state is called a split mirror. A mirror can also be converted to an independent storage element by a "Detach" operation following a fracture operation. The detached mirror is equivalent to a clone element created with a CopyType "UnSyncUnAssoc" request (discussed below).

A local mirror target element is hosted on the same system as the source element. An operation to create a mirror includes the following steps:

Step 1: search the target host using the GetElementsBasedOnUsage method with the Usage parameter value set to "Local Replica Target". The client can search the entire host or selected pools on the host. The client interfaces to the host system for the source element if a local mirror is created. The client shall provide a replica size value for the screening operation. Normally, this is the same size value as the source element. Select a candidate volume based on best fit or some other appropriate filter. Proceed to step 3 if a candidate is selected from existing elements.

Step 2: select a pool for creation of a new target element. For the pool being screened, access the associated StorageCapabilities instance and invoke CreateSetting to generate a modifiable setting object that is used as a goal parameter for one or more method invocations. Set StorageExtentInitialUsage to either "Local Replica Target". Invoke GetSupportedSizes or GetSupportedSizeRange and screen the pool based on the target element size. If the pool does not support the required size, proceed to the next candidate pool. If a candidate pool is found and CreateReplica will be used to create the new mirror, proceed to step 3. Otherwise, the client may follow operations described in Clause 6 Block Services Package to create a new replica target candidate. Note that a client may elect to bypass screening and require a user to manually select a candidate pool or target element.

Step 3: invoke AttachReplica or CreateReplica to create a new mirror replica. If the provider returns "job completed" status, the client can immediately access the StorageSynchronized association instance for the new replica. If the provider returns "job started" status, the client may need to wait for accessibility to the StorageSynchronized association as described in 10.1.10 State Management For Associated Replicas. The client may need to initiate additional operations to bring the new replica to the required synchronization state. If the provider supports an InitialReplicationState of "Initialized", the copy engine has not started a background copy operation and the client may invoke ModifySynchronization requesting a "Prepare" or "Resync" operation as needed.

The ModifySynchronization method can be invoked to manage existing mirrors. The subprofile supports the following operations:

- 1) Mirrors can be split from their associated source element using a "Fracture" operation. A split mirror is a point-in-time image of the source element. The split mirror can be used as a source for a backup operation or can be treated as a temporary clone. A split mirror can be changed back to a current image of the source element using a "Resync" operation.
- 2) Mirrors can be converted to independent storage elements by a sequence of operations including "Fracture" and "Detach".
- 3) The source element can be restored from a mirror by invoking a "Restore" operation. This should normally follow a client action that blocks host I/O to both the source element and all associated replica elements until the restore operation is completed.
- 4) A provider may support "ResetToSync" and "ResetToAsync" operations if availability and performance QoS policies change over time. Invoke "ResetToSync" when availability QoS changes to a higher priority than performance QoS. Invoke "ResetToAsync" when the reverse relationship occurs.

10.6.7 Creating a Clone and Redirected Restore Operations

A clone is a full size, fully copied local replica that becomes an independent storage element as soon as the background copy operation is completed. A clone is usually created by invoking the AttachReplica or CreateReplica methods with the CopyType parameter set to a value of "UnSyncUnAssoc". Alternatively, a clone may be created by detaching a split mirror or a frozen snapshot.

The provider shall automatically initiate a background copy operation when CopyType "UnSyncUnAssoc" is requested by a client. If the provider deploys the method as an asynchronous operation, then the provider may elect to create a temporary StorageSynchronized association that allows the client to manage copy priority for the background copy operation. This temporary association should only indicate a SyncState value of "Resync in progress" and the provider shall automatically delete the association when the background copy operation is completed. The client can modify the value of CopyPriority while the copy operation is in progress. The temporary association cannot be used for any other purpose and the client shall never invoke ModifySynchronization against this type of association.

A provider may allow a frozen snapshot to be treated as a clone. The client observes that a replica previously created with CopyType "UnSyncAssoc" has a SyncState value of "Frozen". If the provider supports the ModifySynchronization Start Copy operation, this operation may be invoked to bring the replica from idle state to frozen state. The provider may allow copy priority to be managed as described in 10.6.9 "Managing Background Copy".

The clone is a point-in-time image of the source element. The client shall supply any needed date/time value for the point-in-time because a guaranteed WhenSynced property value is not available for a clone created by a CopyType "UnSyncUnAssoc" operation. A provider may create a clone as either a synchronous or asynchronous operation. When the operation is completed, the client assumes the clone is ready to manage as an independent element if the OperationalStatus property indicates a value of "OK".

The Restore operation for the ModifySynchronization method only allows restoration to the source element associated with a replica. If a provider supports multi-level replication, a variation of clone creation may be used to restore a replica to a redirected location. Invoke a replica creation method supported by the provider passing a replica element as the source parameter and also indicate CopyType "UnSyncUnAssoc". The target may be a new element or an existing independent element.

EXPERIMENTAL

10.6.8 Creating and Managing Snapshots

Snapshot replicas are point-in-time images created with CopyType value "UnSyncAssoc". Snapshots can be created as full size replicas of a source element or as delta replicas of a source element. Snapshots usually have lower space consumption and lower copy engine overhead than either split mirrors or clones used as point-in-time images. Snapshots are only supported as local replicas hosted on the same storage system as the associated source element. A provider defines only one instance of StorageReplicationCapabilities for managing snapshots. This instance indicates one of two values for SupportedSynchronizationType:

- Full size: SupportedSynchronizationType = "UnSyncAssoc-Full"
- Delta: SupportedSynchronizationType = "UnSyncAssoc-Delta"

Snapshot providers may deploy either a fixed space consumption model or a variable space consumption model for snapshot replicas. A full size replica always uses a fixed space consumption model. A delta replica may use either a fixed or a variable model. Replica elements based on the variable model shall be created in special pools for delta replicas. A provider indicates support for special pools by including the value "Reserved as a Delta Replica Container" in StorageConfigurationCapabilities.SupportedStoragePoolUsage[]. The

replica AllocatedFromStoragePool.SpaceConsumed property has a constant value for the fixed model and a volatile, increasing value for the variable model. The RemainingManagedSpace property for the corresponding pool has a volatile, decreasing value if the pool contains replicas based on the variable model. Figure 58: "Fixed Space Consumption" and Figure 59: "Variable Space Consumption" show the fixed and variable space consumption models for delta snapshots:

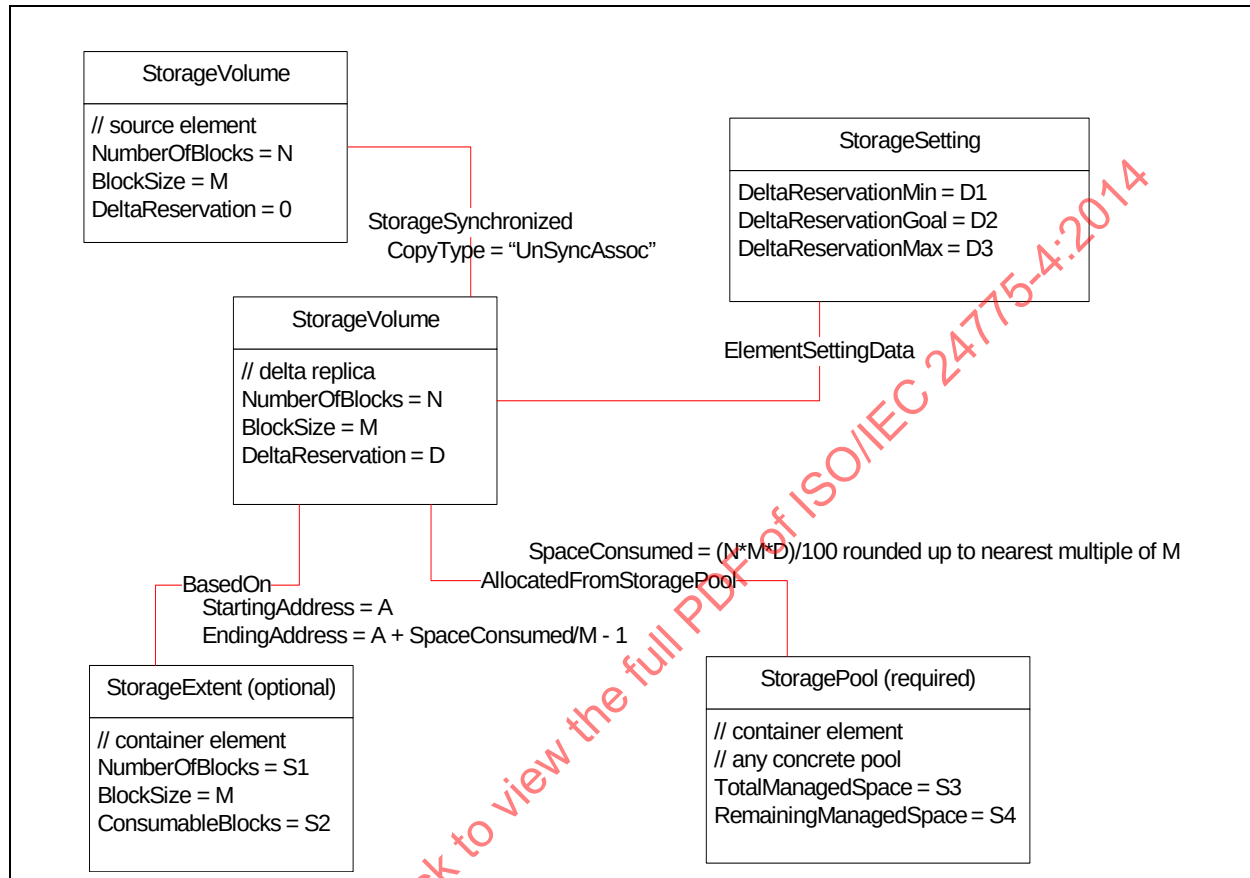
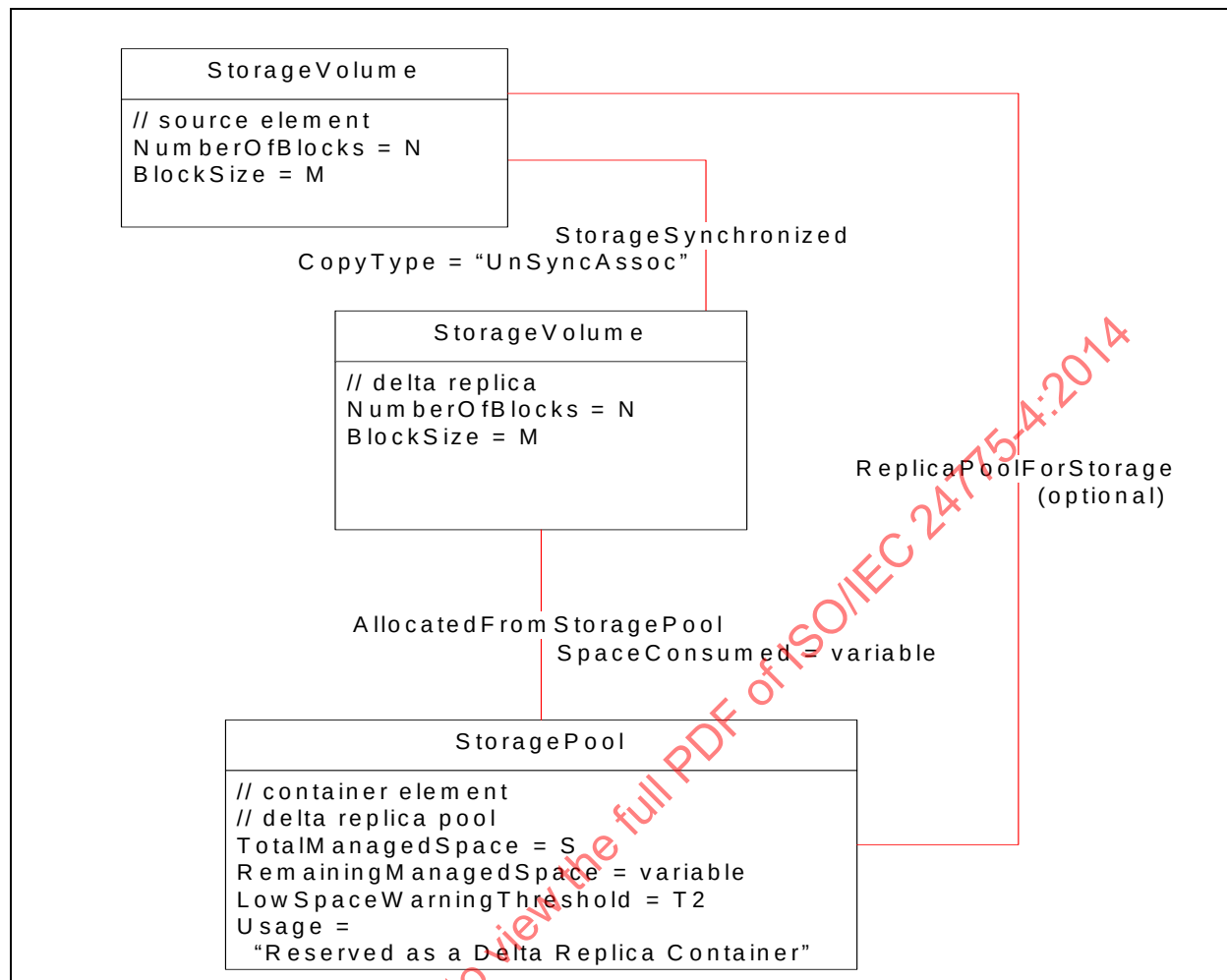


Figure 58 - Fixed Space Consumption

For full size snapshots, NumberOfBlocks and BlockSize indicate the actual size of the target element which is as large or larger than the source element. For delta snapshots, NumberOfBlocks and BlockSize have the same values as the associated source element. Delta reservation properties are only used for snapshots created by the CreateReplica method using fixed space consumption.

**Figure 59 - Variable Space Consumption**

The instances of StorageReplicationCapabilities for “UnSyncAssoc-Delta” and “UnSyncAssoc-Full” may use the patterns detailed in Table 226.

Table 226 - Patterns Supported for StorageReplicationCapabilities

SupportedSynchroniz ationType	Supported...Actions[n]	DeltaReplicaPoolAccess	Space Consumption
UnSyncAssoc-Delta	“Replica Attachment”	Any pool or extent	Fixed
UnSyncAssoc-Delta	“Replica Creation”	Any pool or extent	Fixed
UnSyncAssoc-Delta	“Replica Attachment”	Shared or Exclusive	Variable
UnSyncAssoc-Delta	“Replica Creation”	Shared or Exclusive	Variable
UnSyncAssoc-Full	“Replica Attachment”	n/a	Fixed
UnSyncAssoc-Full	“Replica Creation”	n/a	Fixed

The steps required to create a snapshot vary for each pattern. There are a number of common steps.

Step 1 the provider may limit the maximum number of replicas per source element. Verify that the limit is not exceeded when a new replica is created. The provider may restrict snapshots to

independent source elements. If the source element is a replica, verify that the provider allows snapshots of local replicas.

Step 2: locate a candidate pool eligible to contain a new snapshot. This is a special pool if the `DeltaReplicaPoolAccess` value is "Shared" or "Exclusive". A shared, special pool is a preexisting element supplied by the provider. The special pool may be populated with virtual devices that do not consume space until the `AttachReplica` method is invoked at a later time. An exclusive, special pool is created the first time a new delta replica is created for a source element that currently has no associated delta replicas. The operation for locating or creating a special pool for delta replicas is described in 10.6.5 Creating and Managing Pools for Delta Replicas. If snapshots can be created in any pool, enumerate all existing pool instances and begin screening the pools for eligibility. If snapshots are created by the `AttachReplica` method, all existing storage elements in each candidate pool should be screened for eligibility in a subsequent step.

Step 3: For the special pool or for the pool being screened, access the associated `StorageCapabilities` instance and invoke `CreateSetting` to generate a modifiable setting object to be used as a goal parameter for one or more method invocations. Set `StorageExtentInitialUsage` to either "Local Replica Target" for a full snapshot or "Delta Replica Target" for a delta snapshot.

If the operation will use `CreateReplica` to create a delta snapshot using fixed space consumption, the `DeltaReservationMin`, `DeltaReservationGoal` and `DeltaReservationMax` properties are set by the client to appropriate values for a new delta replica. The values are set in the unassociated `StorageSetting` element to be passed as a goal parameter to an extrinsic method. The client cannot modify the values of delta reservation properties in a `StorageSetting` element associated to an existing storage element. The values set by the client satisfy the relationship:

$$\text{DeltaReservationMin} \leq \text{DeltaReservationGoal} \leq \text{DeltaReservationMax}$$

as constrained by the provider. The client cannot decrease the value of `DeltaReservationMin` and cannot increase the value of `DeltaReservationMax` returned by the provider. If the provider supports a fixed space consumption model, the client estimates the fixed size of the delta replica as a percentage of the source element size and the provider determines the actual size when the element is created.

Step 4: Skip this step if `CreateReplica` is used to create a delta replica with variable space consumption. For all other cases, screen the candidate pool or the storage elements contained in the pool. If `AttachReplica` is used to create a delta replica with variable space consumption, search the special delta replica pool for a virtual storage element not in use as a replica target. For all fixed space consumption cases, the client calculates a replica size value for the screening operation. Use the source element size if a full snapshot replica is created. Use the `DeltaReplicaMax` percentage times the source element size if a delta snapshot replica is created. The generated setting created in step 3 is used as the goal parameter for the screening methods. Search existing volumes for replica target candidates as described in 10.6.4 Finding and Creating Target Elements if `AttachReplica` is used as the method to create the replica. Select a returned volume based on best fit or some other appropriate filter. Invoke `GetSupportedSizes` or `GetSupportedSizeRange` and verify that the replica size is supported by the candidate pool if `CreateReplica` is used. Proceed to step 5 if an eligible candidate element is found. Otherwise, proceed to the next candidate pool. If no candidates are located from existing pools, the client may follow recipes in block services to create a new candidate pool or element. Omit the `Size` parameter whenever a virtual replica element is created. Note that a client may elect to bypass screening and require a user to manually select a candidate pool or target element.

Step 5: invoke `AttachReplica` or `CreateReplica` to create a new snapshot. The setting property values from the goal parameter apply to the new replica. The provider determines which setting property values from the goal parameter are copied to an existing setting instance when `AttachReplica` is invoked. If a delta replica is created, the `NumberOfBlocks` and `BlockSize` values of the source element are assigned to the target.

The properties listed in Table 227 are used to monitor and manage space consumption for delta replicas using a variable space consumption pattern.

Table 227 - Space Consumption Properties

Delta Replica Property – Variable Space Consumption	Value	Modifiable
StorageExtent.NumberOfBlocks: valid for all elements. Same value as associated source element.	constant	no
StorageExtent.BlockSize: valid for all elements. Same value as associated source element.	constant	no
StoragePool.RemainingManagedSpace: valid for all pools. Value decreases by BlockSize each time replica consumes a block in the pool.	volatile	no
StoragePool.TotalManagedSpace: valid for all pools.	constant	no
StoragePool.LowSpaceWarningThreshold: valid for special delta replica pools if provider supports pool warning thresholds. Value 0 to 100.	constant	yes
AllocatedFromStoragePool.SpaceConsumed: valid for all elements. Value increases by BlockSize each time replica consumes a block in the pool.	volatile	no

The properties listed in Table 228 are used to monitor and manage space consumption for delta replicas using a fixed space consumption pattern.

Table 228 - Space Consumption Properties, Fixed Pattern

Delta Replica Property – FixedSpace Consumption	Value	Modifiable
StorageExtent.NumberOfBlocks: valid for all elements. Same value as associated source element.	constant	no
StorageExtent.BlockSize: valid for all elements. Same value as associated source element.	constant	no
StorageExtent.DeltaReservation: valid for target elements. Value set by CreateReplica method providers for delta replicas.	constant	no
StoragePool.RemainingManagedSpace: valid for all pools. Value decreases by fixed element size when element is created.	constant	no
StoragePool.TotalManagedSpace: valid for all pools.	constant	no
AllocatedFromStoragePool.SpaceConsumed: valid for all elements. Value set to fixed element size when element is created.	constant	no
StorageSetting.DeltaReservationMin: Value is % of source element size that is minimum fixed size. Used only with CreateReplica method for delta replicas.	constant	yes (goal)
StorageSetting.DeltaReservationMax: Value is % of source element size that is maximum fixed size. Used only with CreateReplica method for delta replicas.	constant	yes (goal)
StorageSetting.DeltaReservationGoal: Value is % of source element size that is the client goal for the fixed size. Used only with CreateReplica method for delta replicas.	constant	yes (goal)

Two of the above properties have volatile values automatically changed by the provider when a delta replica uses a variable space consumption model. SpaceConsumed increases and RemainingManagedSpace decreases as the associated source element is updated. When a delta replica consumes an additional block, SpaceConsumed increases by the value of BlockSize and RemainingManagedSpace decreases by the value of BlockSize. If the replica uses a fixed space consumption model, the values of these two properties are constant and change only when an extrinsic method is invoked to create or modify the replica element. The value of SpaceConsumed at the instant the delta replica is created is zero if no space is reserved or greater than zero if space is reserved. The value of RemainingManagedSpace is decreased by the value of SpaceConsumed at the instant the replica is created.

The `ModifySynchronization` method can be invoked to manage existing snapshots. The subprofile supports the following operations:

- 1) A snapshot can be reused by invoking a “Resync” operation. This releases all of the space consumed by a snapshot using the variable space consumption model. The `WhenSynced` property in `StorageSynchronized` is reset to a new date/time value.
- 2) A “Detach” operation releases all of the space consumed by a snapshot using the variable space consumption model. The detached target element can be reused for another purpose or deleted by invoking the `ReturnToStoragePool` method. If the snapshot was not previously detached, invocation of `ReturnToStoragePool` deletes the `StorageSynchronized` association.
- 3) Snapshot space consumption can be stopped by invoking a “Quiesce” operation. If the associated source element is updated while the snapshot is in “Quiesced” state it is no longer a valid point-in-time image.
- 4) The source element can be restored from a snapshot by invoking a “Restore” operation. This may follow a client action that blocks host I/O to both the source element and all associated snapshot elements until the restore operation is completed.

10.6.9 Managing Background Copy

Background copy is a full copy operation that copies all blocks from a source element to a replica element. An initial background copy is normally started by a provider when a mirror or a clone is created. Initial background copy is not normally started when a snapshot is created. A provider may allow a client to initiate a deferred background copy. Management of background copy is an optional provider capability indicated to a client for each supported `CopyType` value using properties in `StorageReplicationCapabilities`. Deferred background copy for snapshots is supported if `SupportedModifyOperations[]` includes “Start Copy” and “Stop Copy”. Deferred background copy for mirrors is supported if `InitialSynchronizationDefault` has a value other than “Not Managed” or “Not Applicable”. Copy priority can be managed for any `CopyType` if `ReplicationPriorityDefault` has a value other than “Not Managed” or “Not Applicable”.

A `ModifySynchronization` Operation value of “Start Copy” or “Stop Copy” may be invoked for snapshots. A “Start Copy” operation causes a snapshot to transition from “Idle” state to “Copy In Progress” state to “Frozen” state. A “Stop Copy” operation causes a snapshot to transition from “Copy In Progress” state to “Idle” state.

If initial background copy is not initiated when a mirror is created, a subsequent sequence of `ModifySynchronization` operations that may include `Prepare` and `Resync` should start a background copy operation.

The `InitialSynchronization` property in the goal parameter may be set to indicate whether or not an initial background copy operation is initiated at the time a replica is created. The `ReplicationPriority` property in the goal parameter may be set to override the default copy I/O rate priority.

A client may invoke `ModifyInstance` to modify the value of `CopyPriority` for a `StorageSynchronized` association. This allows a client to manage the copy I/O rate and the priority of peer I/O operations relative to host I/O operations. `CopyPriority` may be modified before or during a background copy operation. Standard `CopyPriority` values are:

- Low – peer I/O is lower priority than host I/O
- Medium – peer I/O is the same priority as host I/O
- High – peer I/O is higher priority than host I/O

EXPERIMENTAL

EXPERIMENTAL

By default, replication service performs the copy operations in the background. In other words, the methods such as `CreateElementReplica`, start the copy operation (or start a job) and return while the copy operation is in progress. To perform a copy operation in the foreground, the method may specify the `WaitForCopyState` of `Synchronized`, in which case the call will not return until the copy operation is complete.

Alternatively, the methods `CreateElementReplica` may specify the `WaitForCopyState` of `Inactive` if the `ReplicationType` supports it. In this case, the copy operation is not started until the inactive synchronization is activated (using the `ModifyReplicaSynchronization` or `ModifyListSynchronization` methods).

EXPERIMENTAL

10.6.10 Recipes

The recipes show how the extrinsic methods of the subprofile may be used. The `ModifySynchronization` method is the only mandatory method of the subprofile. Instances of `StorageReplicationCapabilities` indicate to the client which of the optional extrinsic methods are supported by a provider. If the provider supports an extrinsic method, the corresponding recipe shows required behavior.

10.6.11 Replica Modification

This recipe shows how to use the mandatory `ModifySynchronization` method.

```
// NAME: Replica Modification
// FILE: CopyServicesSP_ModSync
//
// DESCRIPTION: The synchronization state of an associated replica target
// is modified by invocation of the ModifySynchronization extrinsic
// method. The client verifies that the requested operation is supported
// by the provider before the method is invoked. The client does not
// proceed
// if an invalid state transition is attempted.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTIONS:
//
// $StgSync is a reference to a StorageSynchronized association to be
// modified. #Operation is the ModifySynchronization operation to be
// executed. If the operation completes successfully, the $StgSync
// instance is refreshed to get the current SyncState and WhenSynced
// values.
//
//
// Locate the instance of StorageConfigurationService to be used for
// method invocation. Locate the instance of
// StorageReplicationCapabilities
// to be used for validity checking.
//
//
$SourceVol = GetInstance (
    $StgSync.SystemElement,
    false, false, false,
```

```
    {"SystemName", "Usage"})
$TargetVol = GetInstance (
    $StgSync.SyncedElement,
    false, false, false,
    {"SystemName", "Usage"})
$Host = Associators (
    $SourceVol->,
    "CIM_SystemDevice",
    "CIM_ComputerSystem",
    null, null, false, false, null)
$SCS = Associators (
    $Host->,
    "CIM_HostedService",
    "CIM_StorageConfigurationService",
    null, null, false, false, null)
$L[] = Associators (
    $SCS->,
    "CIM_ElementCapabilities",
    "CIM_StorageReplicationCapabilities",
    null, null, false, false, null)

// Map StorageSynchronized.CopyType to
// StorageReplicationCapabilities.SupportedSynchronizationType.

#CT_to_SST_map = {2, 4, 6, 8, 9}
#CT = $StgSync.CopyType
#SST = #CT_to_SST_map[#CT] // 1st mapping step
if ($TargetVol.Usage == 12) {#SST++} // 2nd step -- delta snapshot
if ($TargetVol.SystemName != $SourceVol.SystemName) {#SST++}
    // 3rd step -- remote mirror

// finished with mapping

for #i in $L[]
{ // find the correct instance of StorageReplicationCapabilities
    if ($L[#i].SupportedSynchronizationType == #SST)
    {
        $SRC = $L[#i]
        break
    }
}

//
// Verify that the requested operation is supported by the provider.
//
if (!contains(#Operation, $SRC.SupportedModifyOperations[]))
{
    <error: requested ModifySynchronization operation unsupported>
}

//
// Verify that StorageSynchronized.SyncState is in a valid state for
```

```

// entry to the state diagram for the requested operation. A client
// should construct a valid transition table for each provider. The
// following tables are based on state transition diagrams in the Copy
// Services subprofile specification. The tables are indexed by
// #Operation.

// Values in the table are:
// 0: Invalid value, will not match
// 1: Operation can start from any steady state
// 2-15: specific prerequisite SyncState values
// Most operations can start from one or two steady states.
//

#Detach = 2 // Detach operation can start from any state
// Cannot start other operations from "in progress" states
#InProgress[] = {3, 5, 7, 8, 10, 15}
#StartState_Mirror1 = {0, 0, 0, 9, 4, 13, 2, 4, 6, 9, 1, 1, 0, 0}
#StartState_Mirror2 = {0, 0, 0, 9, 4, 13, 13, 4, 6, 9, 1, 1, 0, 0}
#StartState_Snapshot1 = {0, 0, 0, 0, 4, 11, 2, 4, 11, 0, 0, 0, 11, 15}
#StartState_Snapshot2 = {0, 0, 0, 0, 4, 14, 9, 4, 14, 0, 0, 0, 11, 15}

#SS = $StgSync.SyncState
if (#Operation != #Detach) // Detach can start from any state
{
    if (contains(#SS, #InProgress[]))
    {
        <error: invalid state transition attempted>
    }
    if (#CT == 4) // Check snapshot state transitions
    {
        if ((#SS != #StartState_Snapshot1[#Operation]) &&
            (#SS != #StartState_Snapshot2[#Operation]))
        {
            <error: invalid state transition attempted>
        }
    }
    else // Check mirror state transitions
    {
        if ((#SS != #StartState_Mirror1[#Operation]) &&
            (#SS != #StartState_Mirror2[#Operation]) &&
            (#StartState_Mirror1[#Operation] != 1))
        {
            <error: invalid state transition attempted>
        }
    }
}

//
// Passed all validation checks. Invoke ModifySynchronization.
%InArguments["Synchronization"] = $StgSync->
%InArgument["Operation"] = #Operation
#r = InvokeMethod(
    $SCS->,

```

```

        "ModifySynchronization",
        %InArguments,
        %OutArguments)
if (#r != 0 && #r != 4096)
{
    <error: modify failed, stop recipe and examine CIM_Error>
}
if (#r == 4096)
{
    $Job-> = %OutArguments["Job"]
    <wait for instance modification indication for job completion>
    $Job = GetInstance(
        $Job->,
        false, false, false, null)
    if ($Job.JobState != 7) // is it "Completed"?
    {
        <error: modify job failed, stop and examine CIM_Error>
    }
}
if (#Operation != #Detach)
// if not "detach" get a fresh copy of StorageSynchronized
{
    $StgSync = GetInstance(
        $StgSync->,
        false, false, false,
        {"WhenSynced", "SyncState"})
    if (contains($StgSync.SyncState, #InProgress)) // In a transition?
    { // This is an optional wait for an instance mod indication
        // showing a steady state.
        <wait for instance mod indication for $StgSync.SyncState>
        $StgSync = GetInstance( // refresh to show steady state
            $StgSync->,
            false, false, false,
            {"WhenSynced", "SyncState"})
    }
}

// Recipe complete -- $StgSync is now the StorageSynchronized association
// instance showing the final SyncState for the modify operation unless
// the operation was "detach". If the operation is detach, the
// StorageSynchronized association was deleted.

```

10.6.12 Replica Creation Or Attachment

This recipe shows how to use the CreateReplica and the AttachOrModifyReplica methods.

```

// NAME: Replica Creation Or Attachment
// FILE: CopyServicesSP_CreateOrAttach
//
// DESCRIPTION: Create a new replica target element or attach an
// existing element eligible to be used as a replica target. The client
// performs a sequence of validation steps to ensure that the operation
// will

```

```

// succeed using the selected input elements.
// The recipe supports both the CreateReplica and the
// AttachOrModifyReplica
// extrinsic methods.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTIONS:
//
// $SourceElement is the replication source and must be supplied.
// $TargetElement is an optional element selected as the replication
// target.
// If $TargetElement is supplied, the "attach" method will be invoked.
// If not, the "create" method will be invoked.
// $SCC is the instance of CIM_StorageConfigurationCapabilities
// controlling the recipe.
// $SRC is the instance of CIM_StorageReplicationCapabilities
// controlling the recipe.
// $SCS is the instance of CIM_StorageConfigurationService controlling
// the recipe.
// $TargetPool is an optional pool where a new replica is created.
// $TargetPool is not supplied if $TargetElement is supplied.
// $Goal is an optional instance of StorageSetting to be used as a goal
// parameter for a replica created in $TargetPool.
// $Pipe is an optional instance of ReplicationPipe that may be supplied
// for remote replication operations.
//
// Map StorageReplicationCapabilities.SupportedSynchronizationType value
// to the
// corresponding StorageSynchronized.CopyType value.
#SST_to_CT_map[] = {0, 0, 2, 2, 3, 3, 4, 4, 5, 6}
#SST = $SRC.SupportedSynchronizationType
#CT = #SST_to_CT_map[#SST]

// Create or Attach?
if ($TargetElement == null)
{ // Use CreateReplica
  if ((!contains(2, $SRC.SupportedAsynchronousActions[]) &&
    (!contains(2, $SRC.SupportedSynchronousActions[])))
  (
    <error: replica creation not supported>
  )
  #attach = false
} else
(
  if ((!contains(4, $SRC.SupportedAsynchronousActions[]) &&
    (!contains(4, $SRC.SupportedSynchronousActions[])))
  (
    <error: replica attachment not supported>
  )
  #attach = true
}

```

```
// Tables for checking $TargetElement.Usage value
#AllUsage[] = {2, 8, 9, 10, 11, 12}
#RemoteUsage[] = {2, 9, 11}
#LocalUsage[] = {2, 8, 10}
#DeltaSnapshotUsage = 12
// Table for checking $TargetPool.Usage value
#PoolUsage[] = {0, 0, 6, 7, 6, 7, 6, 4, 6, 5}

if (#attach)
{ // validation checks for replica attachment
    $Refs[] = ReferenceNames(
        $TargetVol.GetObjectPath(),
        "CIM_StorageSynchronized",
        null)
    if ($Refs[].size() != 0) // element already a replica source or target
    {
        <error: invalid replication target element>
    }
    if ((#SST != 6) && (#SST != 7)) // Check size unless creating a snapshot
    {
        #SourceSize = $SourceElement.NumberOfBlocks *
            $SourceElement.BlockSize
        #TargetSize = $TargetElement.NumberOfBlocks *
            $TargetElement.BlockSize
        if (#TargetSize < #SourceSize)
        (
            <error: replication target element has insufficient size>
        )
    }
    if ((#SST != 3) && (#SST != 5))
    ( // remote replication -- source and target must have different hosts
        if ($SourceVol.SystemName == $TargetVol.SystemName)
        {
            <error: target must be located on a remote host>
        }
    } else
    { // local replication -- source and target must have the same host
        if ($SourceVol.SystemName != $TargetVol.SystemName)
        {
            <error: target and source must be located on the same host>
        }
    }
    #Usage = $TargetElement.Usage
    if ((contains(#Usage, #AllUsage[]) && (#Usage != 2))
    { // continue unless Usage is "Unrestricted"
        if (#SST == 7) // Delta snapshot
        {
            if (#Usage != #SnapshotUsage)
            {

```



```

        <error: invalid usage restriction for target element>
    }
} else
{
    if ((#SST == 3) || (#SST == 5) // remote replica
    {
        if (!contains(#Usage, #RemoteUsage[]))
        {
            <error: invalid usage restriction for target element>
        }
    } else
    { // all local replica types except delta snapshots
        if (!contains(#Usage, #LocalUsage[]))
        {
            <error: invalid usage restriction for target element>
        }
    }
}
}
if ($Pipe != null)
{
    if ($Pipe.AggregationBehavior != 4)
    {
        <error: not a top-level replication pipe>
    }
}
} else
{ // validation checks for replica creation
    if ($TargetPool != null)
    {
        #Usage = $TargetPool.Usage
        if ((#Usage != 2) && (#Usage != #PoolUsage[#SST]))
        {
            <error: invalid usage restriction for target pool>
        }
    }
} // completed all validation checks

// Invoke either AttachOrModifyReplica or CreateReplica
if (#Attach)
{
    %InArguments["SourceElement"] = $SourceElement->
    %InArguments["TargetElement"] = $TargetElement->
    %InArguments["CopyType"] = #CT
    %InArguments["ReplicationPipe"] = $Pipe
    #r = InvokeMethod(
        $SCS->,
        "AttachOrModifyReplica",
        %InArguments,
        %OutArguments)
} else

```

```
{
    %InArguments["SourceElement"] = $SourceElement->
    %InArguments["CopyType"] = #CT
    %InArguments["TargetSettingGoal"] = $Goal->
    %InArguments["TargetPool"] = $TargetPool->
    #r = InvokeMethod(
        $SCS->,
        "CreateReplica",
        %InArguments,
        %OutArguments)
}
if (#r != 0 && #r != 4096)
{
    <error: replication method failed, stop and examine CIM_Error>
}
if (#r == 4096)
{
    $Job-> = %OutArguments["Job"]
    <wait for instance modification indication for job completion>
    $Job = GetInstance(
        $Job->,
        false, false, false, null)
    if ($Job.JobState != 7) // "Completed"?
    {
        <error: replication job failed, stop and examine CIM_Error>
    }
    if (!#Attach)
    {
        $TL[] = Associators(
            $Job->,
            "CIM_AffectedJobElement",
            "CIM_StorageVolume",
            null, null, false, false, null)
        $TargetElement = $TL[0]
    }
} else
{
    if (!#Attach)
    {
        $TargetElement-> = %OutArguments["TargetElement"]
        $TargetElement = GetInstance(
            $TargetElement->,
            false, false, false, null)
    }
}
}
if (#CT != 5) // not "UnSyncUnAssoc"
{ // locate new StorageSynchronized association for the target
    $SL[] = References(
        $TargetElement->,
        "CIM_StorageSynchronized",
```

```

        "SyncedElement",
        false, false, null)
    $StgSync = $SL[0]
)

```

```

// End of recipe. If successful, $TargetElement is the target replica
// instance and $StgSync is an instance of the StorageSynchronized
// association between $SourceElement and $TargetElement.

```

10.7 CIM Elements

10.7.1 General

Table 229 describes the CIM elements for Copy Services.

Table 229 - CIM Elements for Copy Services

Element Name	Requirement	Description
10.7.2 CIM_ElementCapabilities (Associates ReplicationServiceCapabilities and ReplicationService)	Conditional	Experimental. Conditional requirement: The ReplicationService is implemented.
10.7.3 CIM_ElementCapabilities (Associates StorageReplicationCapabilities and StorageConfigurationService)	Mandatory	
10.7.4 CIM_ElementCapabilities (StorageConfigurationCapabilities to StorageConfigurationService)	Mandatory	Associates StorageConfigurationCapabilities with StorageConfigurationService.
10.7.5 CIM_ElementCapabilities (StorageConfigurationCapabilities to StoragePool)	Optional	Associates StorageConfigurationCapabilities with StoragePool.
10.7.6 CIM_HostedService (Replication Service)	Conditional	Experimental. Conditional requirement: The ReplicationService is implemented.
10.7.7 CIM_HostedService (Storage Configuration Service)	Mandatory	
10.7.8 CIM_ReplicaPoolForStorage	Optional	Experimental. Associates special storage pool for Snapshots (delta replicas) to a source element.
10.7.9 CIM_ReplicationService	Optional	Experimental. Base class for Replication Services. Methods are described in the Extrinsic Methods clause.
10.7.10 CIM_ReplicationServiceCapabilities	Conditional	Experimental. Conditional requirement: The ReplicationService is implemented. A set of properties and methods that describe the capabilities of a replication services provider.
10.7.11 CIM_ReplicationSettingData	Optional	Experimental. Contains special options for use by methods of Replication Services.
10.7.12 CIM_SettingsDefineState	Optional	Experimental. Associates a storage object to an instance of SynchronizationAspect.
10.7.13 CIM_StorageCapabilities	Mandatory	Base definition is in Block Services Package.
10.7.14 CIM_StorageConfigurationCapabilities	Mandatory	Base definition is in Block Services Package. Adds two properties.

Table 229 - CIM Elements for Copy Services

Element Name	Requirement	Description
10.7.15 CIM_StorageConfigurationService	Mandatory	Base definition is in Block Services Package. Methods are described in the Extrinsic Methods clause. The methods of this Service are being Deprecated in favor of CIM_ReplicationService methods.
10.7.16 CIM_StoragePool	Mandatory	Base definition is in Block Services Package.
10.7.17 CIM_StorageReplicationCapabilities	Mandatory	A set of properties that describe the capabilities of a copy services provider.
10.7.18 CIM_StorageSetting	Mandatory	Base definition is in Block Services Package.
10.7.19 CIM_StorageSynchronized	Conditional	Experimental. Conditional requirement: The ReplicationService is implemented. Associates replica target element to source element. Property definitions and descriptions are identical to those for LogicalDisk usage.
10.7.20 CIM_StorageSynchronized (Between StorageExtent elements)	Mandatory	Associates replica target element to a source element.
10.7.21 CIM_SynchronizationAspect	Optional	Experimental. Keeps track of the source of a copy operation, even after StorageSynchronized is removed. Also keeps track of point-in-time.
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_StorageSynchronized	Mandatory	All instance creation indications for StorageSynchronized. See 10.1.24.2 InstCreation on StorageSynchronized.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_StorageSynchronized	Mandatory	All instance deletion indications for StorageSynchronized. See 10.1.24.3 InstDeletion on StorageSynchronized.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_StorageSynchronized AND SourceInstance.CIM_StorageSynchronized::SyncState <> PreviousInstance.CIM_StorageSynchronized::SyncState	Optional	Deprecated. CQL -Synchronization state transition for a replica association. This Indication is being DEPRECATED. See 10.1.24.4 InstModification on SyncState.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_StorageSynchronized AND SourceInstance.SyncState <> PreviousInstance.SyncState	Mandatory	Deprecated. Deprecated WQL - Synchronization state transition for a replica association. This Indication is being DEPRECATED. See 10.1.24.4 InstModification on SyncState.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_StorageSynchronized AND OBJECTPATH(SourceInstanceModelpath) = OBJECTPATH('string-key-of-storage-synchronized')	Optional	CQL - Instance deletion indications for a specific StorageSynchronized. See 10.1.24.5 Qualified InstDeletion on StorageSynchronized.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_StorageSynchronized AND SourceInstance.CIM_StorageSynchronized::CopyState <> PreviousInstance.CIM_StorageSynchronized::CopyState AND OBJECTPATH(SourceInstanceModelpath) = OBJECTPATH('string-key-of-storage-synchronized')	Optional	Experimental. CQL - Synchronization state transition for a specific replica association. See 10.1.24.6 Qualified InstModification on CopyState.

Table 229 - CIM Elements for Copy Services

Element Name	Requirement	Description
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_StorageSynchronized AND SourceInstance.CIM_StorageSynchronized::ProgressStat us <> PreviousInstance.CIM_StorageSynchronized::ProgressSt atus AND OBJECTPATH(SourceInstanceModelpath) = OBJECTPATH('string-key-of-storage-synchronized')	Optional	Experimental. CQL - Progress status transition for a specific replica association. See 10.1.24.7 Qualified InstModification on ProgressStatus.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_StorageSynchronized AND SourceInstance.CIM_StorageSynchronized::ProgressStat us <> PreviousInstance.CIM_StorageSynchronized::ProgressSt atus	Optional	Experimental. CQL - Progress status transition for replica associations. See 10.1.24.8 InstModification on ProgressStatus.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity = "SNIA" AND AlertingManagedElement ISA CIM_StorageSynchronized	Optional	Experimental. Be notified when CopyState is set to Broken. See 10.1.24.9 AlertIndication on StorageSynchronized.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity = "SNIA" AND AlertingManagedElement ISA CIM_StoragePool	Optional	Experimental. Remaining pool space either below warning threshold set for the pool or there is no remaining space in the pool. See 10.1.24.10 AlertIndication on StoragePool.

10.7.2 CIM_ElementCapabilities (Associates ReplicationServiceCapabilities and Repl- icationService)

Experimental.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: The ReplicationService is implemented.

Table 230 describes class CIM_ElementCapabilities (Associates
ReplicationServiceCapabilities and ReplicationService).

**Table 230 - SMI Referenced Properties/Methods for CIM_ElementCapabilities
(Associates ReplicationServiceCapabilities and ReplicationService)**

Properties	Requirement	Description and Notes
Capabilities	Mandatory	
ManagedElement	Mandatory	

10.7.3 CIM_ElementCapabilities (Associates StorageReplicationCapabilities and Stor- ageConfigurationService)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 231 describes class CIM_ElementCapabilities (Associates StorageReplicationCapabilities and StorageConfigurationService).

Table 231 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (Associates StorageReplicationCapabilities and StorageConfigurationService)

Properties	Requirement	Description and Notes
Capabilities	Mandatory	
ManagedElement	Mandatory	

10.7.4 CIM_ElementCapabilities (StorageConfigurationCapabilities to StorageConfigurationService)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 232 describes class CIM_ElementCapabilities (StorageConfigurationCapabilities to StorageConfigurationService).

Table 232 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (StorageConfigurationCapabilities to StorageConfigurationService)

Properties	Requirement	Description and Notes
Capabilities	Mandatory	The capabilities object associated with the element.
ManagedElement	Mandatory	The managed element.

10.7.5 CIM_ElementCapabilities (StorageConfigurationCapabilities to StoragePool)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 233 describes class CIM_ElementCapabilities (StorageConfigurationCapabilities to StoragePool).

Table 233 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (StorageConfigurationCapabilities to StoragePool)

Properties	Requirement	Description and Notes
Capabilities	Mandatory	The capabilities object associated with the element.
ManagedElement	Mandatory	The managed element.

10.7.6 CIM_HostedService (Replication Service)

Experimental.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: The ReplicationService is implemented.

Table 234 describes class CIM_HostedService (Replication Service).

Table 234 - SMI Referenced Properties/Methods for CIM_HostedService (Replication Service)

Properties	Requirement	Description and Notes
Antecedent	Mandatory	The hosting System.
Dependent	Mandatory	The Replication Service hosted on the System.

10.7.7 CIM_HostedService (Storage Configuration Service)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 235 describes class CIM_HostedService (Storage Configuration Service).

Table 235 - SMI Referenced Properties/Methods for CIM_HostedService (Storage Configuration Service)

Properties	Requirement	Description and Notes
Antecedent	Mandatory	The hosting System.
Dependent	Mandatory	The Storage Configuration Service hosted on the System.

10.7.8 CIM_ReplicaPoolForStorage

Experimental. Associates special storage pool for Snapshots (delta replicas) to a source element.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 236 describes class CIM_ReplicaPoolForStorage.

Table 236 - SMI Referenced Properties/Methods for CIM_ReplicaPoolForStorage

Properties	Requirement	Description and Notes
Antecedent	Mandatory	
Dependent	Mandatory	

10.7.9 CIM_ReplicationService

Experimental. Base class for Replication Services.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 237 describes class CIM_ReplicationService.

Table 237 - SMI Referenced Properties/Methods for CIM_ReplicationService

Properties	Requirement	Description and Notes
CreateElementReplica()	Mandatory	
CreateSynchronizationAspect()	Optional	
ModifyReplicaSynchronization()	Mandatory	
ModifyListSynchronization()	Optional	
ModifySettingsDefineState()	Optional	
GetAvailableTargetElements()	Optional	
GetReplicationRelationships()	Optional	

10.7.10 CIM_ReplicationServiceCapabilities

Experimental. This class defines all of the capability properties for the replication services.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: The ReplicationService is implemented.

Table 238 describes class CIM_ReplicationServiceCapabilities.

Table 238 - SMI Referenced Properties/Methods for CIM_ReplicationServiceCapabilities

Properties	Requirement	Description and Notes
SupportedReplicationTypes	Mandatory	Enumeration indicating the supported CopyType/Mode/ Local-or-Remote combinations. Values: 2: Synchronous Mirror Local 3: Asynchronous Mirror Local 6: Synchronous Snapshot Local 7: Asynchronous Snapshot Local 10: Synchronous Clone Local 11: Asynchronous Clone Local.
SupportedStorageObjects	Mandatory	Enumeration indicating the supported storage objects. Values: 2: StorageVolume 3: LogicalDisk.
SupportedAsynchronousActions	Mandatory	Identify replication methods using job control. Values: 2: CreateReplica 4: CreateSynchronizationAspect 5: ModifySynchronization 7: ModifySettingsDefineState 8: GetAvailableTargetElements 10: GetReplicationRelationships.

Table 238 - SMI Referenced Properties/Methods for CIM_ReplicationServiceCapabilities

Properties	Requirement	Description and Notes
SupportedSynchronousActions	Mandatory	Identify replication methods not using job control. Values: 2: CreateReplica 4: CreateSynchronizationAspect 5: ModifySynchronization 7: ModifySettingsDefineState 8: GetAvailableTargetElements 10: GetReplicationRelationships.
ConvertSyncTypeToReplicationType()	Mandatory	
ConvertReplicationTypeToSyncType()	Mandatory	
GetSupportedCopyStates()	Mandatory	
GetSupportedFeatures()	Mandatory	
GetSupportedConsistency()	Optional	
GetSupportedOperations()	Mandatory	
GetSupportedSettingsDefineStateOperations()	Optional	
GetSupportedThinProvisioningFeatures()	Optional	
GetSupportedMaximum()	Optional	
GetDefaultReplicationSettingData()	Optional	
GetSupportedReplicationSettingData()	Optional	

10.7.11 CIM_ReplicationSettingData

Experimental. Contains special options for use by methods of Replication Services.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 239 describes class CIM_ReplicationSettingData.

Table 239 - SMI Referenced Properties/Methods for CIM_ReplicationSettingData

Properties	Requirement	Description and Notes
Pairing	Optional	Controls how source and target elements are paired. Values: 2: Instrumentation decides 3: Exact order 4: Optimum (If possible source and target elements on different adapters).
DesiredCopyMethodology	Optional	Request specific copy methodology. Values: 1: Other 2: Instrumentation decides 3: Full-Copy 4: Incremental-Copy 5: Differential-Copy 6: Copy-On-Write 7: Copy-On-Access 8: Delta-Update.
TargetElementSupplier	Optional	If target elements are not supplied, this property indicates where the target elements should come from. Values: 1: Use existing elements 2: Create new elements 3: Use existing or Create new elements 4: Instrumentation decides.
ThinProvisioningPolicy	Optional	If the target element is not supplied, this property specifies the provisioning of the target element. Values: 2: Copy thin source to thin target 3: Copy thin source to full target 4: Copy full source to thin target 5: Provisioning of target same as source 6: Target pool decides provisioning of target element 7: Implementation decides provisioning of target.
ConsistentPointInTime	Optional	If it is true, it means the point-in-time to be created at an exact time with no I/O activities in such a way the data is consistent among all the elements or the group.
DeltaUpdateInterval	Optional	If non-zero, it specifies the interval between the snapshots of source element, for example, every 23 min (00000000002300.000000:000). If zero or NULL, the implementation decides.
Multihop	Optional	This property applies to multihop copy operation. It specifies the number of hops the starting source (or group) element is expected to be copied. Default is 1.

10.7.12 CIM_SettingsDefineState

Experimental. Associates a storage object to an instance of SynchronizationAspect.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 240 describes class CIM_SettingsDefineState.

Table 240 - SMI Referenced Properties/Methods for CIM_SettingsDefineState

Properties	Requirement	Description and Notes
ManagedElement	Mandatory	Storage Element.
SettingData	Mandatory	Synchronization Aspect.

10.7.13 CIM_StorageCapabilities

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 241 describes class CIM_StorageCapabilities.

Table 241 - SMI Referenced Properties/Methods for CIM_StorageCapabilities

Properties	Requirement	Description and Notes
DeltaReservationMin	Mandatory	Refer to property descriptions for CIM_StorageSetting class.
DeltaReservationMax	Mandatory	
DeltaReservationDefault	Mandatory	Initial value for CIM_StorageSetting.DeltaReservationGoal.

10.7.14 CIM_StorageConfigurationCapabilities

This class is only defined to maintain SMI-S 1.0 backward compatibility. This version of SMI-S indicate copy services capabilities using instances of the StorageReplicationCapabilities class.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 242 describes class CIM_StorageConfigurationCapabilities.

Table 242 - SMI Referenced Properties/Methods for CIM_StorageConfigurationCapabilities

Properties	Requirement	Description and Notes
SupportedAsynchronousActions	Mandatory	Identify replication methods using job control. Values: 8: Replica Creation 9: Replica Modification 10: Replica Attachment.
SupportedSynchronousActions	Mandatory	Identify replication methods not using job control. Values: 8: Replica Creation 9: Replica Modification 10: Replica Attachment.
SupportedStorageElementTypes	Mandatory	Storage element types that can be replicated. Values: 2: Storage Volume 4: Logical Disk.
SupportedCopyTypes	Mandatory	CopyType values: 2: Async 3: Sync 4: UnSyncAssoc 5: UnSyncUnAssoc.
InitialReplicationState	Mandatory	The initial SyncState when replica creation is completed. Values: 2: Initialized 3: Prepared 4: Synchronized.

10.7.15 CIM_StorageConfigurationService

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 243 describes class CIM_StorageConfigurationService.

Table 243 - SMI Referenced Properties/Methods for CIM_StorageConfigurationService

Properties	Requirement	Description and Notes
ModifySynchronization()	Mandatory	Deprecated. This method is Deprecated in favor of ReplicationService.ModifySynchronization.
CreateReplica()	Optional	Deprecated. This method is Deprecated in favor of ReplicationService.CreateElementReplica.
AttachReplica()	Optional	Deprecated. This method is Deprecated in favor of ReplicationService.CreateElementReplica.

10.7.16 CIM_StoragePool

LowSpaceWarningThreshold only applies to specialized pools created as containers for delta replica elements using dynamic, variable space consumption. The specialized pool is associated to either the StorageConfigurationService or to a single replica source element.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 244 describes class CIM_StoragePool.

Table 244 - SMI Referenced Properties/Methods for CIM_StoragePool

Properties	Requirement	Description and Notes
LowSpaceWarningThreshold	Optional	Experimental. Percentage of TotalManagedSpace triggering an alert indication. When RemainingManagedSpace reaches or falls below this percentage, the indication is generated.

10.7.17 CIM_StorageReplicationCapabilities

This class defines all of the capability properties for a replication service. A provider shall supply one instance for each SupportedSynchronizationType value supported.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 245 describes class CIM_StorageReplicationCapabilities.

Table 245 - SMI Referenced Properties/Methods for CIM_StorageReplicationCapabilities

Properties	Requirement	Description and Notes
SupportedSynchronizationType	Mandatory	Provider shall supply one instance of this class for each supported value. Values: 2: Async 3: Sync 4: UnSyncAssoc-Full 5: UnSyncAssoc-Delta 6: UnSyncUnAssoc.
SupportedAsynchronousActions	Mandatory	Identify replication methods using job control. Values: 2: Local Replica Creation 4: Local Replica Modification 6: Local Replica Attachment.
SupportedSynchronousActions	Mandatory	Identify replication methods not using job control. Values: 2: Local Replica Creation 4: Local Replica Modification 6: Local Replica Attachment.

Table 245 - SMI Referenced Properties/Methods for CIM_StorageReplicationCapabilities

Properties	Requirement	Description and Notes
InitialReplicationState	Mandatory	The initial SyncState when replica creation is completed. Values: 2: Initialized 3: Prepared 4: Synchronized 5: Idle.
SupportedModifyOperations	Mandatory	Identify ModifySynchronization operations supported for this CopyType. Values: 2: Detach 3: Fracture 4: Resync 5: Restore 6: Prepare 7: Unprepare 8: Quiesce 9: Unquiesce 10: Reset To Sync 11: Reset To Async 12: Start Copy 13: Stop Copy.
ReplicaHostAccessibility	Mandatory	Host access restrictions. Values: 2: Not accessible 3: Any host may access 4: Only accessible by the associated source element host 5: Accessible by hosts other than the source element host.
HostAccessibleState	Mandatory	Associated replicas are host accessible for these SyncState values: 2: Initialized 3: Prepare In Progress 4: Prepared 5: Resync In Progress 6: Synchronized 7: Fracture In Progress 8: Quiesce In Progress 9: Quiesced 10: Restore In Progress 11: Idle 12: Broken 13: Fractured 14: Frozen 15: Copy In Progress.

Table 245 - SMI Referenced Properties/Methods for CIM_StorageReplicationCapabilities

Properties	Requirement	Description and Notes
LocalMirrorSnapshotSupported	Conditional	Conditional requirement: Local or remote mirrors supported. Only valid for CopyType "Sync" and "Async": true: local mirror replicas can be snapshot source element false: local mirrors cannot be snapshot source.
MaximumReplicasPerSource	Mandatory	Maximum replicas of all types allowed for one source element.
MaximumLocalReplicationDepth	Conditional	Conditional requirement: Local or remote mirrors supported. Volume A mirrors Volume B mirrors Volume C to this maximum allowable depth.
InitialSynchronizationDefault	Conditional	Conditional requirement: Managed background copy operations supported. Refer to CIM_StorageSetting.InitialSynchronization.
ReplicationPriorityDefault	Conditional	Conditional requirement: Managed background copy operations supported. Refer to CIM_StorageSetting.ReplicationPriority.
LowSpaceWarningThresholdDefault	Conditional	Conditional requirement: Snapshots supported. Default value for LowSpaceWarningThreshold. Percentage value between 0 and 100.
DeltaReplicaPoolAccess	Conditional	Conditional requirement: Snapshots supported. Indicates if a specialized pool is required as a container for delta replicas. Values: 2: Any pool may contain delta replicas 3: Exclusive special pool per source element 4: Shared special pool for all source elements.

10.7.18 CIM_StorageSetting

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 246 describes class CIM_StorageSetting.

Table 246 - SMI Referenced Properties/Methods for CIM_StorageSetting

Properties	Requirement	Description and Notes
DeltaReservationMin	Mandatory	Minimum space reserved for a delta replica at time of creation. Value 0 to 100 is a percentage of the source element size.
DeltaReservationMax	Mandatory	Maximum space reserved for a delta replica at time of creation. Value 0 to 100 is a percentage of the source element size.

Table 246 - SMI Referenced Properties/Methods for CIM_StorageSetting

Properties	Requirement	Description and Notes
DeltaReservationGoal	Mandatory	Goal for space reserved for a delta replica at time of creation. Value 0 to 100 is a percentage of the source element size.
InitialSynchronization	Optional	Experimental. Indicates that the source element should be fully copied to the target element when a replica is created. Values: 0: Not applicable 1: Not managed 2: Start copy operation 3: Do not start copy operation.
ReplicationPriority	Optional	Experimental. Priority of copy engine I/O relative to host I/O. Values: 0: Not applicable 1: Not managed 0: Not managed 2: Lower than host I/O 3: Same as host I/O 4: Higher than host I/O.

10.7.19 CIM_StorageSynchronized

Experimental. Associates replica target element to source element. CIM_StorageSynchronized is subclassed from CIM_StorageSynchronized.

Created By: Extrinsic: CreateReplica, AttachReplica, CreateElementReplica

Modified By: Extrinsic: ModifySynchronization, ModifyReplicaSynchronization

Deleted By: Extrinsic: ModifySynchronization, ModifyReplicaSynchronization

Requirement: The ReplicationService is implemented.

Table 247 describes class CIM_StorageSynchronized.

Table 247 - SMI Referenced Properties/Methods for CIM_StorageSynchronized

Properties	Requirement	Description and Notes
WhenEstablished	Optional	Specifies when the association was established.
WhenActivated	Optional	Specifies when the association was activated.
WhenSuspended	Optional	Specifies when the association was suspended.
SyncType	Mandatory	Type of association between source and target elements. Values: 6: Mirror 7: Snapshot 8: Clone.
Mode	Mandatory	Specifies when target elements are updated. Values: 2: Synchronous 3: Asynchronous.

Table 247 - SMI Referenced Properties/Methods for CIM_StorageSynchronized

Properties	Requirement	Description and Notes
RequestedCopyState	Mandatory	Indicates the last requested or desired state for the association. Values: 6: Synchronized 13: Fractured 17: Split 18: Inactive 19: Suspended 20: FailedOver.
SyncState	Mandatory	State of association between source and target groups. Values: 2: Initialized 6: Synchronized 12: Broken 13: Fractured 16: Unsynchronized 17: Split 18: Inactive 19: Suspended 20: FailedOver 21: Mixed 22: Not Applicable.
ProgressStatus	Mandatory	Status of association between source and target groups. Values: 2: Completed 3: Dormant 4: Initializing 5: Synchronizing 6: Resyncing 7: Restoring 8: Fracturing 9: Splitting 10: Failing over 11: Failing back 12: Mixed.
PercentSynched	Optional	Specifies the percent of the work completed to reach synchronization. For synchronized associations (e.g. CopyType Mirror), while fractured, the percent difference between source and target elements can be derived by subtracting PercentSynched from 100.
SyncedElement	Mandatory	
SystemElement	Mandatory	

10.7.20 CIM_StorageSynchronized (Between StorageExtent elements)

Created By: Extrinsic: CreateReplica, AttachReplica, CreateElementReplica

Modified By: Extrinsic: ModifySynchronization, ModifyReplicaSynchronization

Deleted By: Extrinsic: ModifySynchronization, ModifyReplicaSynchronization

Requirement: Mandatory.

Table 248 describes class CIM_StorageSynchronized (Between StorageExtent elements).

**Table 248 - SMI Referenced Properties/Methods for CIM_StorageSynchronized
(Between StorageExtent elements)**

Properties	Requirement	Description and Notes
WhenSynced	Mandatory	If the replica is a PIT image, this value is the date/time created.
SyncMaintained	Mandatory	Boolean indicating whether synchronization is maintained.
CopyType	Mandatory	Type of association between source and target. Values: 2: Async 3: Sync 4: UnSyncAssoc 5: UnSyncUnAssoc
ReplicaType	Optional	Informational property describing the type of replication. Values: 0: Not specified 2: Full Copy 3: Before Delta 4: After Delta 5: Log.
SyncState	Mandatory	State of the association between source and target. Values: 2: Initialized 3: PrepareInProgress 4: Prepared 5: ResyncInProgress 6: Synchronized 7: FractureInProgress 8: QuiesceInProgress 9: Quiesced 10: RestoreInProgress 11: Idle 12: Broken 13: Fractured 14: Frozen 15: CopyInProgress.

**Table 248 - SMI Referenced Properties/Methods for CIM_StorageSynchronized
(Between StorageExtent elements)**

Properties	Requirement	Description and Notes
CopyPriority	Optional	Experimental. Priority of copy engine I/O relative to host I/O. Values: 0: Not managed 1: Lower than host I/O 2: Same as host I/O 3: Higher than host I/O.
SyncedElement	Mandatory	
SystemElement	Mandatory	

10.7.21 CIM_SynchronizationAspect

Experimental. Keeps track of source of a copy operation and point-in-time.

Created By: Extrinsic: CreateElementReplica, CreateSynchronizationAspect

Modified By: Extrinsic: ModifyReplicaSynchronization

Deleted By: Extrinsic: ModifyReplicaSynchronization, ModifySettingsDefineState

Requirement: Optional.

Table 249 describes class CIM_SynchronizationAspect.

Table 249 - SMI Referenced Properties/Methods for CIM_SynchronizationAspect

Properties	Requirement	Description and Notes
SyncType	Mandatory	Type of association between source and target elements. Values: 6: Mirror 7: Snapshot 8: Clone.
ConsistencyEnabled	Mandatory	Set to true if consistency is enabled.
ElementName	Optional	A end user relevant name. The value will be stored in the ElementName property of the created SynchronizationAspect.
CopyMethodology	Optional	Indicates the copy methodology utilized for copying. Values: 2: Implementation decides 3: Full-Copy 4: Incremental-Copy 5: Differential-Copy 6: Copy-On-Write 7: Copy-On-Access 8: Delta-Update.
WhenPointInTime	Optional	Specifies when point-in-time was created.
SourceElement	Mandatory	Reference to the source element or the source group of a copy operation and/or a point-in-time.

STABLE

DEPRECATED

11 Disk Drive Subprofile

The functionality of the Disk Drive Subprofile has been subsumed by the Clause 12 Disk Drive Lite Subprofile.

The Disk Drive Subprofile is defined in 7.3.3.4 of SMI-S 1.0.2.

DEPRECATED

IECNORM.COM : Click to view the full PDF of ISO/IEC 24775-4:2014

STABLE

12 Disk Drive Lite Subprofile

12.1 Description

12.1.1 General

The Disk Drive Lite Subprofile is used to model disk drive devices. This subprofile assumes the drive is linked to a larger system (e.g., Array, SDE). The model supports asset information, health and status, and Physical information. The model also supports external links to Pool membership, extent mapping, backend port modeling, SCSI buss and address mapping, and physical containment in system packages. The subprofile also includes active management of an optional location indicator.

12.1.2 Base model

A disk drive is modeled as a single MediaAccessDevice (DiskDrive). The DiskDrive class shall be linked to a single StorageExtent (representing the storage of the drive) by a MediaPresent association. The StorageExtent class represents the storage of the drive and contains its size. Other classes further refine the model. PhysicalPackage contains asset information for the device and shall be connected by a Realizes association. The model can optionally contain SoftwareIdentity that contains information about the firmware and is linked by a DeviceSoftware association.

The CIM_DiskDrive class can be optionally subclassed to SNIA_DiskDrive to add a set of properties that model the capabilities of the drive. The properties include DiskType, FormFactor, and Encryption. DiskType contains information about the technology employed to store data. FormFactor contains the physical size of the drive. Encryption reflects the state of the encryption feature implemented by some disk drives.

Disk Drive Lite also has an optional set of classes to model the ports on the drive. These classes include LogicalPort and ProtocolEndpoint. LogicalPort is subclassed to many different port types (e.g., Fibre channel, SAS, SATA ...). All subclasses shall define the "PortType" property as mandatory so that it can be used to determine the interface on the drive.

Note that the logicalPort class, ProtocolEndpoint, and the DiskDrive properties DiskType, FormFactor, and Encryption will be made mandatory in the future.

12.1.3 Associations to external classes

The Disk Drive Subprofile ties into the rest of the system via a number of key associations.

- ConcreteComponent - Is used to associate the StorageExtent to the StoragePool that the disk is part of. Required when used with Block Services profile.
- BasedOn - Is used to associate The StorageExtent exported by the Disk Drive to another (higher level) extent (or a Volume).
- Container - Is used to associate the physical package of the disk drive to the physical package of the system.
- SystemDevice - Is used to scope the Disk to the system containing it and is mandatory.
- ProtocolControllerAccessesUnit - Is used to link the Disk to system port(s) it is accessed through.
- SCSIInitiatorTargetLogicalUnitPath or MemberOfCollection may be used with Initiator Port Profiles.
- MemberOfCollection - Is used with Storage Device Enclosure.

12.1.4 Active Management

The DiskDrive class has been enhanced by the addition of a property (LocationIndicator) to read or set the state of a location indicator. When read the property returns a value that can be used to determine if the indicator is supported and its value. When written the indicator's state is set.

12.1.5 Diagram of CIM Elements

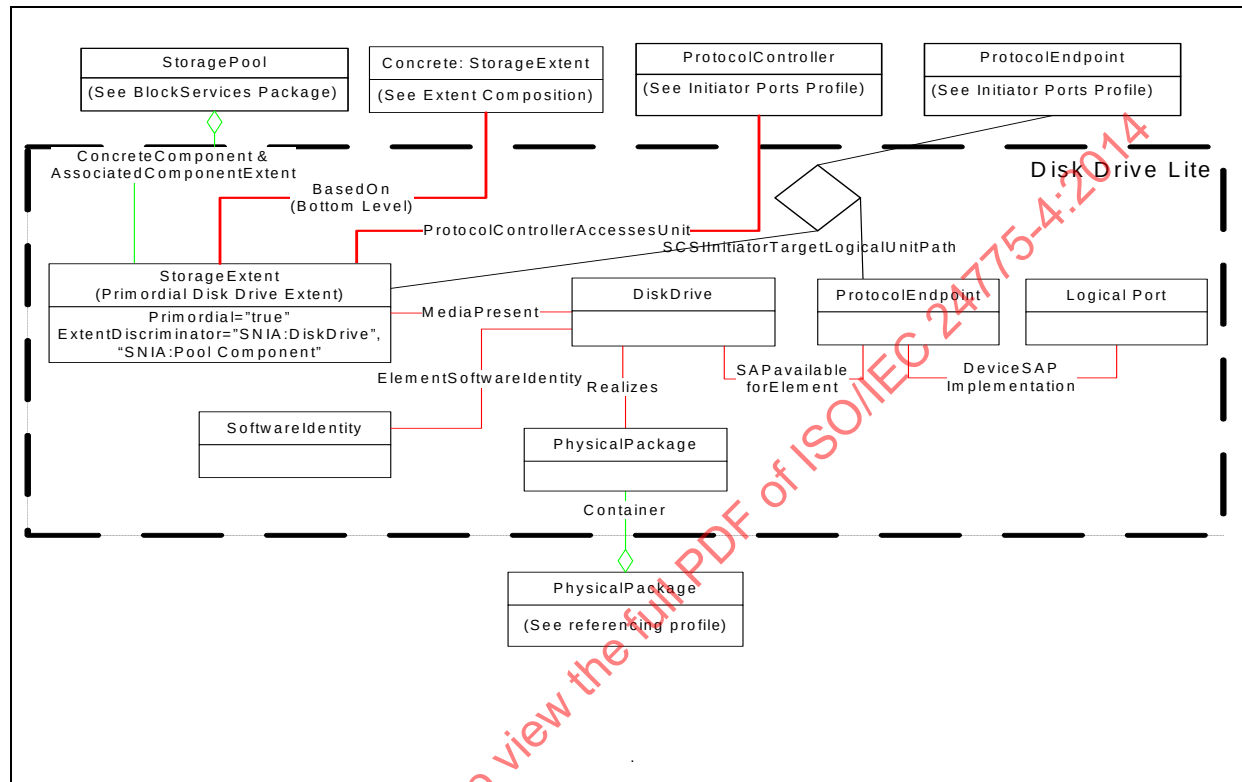


Figure 60 - CIM Elements in the Disk Drive Model

Figure 60 illustrates the CIM elements for modeling of Disk Drives.

This Profile defines the following CIM Classes (and their uses):

- DiskDrive - Used to represent the drive characteristics.
- LogicalPort - To represent (target ports) for accessing the disk drive. This is optional.
- PhysicalPackage - Used to represent the physical packaging aspects of the drive.
- ProtocolEndpoint - To represent the protocol used (SCSI or ATA) for accessing the disk drive. This is optional.
- SoftwareIdentity - Used to represent the firmware information for the disk drive.
- StorageExtent (Primordial Disk Drive Extents) - Used to represent the storage media on a disk drive.

12.1.6 Durable Names and Correlatable IDs of the Profile

None.

12.1.7 Conditional Associations to other profiles

The following associations shall be implemented if certain other profiles are implemented:

DEPRECATED

- ConcreteComponent

When implementing the Disk Drive Lite Subprofile with the Block Services Package, the ConcreteComponent association between the disk drive StorageExtent and the primordial StoragePool to which it is assigned shall be implemented. Block Services models logical storage (StoragePools) and Disk Drive Lite models is StorageExtents that provide storage for a primordial storage pool.

DEPRECATED

- AssociatedComponentExtent

When implementing the Disk Drive Lite profile with the Block Services Package, the AssociatedComponentExtent association between the disk drive StorageExtent and a primordial StoragePool to which it is assigned shall be implemented. Block Services models logical storage (StoragePools) and Disk Drive Lite models is StorageExtents that provide storage for a primordial storage pool.

- BasedOn

When implementing the Disk Drive Lite subprofile with Extent Composition, the BasedOn association between the primordial disk drive StorageExtent and higher level concrete StorageExtents that directly use storage from the disk drive extent shall be implemented.

12.1.8 Optional Associations to other profiles

The SCSIInitiatorTargetLogicalUnitPath or MemberOfCollection from CIM_ProtocolEndpoint may be used with Initiator Port Profiles.

The MemberOfCollection association from the LogicalPort is used with enclosure profiles.

12.2 Health and Fault Management Considerations

The DiskDrive.OperationalStatus contains the overall status of the disk, summarized in Table 250.

Table 250 - OperationalStatus For DiskDrive

Primary Operational Status	Subsidiary Operational Status	Description
2 "OK"		Disk Drive is enabled.
5 "Predictive Failure"		Disk Drive is functioning nominally but is predicting a failure in the near future
6 "Error"		Disk Drive is no longer functioning.
8 "Starting"		Disk Drive is becoming enabled.
9 "Stopping"		Disk Drive is being disabled.
10 "Stopped"		Disk Drive is disabled.

Table 251 shows the relationship between the EnabledState of a disk drive to the drives OperationalStatus and the disk drive StorageExtent OperationalStatus.

Table 251 - Enabled State

StorageExtent. OperationalStatus	DiskDrive. OperationStatus	DiskDrive. EnabledState
2, OK	2, OK	2, Enabled
13, Lost Communication	10, Stopped	3, Disabled
13, Lost Communication	9, Stopping	4, Shutting Down
13, Lost Communication	2, OK	6, Enabled but Offline
13, Lost Communication	8, Starting	10, Starting

12.3 Cascading Considerations

Not defined in this International Standard.

12.4 Supported Profiles, Subprofiles and Packages

Related Profiles for Disk Drive Lite: Not defined in this International Standard.

12.5 Methods of this Profile – Extrinsic Methods on Disk Drives

Request State Change

uint32 RequestStateChange(

[In] uint16 RequestedState,

[Out] CIM_ConcreteJob REF Job,

[In] datetime TimeoutPeriod)

The allowed state changes are indicated by the RequestedStatesSupported property of EnabledLogicalElementCapabilities. A Job shall be returned if the operation takes longer than the TimeoutPeriod. The Requested State of Offline makes a drives extents unavailable to the dependent volume.

The Job may represent a drive rebuild if the RequestedState of the drive is Offline and a failover shall be complete before the offline operation can finish.

12.6 Registered Name and Version

Disk Drive Lite version 1.5.0 (Component Profile).

12.7 CIM Elements

12.7.1 General

Table 252 describes the CIM elements for Disk Drive Lite.

Table 252 - CIM Elements for Disk Drive Lite

Element Name	Requirement	Description
12.7.2 CIM_ATAPort (Disk Drive Target ATA Port)	Optional	Represents an ATA target port for the disk drive.
12.7.3 CIM_ATAProtocolEndpoint (Disk Drive target ATA Protocol Endpoint)	Optional	A target ATA protocol endpoint for a disk drive if ATA protocols are supported.
12.7.4 CIM_AssociatedComponentExtent (Pool Component to Primordial Pool)	Mandatory	

Table 252 - CIM Elements for Disk Drive Lite (Continued)

Element Name	Requirement	Description
12.7.5 CIM_BasedOn (Bottom Level BasedOn)	Conditional	Conditional requirement: Implementation of the Extent Composition profile. Associates a concrete StorageExtent representing a decomposition (partial allocation) or composition to the disk drive StorageExtent that it is allocated from.
12.7.6 CIM_ConcreteComponent (Disk Extent to Primordial Pool)	Conditional	Deprecated. Conditional requirement: Implementation of the Block Services Package. Associates a disk drive extent to a primordial storage pool.
12.7.7 CIM_Container	Optional	Associates a disk drive physical package to its higher level package.
12.7.8 CIM_DeviceSAPImplementation (ATA)	Optional	Associates a target ATA protocol endpoint to the target port for the drive.
12.7.9 CIM_DeviceSAPImplementation (SCSI)	Optional	Associates a target SCSI protocol endpoint to the target port for the drive.
12.7.10 CIM_DiskDrive	Mandatory	Represents the disk drive.
12.7.11 CIM_ElementSoftwareIdentity	Mandatory	Associates the firmware (SoftwareIdentity) to a disk drive.
12.7.12 CIM_FCPort (Disk Drive Target FC Port)	Optional	Represents an FC target port for the disk drive.
12.7.13 CIM_FilterCollection (Disk Drive Lite Predefined FilterCollection)	Conditional	Experimental. Conditional requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='5' (Predefined Filter Collections). This is a collection of predefined IndicationFilters to which a client may subscribe.
12.7.14 CIM_HostedCollection (System to predefined IndicationFilters)	Conditional	Experimental. Conditional requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='5' (Predefined Filter Collections).
12.7.15 CIM_IndicationFilter (Disk Drive Creation)	Conditional	Experimental. Conditional requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='5' (Predefined Filter Collections). This is the 'pre-defined' CIM_IndicationFilter instance for the addition of a new DiskDrive instance.
12.7.16 CIM_IndicationFilter (Disk Drive Deletion)	Conditional	Experimental. Conditional requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='5' (Predefined Filter Collections). This is the 'pre-defined' CIM_IndicationFilter instance for the deletion of a DiskDrive instance.
12.7.17 CIM_MediaPresent	Mandatory	Associates a disk drive to its storage extent.

Table 252 - CIM Elements for Disk Drive Lite (Continued)

Element Name	Requirement	Description
12.7.18 CIM_MemberOfCollection (Disk Drive Lite Filter Collection to FilterCollection)	Conditional	Experimental. Conditional requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='5' (Predefined Filter Collections). This associates the Disk Drive Lite predefined FilterCollection to the FilterCollection for the autonomous profile (e.g., the Array FilterCollection).
12.7.19 CIM_MemberOfCollection (Predefined Filter Collection to Disk Drive Lite Filters)	Conditional	Experimental. Conditional requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='5' (Predefined Filter Collections). This associates the Disk Drive Lite predefined FilterCollection to the predefined Filters supported by the implementation.
12.7.20 CIM_PhysicalPackage	Mandatory	The physical package for the disk drive.
12.7.21 CIM_ProtocolControllerAccessesUnit	Optional	Deprecated. Associates an initiator protocol controller to the disk drive storage extent.
12.7.22 CIM_Realizes	Mandatory	Associates the disk drive to its physical package.
12.7.23 CIM_SAPAvailableForElement	Optional	Associates the target protocol endpoint to the disk drive.
12.7.24 CIM_SASPort (Disk Drive Target SAS Port)	Optional	Represents a SAS target port for the disk drive.
12.7.25 CIM_SCSIInitiatorTargetLogicalUnitPath	Optional	Associates protocol endpoints of the initiator and target ports to the extent that is exposed through the ports.
12.7.26 CIM_SCSIProtocolEndpoint (Disk Drive target SCSI Protocol Endpoint)	Optional	A target SCSI protocol endpoint for a disk drive if SCSI protocols are supported.
12.7.27 CIM_SPIPort (Disk Drive Target Parallel SCSI Port)	Optional	Represents a parallel SCSI target port for the disk drive.
12.7.28 CIM_SoftwareIdentity	Mandatory	Represents the firmware information for the disk drive.
12.7.29 CIM_StorageExtent (Primordial Disk Drive Extent)	Mandatory	The storage extent that represents the storage of the disk drive.
12.7.30 CIM_SystemDevice (Disk Drive System)	Mandatory	Associates DiskDrive to a hosting computer system.
12.7.31 CIM_SystemDevice (Port System)	Optional	Associates disk drive Ports to a hosting computer system.
12.7.32 CIM_SystemDevice (Storage Extent System)	Mandatory	Associates a StorageExtent to a hosting computer system.
12.7.33 SNIA_DiskDrive	Optional	This is a subclass of CIM_DiskDrive. CIM_DiskDrive may be subclassed as SNIA_DiskDrive to add additional properties.
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_DiskDrive	Mandatory	Addition of a new Disk Drive instance.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_DiskDrive	Mandatory	Deletion of a Disk Drive instance.

12.7.2 CIM_ATAPort (Disk Drive Target ATA Port)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 253 describes class CIM_ATAPort (Disk Drive Target ATA Port).

Table 253 - SMI Referenced Properties/Methods for CIM_ATAPort (Disk Drive Target ATA Port)

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	
OperationalStatus		Optional	
UsageRestriction		Mandatory	Shall be 2 for disk drive target ports.
PortType		Mandatory	Shall be 92 93 (SATA or SATA2) .

12.7.3 CIM_ATAProtocolEndpoint (Disk Drive target ATA Protocol Endpoint)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 254 describes class CIM_ATAProtocolEndpoint (Disk Drive target ATA Protocol Endpoint).

Table 254 - SMI Referenced Properties/Methods for CIM_ATAProtocolEndpoint (Disk Drive target ATA Protocol Endpoint)

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
Name		Mandatory	
Role		Mandatory	Shall be 3 (Target).
ProtocolIFType		Mandatory	
OtherTypeDescription		Mandatory	
ConnectionType		Mandatory	

12.7.4 CIM_AssociatedComponentExtent (Pool Component to Primordial Pool)

The referenced primordial disk drive StorageExtent represents capacity has not been allocated, is allocated in part, or is allocated in its entirety.

Requirement: Mandatory.

Table 255 describes class CIM_AssociatedComponentExtent (Pool Component to Primordial Pool).

Table 255 - SMI Referenced Properties/Methods for CIM_AssociatedComponentExtent (Pool Component to Primordial Pool)

Properties	Flags	Requirement	Description and Notes
GroupComponent		Mandatory	The Primordial StoragePool.
PartComponent		Mandatory	The disk drive storage extent that is a component of the primordial storage pool.

12.7.5 CIM_BasedOn (Bottom Level BasedOn)

Created By: External

Modified By: External

Deleted By: External

Requirement: Implementation of the Extent Composition profile.

Table 256 describes class CIM_BasedOn (Bottom Level BasedOn).

Table 256 - SMI Referenced Properties/Methods for CIM_BasedOn (Bottom Level BasedOn)

Properties	Flags	Requirement	Description and Notes
StartingAddress		Optional	This should be specified if the concrete extent does not use the whole disk drive extent.
EndingAddress		Optional	This should be specified if the concrete extent does not use the whole disk drive extent.
Dependent		Mandatory	This is a reference to the concrete storage extent.
Antecedent		Mandatory	This is a reference to the disk drive storage extent.

12.7.6 CIM_ConcreteComponent (Disk Extent to Primordial Pool)

Deprecated. Associates a disk drive extent to a primordial storage pool. This is Deprecated since its function is better covered by AssociatedComponentExtent.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Implementation of the Block Services Package.

Table 257 describes class CIM_ConcreteComponent (Disk Extent to Primordial Pool).

Table 257 - SMI Referenced Properties/Methods for CIM_ConcreteComponent (Disk Extent to Primordial Pool)

Properties	Flags	Requirement	Description and Notes
PartComponent		Mandatory	A reference to an instance of CIM_StorageExtent that represents the storage on the disk drive. The extent shall have its Primordial property set to true.
GroupComponent		Mandatory	A reference to an instance of CIM_StoragePool with the Primordial property set to true.

12.7.7 CIM_Container

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 258 describes class CIM_Container.

Table 258 - SMI Referenced Properties/Methods for CIM_Container

Properties	Flags	Requirement	Description and Notes
GroupComponent		Mandatory	A reference to an instance of CIM_PhysicalPackage that represents the higher level package that contains the disk drive package.
PartComponent		Mandatory	A reference to an instance of CIM_PhysicalPackage that represents the packaging for the disk drive.

12.7.8 CIM_DeviceSAPImplementation (ATA)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 259 describes class CIM_DeviceSAPImplementation (ATA).

Table 259 - SMI Referenced Properties/Methods for CIM_DeviceSAPImplementation (ATA)

Properties	Flags	Requirement	Description and Notes
Antecedent		Mandatory	A reference to an instance of an ATA port with a UsageRestriction property set to '2' (Target).
Dependent		Mandatory	A reference to an instance of an ATA protocol endpoint with a Role property set to '3' (Target).

12.7.9 CIM_DeviceSAPImplementation (SCSI)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 260 describes class CIM_DeviceSAPImplementation (SCSI).

Table 260 - SMI Referenced Properties/Methods for CIM_DeviceSAPImplementation (SCSI)

Properties	Flags	Requirement	Description and Notes
Antecedent		Mandatory	A reference to an instance of a parallel SCSI (SPI), SAS or FC port with a UsageRestriction property set to '2' (Target).
Dependent		Mandatory	A reference to an instance of a SCSI protocol endpoint with a Role property set to '3' (Target).

12.7.10 CIM_DiskDrive

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 261 describes class CIM_DiskDrive.

Table 261 - SMI Referenced Properties/Methods for CIM_DiskDrive

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	
Name		Mandatory	
OperationalStatus		Mandatory	Possible OperationalStatus values are 2 (OK), 5 (Predictive Failure), 6 (Error), 8 (Starting), 9 (Stopping) or 10 (Stopped).
EnabledState		Mandatory	Possible EnabledStates are 2 (Enabled), 3 (Disabled), 4 (Shutting Down), 6 (Enabled but Offline) or 10 (Starting). Enabled - drive is spun up and online. Disabled - drive is spun down, and offline. Shutting down - drive is spinning down. Enabled but Offline - drive is spun up but offline. Starting - drive is spinning up.
RequestedState		Optional	Possible RequestedStates are 2 (Enabled), 4 (Shutting Down) and 6 (Offline). Enabled - Spin up drive if it was spun down and Online the drive if it was offline. Shutting down - spin down drive. Offline - offline drive.
RequestStateChange()		Optional	

12.7.11 CIM_ElementSoftwareIdentity

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 262 describes class CIM_ElementSoftwareIdentity.

Table 262 - SMI Referenced Properties/Methods for CIM_ElementSoftwareIdentity

Properties	Flags	Requirement	Description and Notes
Antecedent		Mandatory	A reference to an instance of CIM_SoftwareIdentity that represents the software of the disk drive.
Dependent		Mandatory	A reference to an instance of CIM_DiskDrive or SNIA_diskdrive.

12.7.12 CIM_FCPort (Disk Drive Target FC Port)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 263 describes class CIM_FCPort (Disk Drive Target FC Port).

Table 263 - SMI Referenced Properties/Methods for CIM_FCPort (Disk Drive Target FC Port)

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	
OperationalStatus		Optional	
UsageRestriction		Mandatory	Shall be 2 for disk drive target ports.
PortType		Mandatory	Shall be 0 1 10 11 12 13 14 15 16 17 18 (Unknown or Other or N or NL or F/NL or Nx or E or F or FL or B or G).
PermanentAddress	CD	Mandatory	Port WWN. Shall be 16 unseparated uppercase hex digits.
SupportedCOS		Optional	
ActiveCOS		Optional	
SupportedFC4Types		Optional	
ActiveFC4Types		Optional	

12.7.13 CIM_FilterCollection (Disk Drive Lite Predefined FilterCollection)

Experimental. This is a collection of predefined IndicationFilters to which a client may subscribe. A Disk Drive Lite implementation shall indicate support for predefined FilterCollections by the SNIA_IndicationConfigurationCapabilities.FeaturesSupported = '5' (Predefined Filter Collections).

Requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='5' (Predefined Filter Collections).

Table 264 describes class CIM_FilterCollection (Disk Drive Lite Predefined FilterCollection).

**Table 264 - SMI Referenced Properties/Methods for CIM_FilterCollection
(Disk Drive Lite Predefined FilterCollection)**

Properties	Flags	Requirement	Description and Notes
InstanceID		Mandatory	Shall specify the unique identifier for an instance of this class within the Implementation namespace.
CollectionName		Mandatory	The value of CollectionName shall be 'SNIA:Disk Drive Lite'.

12.7.14 CIM_HostedCollection (System to predefined IndicationFilters)

Experimental.

Requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='5' (Predefined Filter Collections).

Table 265 describes class CIM_HostedCollection (System to predefined IndicationFilters).

**Table 265 - SMI Referenced Properties/Methods for CIM_HostedCollection
(System to predefined IndicationFilters)**

Properties	Flags	Requirement	Description and Notes
Dependent		Mandatory	Reference to the predefined FilterCollection for Disk Drive Lite.
Antecedent		Mandatory	Reference to the 'Top level' System.

12.7.15 CIM_IndicationFilter (Disk Drive Creation)

Experimental. This is the 'pre-defined' CIM_IndicationFilter instance for the addition of a new DiskDrive instance.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='5' (Predefined Filter Collections).

Table 266 describes class CIM_IndicationFilter (Disk Drive Creation).

Table 266 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Disk Drive Creation)

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	
CreationClassName		Mandatory	
SystemName		Mandatory	
Name		Mandatory	This shall be 'SNIA:Disk Drive Lite:DiskDriveCreation'.
SourceNamespace	N	Optional	Deprecated. For Predefined IndicationFilters in the Implementation Namespace this shall be NULL.

Table 266 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Disk Drive Creation)

Properties	Flags	Requirement	Description and Notes
SourceNamespaces	N	Optional	Experimental. For Predefined IndicationFilters in the Implementation Namespace this shall be NULL.
Query		Mandatory	SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_DiskDrive.
QueryLanguage		Mandatory	This should be 'DMTF:CQL' for CQL queries, but may be 'WQL' or 'SMI-S V1.0'. WQL and SMI-S V1.0 are deprecated in favor of 'DMTF:CQL'.
ElementName	N	Optional	This should be NULL for predefined indication filters.

12.7.16 CIM_IndicationFilter (Disk Drive Deletion)

Experimental. This is the 'pre-defined' CIM_IndicationFilter instance for the deletion of a DiskDrive instance.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures=5' (Predefined Filter Collections).

Table 267 describes class CIM_IndicationFilter (Disk Drive Deletion).

Table 267 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Disk Drive Deletion)

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	
CreationClassName		Mandatory	
SystemName		Mandatory	
Name		Mandatory	This shall be 'SNIA:Disk Drive Lite:DiskDriveDeletion'.
SourceNamespace	N	Optional	Deprecated. For Predefined IndicationFilters in the Implementation Namespace this shall be NULL.
SourceNamespaces	N	Optional	Experimental. For Predefined IndicationFilters in the Implementation Namespace this shall be NULL.
Query		Mandatory	SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_DiskDrive.
QueryLanguage		Mandatory	This should be 'DMTF:CQL' for CQL queries, but may be 'WQL' or 'SMI-S V1.0'. WQL and SMI-S V1.0 are deprecated in favor of 'DMTF:CQL'.
ElementName	N	Optional	This should be NULL for predefined indication filters.

12.7.17 CIM_MediaPresent

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 268 describes class CIM_MediaPresent.

Table 268 - SMI Referenced Properties/Methods for CIM_MediaPresent

Properties	Flags	Requirement	Description and Notes
Dependent		Mandatory	A reference to an instance of CIM_StorageExtent with the Primordial property set to true (a disk drive extent).
Antecedent		Mandatory	A reference to an instance of CIM_DiskDrive or SNIA_DiskDrive.

12.7.18 CIM_MemberOfCollection (Disk Drive Lite Filter Collection to FilterCollection)

Experimental. This associates the Disk Drive Lite predefined FilterCollection to the FilterCollection for the autonomous profile (e.g., the Array FilterCollection).

Requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='5' (Predefined Filter Collections).

Table 269 describes class CIM_MemberOfCollection (Disk Drive Lite Filter Collection to FilterCollection).

Table 269 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Disk Drive Lite Filter Collection to FilterCollection)

Properties	Flags	Requirement	Description and Notes
Collection		Mandatory	Reference to the Disk Drive Lite predefined FilterCollection.
Member		Mandatory	Reference to the Disk Drive Lite predefined FilterCollection.

12.7.19 CIM_MemberOfCollection (Predefined Filter Collection to Disk Drive Lite Filters)

Experimental. This associates the Disk Drive Lite predefined FilterCollection to the predefined Filters supported by the implementation.

Requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='5' (Predefined Filter Collections).

Table 270 describes class CIM_MemberOfCollection (Predefined Filter Collection to Disk Drive Lite Filters).

Table 270 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Predefined Filter Collection to Disk Drive Lite Filters)

Properties	Flags	Requirement	Description and Notes
Collection		Mandatory	Reference to the Disk Drive Lite predefined FilterCollection.
Member		Mandatory	Reference to the predefined IndicationFilters of the Disk Drive Lite implementation.

12.7.20 CIM_PhysicalPackage

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 271 describes class CIM_PhysicalPackage.

Table 271 - SMI Referenced Properties/Methods for CIM_PhysicalPackage

Properties	Flags	Requirement	Description and Notes
CreationClassName		Mandatory	
Tag		Mandatory	
Manufacturer		Mandatory	
Model		Mandatory	
SerialNumber		Optional	
PartNumber		Optional	

12.7.21 CIM_ProtocolControllerAccessesUnit

Deprecated.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 272 describes class CIM_ProtocolControllerAccessesUnit.

Table 272 - SMI Referenced Properties/Methods for CIM_ProtocolControllerAccessesUnit

Properties	Flags	Requirement	Description and Notes
Dependent		Mandatory	A reference to an instance of CIM_StorageExtent with the Primordial property set to true (the disk drive extent).
Antecedent		Mandatory	A reference to a CIM_ProtocolController (from the Initiator for this disk drive).

12.7.22 CIM_Realizes

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 273 describes class CIM_Realizes.

Table 273 - SMI Referenced Properties/Methods for CIM_Realizes

Properties	Flags	Requirement	Description and Notes
Antecedent		Mandatory	A reference to an instance of a physical package that represents the packaging for the disk drive.
Dependent		Mandatory	A reference to an instance of CIM_DiskDrive or SNIA_DiskDrive.

12.7.23 CIM_SAPAvailableForElement

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 274 describes class CIM_SAPAvailableForElement.

Table 274 - SMI Referenced Properties/Methods for CIM_SAPAvailableForElement

Properties	Flags	Requirement	Description and Notes
AvailableSAP		Mandatory	A reference to an instance of a SCSI or ATA protocol endpoint that represents the target endpoint (role='3') for the disk drive.
ManagedElement		Mandatory	A reference to an instance of a Disk Drive or SNIA_DiskDrive.

12.7.24 CIM_SASPort (Disk Drive Target SAS Port)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 275 describes class CIM_SASPort (Disk Drive Target SAS Port).

Table 275 - SMI Referenced Properties/Methods for CIM_SASPort (Disk Drive Target SAS Port)

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	
OperationalStatus		Optional	
UsageRestriction		Mandatory	Shall be 2 for disk drive target ports.
PermanentAddress		Mandatory	SAS Address. Shall be 16 un-separated upper case hex digits.
PortType		Mandatory	Shall be 94 (SAS).

12.7.25 CIM_SCSIInitiatorTargetLogicalUnitPath

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 276 describes class CIM_SCSIInitiatorTargetLogicalUnitPath.

Table 276 - SMI Referenced Properties/Methods for CIM_SCSIInitiatorTargetLogicalUnitPath

Properties	Flags	Requirement	Description and Notes
Initiator		Mandatory	The protocol endpoint for the back end initiator port for accessing the disk drive.
Target		Mandatory	A reference to an instance of a SCSI or ATA protocol endpoint that represents the target endpoint (role='3') for the disk drive.
LogicalUnit		Mandatory	Shall reference the StorageExtent associated to the DiskDrive.

12.7.26 CIM_SCSIProtocolEndpoint (Disk Drive target SCSI Protocol Endpoint)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 277 describes class CIM_SCSIProtocolEndpoint (Disk Drive target SCSI Protocol Endpoint).

Table 277 - SMI Referenced Properties/Methods for CIM_SCSIProtocolEndpoint (Disk Drive target SCSI Protocol Endpoint)

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
Name		Mandatory	
Role		Mandatory	Shall be 3 (Target).
ProtocolIFType		Mandatory	
OtherTypeDescription		Mandatory	
ConnectionType		Mandatory	

12.7.27 CIM_SPIPort (Disk Drive Target Parallel SCSI Port)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 278 describes class CIM_SPIPort (Disk Drive Target Parallel SCSI Port).

**Table 278 - SMI Referenced Properties/Methods for CIM_SPIPort
(Disk Drive Target Parallel SCSI Port)**

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	
OperationalStatus		Optional	
UsageRestriction		Mandatory	Shall be 2 for disk drive target ports.
PortType		Mandatory	Shall be 101 (SCSI Parallel).

12.7.28 CIM_SoftwareIdentity

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 279 describes class CIM_SoftwareIdentity.

Table 279 - SMI Referenced Properties/Methods for CIM_SoftwareIdentity

Properties	Flags	Requirement	Description and Notes
InstanceID		Mandatory	
VersionString		Mandatory	
Manufacturer		Optional	
BuildNumber		Optional	
MajorVersion		Optional	
RevisionNumber		Optional	
MinorVersion		Optional	

12.7.29 CIM_StorageExtent (Primordial Disk Drive Extent)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 280 describes class CIM_StorageExtent (Primordial Disk Drive Extent).

**Table 280 - SMI Referenced Properties/Methods for CIM_StorageExtent
(Primordial Disk Drive Extent)**

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	
BlockSize		Mandatory	
NumberOfBlocks		Mandatory	The number of blocks as reported by the hardware.
ConsumableBlocks		Mandatory	The number of usable blocks.
Primordial		Mandatory	Shall be true.
ExtentStatus		Mandatory	
OperationalStatus		Mandatory	
ExtentDiscriminator		Mandatory	Experimental. This is an array of values that shall contain 'SNIA:Pool Component' and 'SNIA:DiskDrive'.

12.7.30 CIM_SystemDevice (Disk Drive System)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 281 describes class CIM_SystemDevice (Disk Drive System).

Table 281 - SMI Referenced Properties/Methods for CIM_SystemDevice (Disk Drive System)

Properties	Flags	Requirement	Description and Notes
GroupComponent		Mandatory	A reference to an instance of Computer System.
PartComponent		Mandatory	A reference to an instance of CIM_DiskDrive or SNIA_DiskDrive used in this profile.

12.7.31 CIM_SystemDevice (Port System)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 282 describes class CIM_SystemDevice (Port System).

Table 282 - SMI Referenced Properties/Methods for CIM_SystemDevice (Port System)

Properties	Flags	Requirement	Description and Notes
GroupComponent		Mandatory	A reference to an instance of Computer System.
PartComponent		Mandatory	A reference to an instance of CIM_FCPort, CIM_SPIPort, CIM_SASPort or CIM_ATAPort used in this profile.

12.7.32 CIM_SystemDevice (Storage Extent System)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 283 describes class CIM_SystemDevice (Storage Extent System)

Table 283 - SMI Referenced Properties/Methods for CIM_SystemDevice (Storage Extent System)

Properties	Flags	Requirement	Description and Notes
GroupComponent		Mandatory	A reference to an instance of Computer System.
PartComponent		Mandatory	A reference to an instance of CIM_StorageExtent used in this profile.

12.7.33 SNIA_DiskDrive

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 284 describes class SNIA_DiskDrive.

Table 284 - SMI Referenced Properties/Methods for SNIA_DiskDrive

Properties	Flags	Requirement	Description and Notes
DiskType		Mandatory	The technology employed to store data. DiskType values are 0 (Unknown), 1 (Other), 2 (Hard Disk Drive) or 3 (Solid State Disk).
FormFactor		Mandatory	The Physical size of the disk drive. FormFactor values are 0 (Unknown), 1 (Other), 2 (Not Reported), 3 (5.25 inch), 4 (3.5 inch), 5 (2.5 inch), 6 (1.8 inch).
Encryption		Mandatory	This property reflects the state of the encryption feature implemented by some disk drives. Encryption values are 0 (Unknown), 1 (Not Supported), 2 (unlocked) or 3 (locked).

STABLE

IMPLEMENTED

13 Disk Sparing Subprofile

13.1 Description

13.1.1 General

Many block service systems enhance availability by providing backup storage capacity to be used in place of a failed component. The failure of the component may be caused by the failure of a physical component that realizes that component, or the invalidation or corruption of the component itself.

The end result of the failure is that the block server is degraded by performance or spare redundancy. In the first case, it is important that the cause of the performance degradation is known so the appropriate response may be taken. In the second case, the administrator will have to know of the loss of redundancy. The administrator can then plan to replace the used redundancy and fix the broken component. A sample instance diagram is provided in Figure 61: "Sparing Instance Diagram".

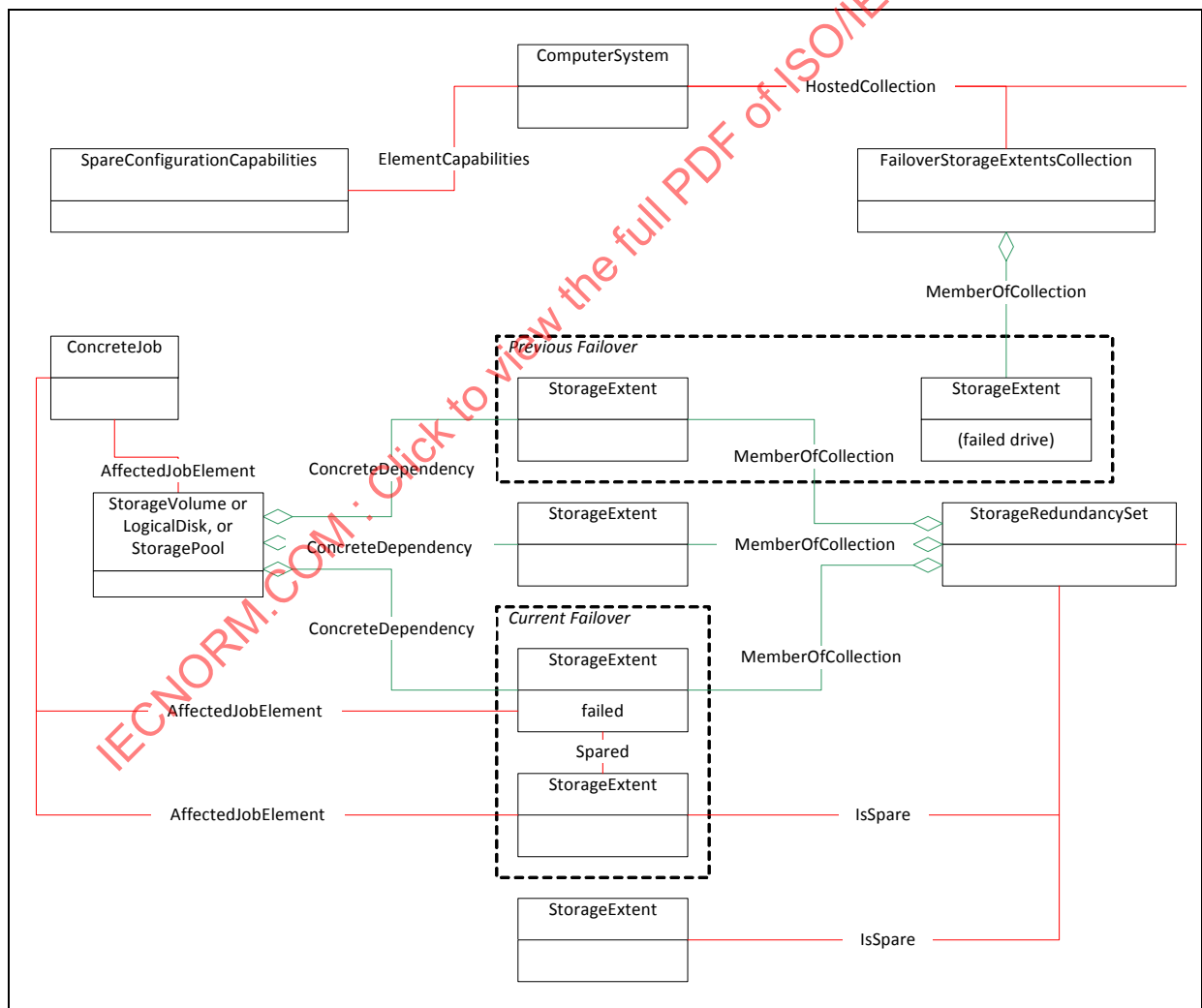


Figure 61 - Sparing Instance Diagram

15 "Extent Composition Subprofile" focuses on the mapping of storage to storage elements, StorageVolume and LogicalDisk. This subprofile enhances that picture by representing how spare physical storage components like disk drives or purely logical constructs like LUNs or

even host partitions, can be used to provide redundancy for storage elements. The spare elements are represented as StorageExtents themselves.

12 "Disk Drive Lite Subprofile" can be used to supplement this subprofile by explicitly listing the changes in operational status resulting from the failure of disks and the affect of this failure on the StorageVolumes or LogicalDisks they support. In conjunction with *Information Technology – Storage Management, Part 3 Common Profiles*, Clause 26 Health Package and the RelatedElementCausingError association, a client can tell, unambiguously the effect and cause of the storage component failure.

Fail Over is the name of the process by which the capacity provided by one StorageExtent is replaced by that of the spare StorageExtent. The block contents of the original StorageExtent is copied to the replacement StorageExtent. During this process a ConcreteJob shall be created to represent this process and report the progress and status of the fail over.

The functionality provided by this subprofile includes:

- The representation of the current state of the spares whether they are not in use, are in use, or in transition from not in use to being put into service. All three of these states can be present at once.
- The detection of the addition of another spare element and whether the implementation requires client intervention to assign the spare element.
- Client initiated fail over. A client may cause the fail over process to start.
- Client initiated rebuild of Extent data.
- Client initiated check and rebuild of Extent parity.

13.1.2 Durable Names and Correlatable IDs of the Profile

The StorageVolumes are required to provide the correlatable ID, Name. See *Information Technology – Storage Management, Part 2 Common Architecture*, 8.2, "Guidelines for SCSI Logical Unit Names".

13.1.3 Sparing Model

StorageExtents are used as the unit of redundancy in this model. StorageExtents can be said to be a grouping of capacity. For the question of what component of the system has failed, the StorageExtent should be realized by a DiskDrive or some of component to which the failure is meaningful. This model represents how the capacity is used in the protection of the data. Other models define how StorageExtents are realized by other components or devices.

A *spare* is, functionally, the union of the StorageExtent representation and the associated component representation that realizes the Extent. This subprofile uses this term in this union.

The sparing model provides for mechanisms to:

- Group StorageExtents that have failed.
- Group spares that can be used to replace failed components. The group of spares may be shared across StorageVolumes, LogicalDisks, or StoragePools.
- Report what component is being spared or replaced by the spare.
- Report the process of a fail over, sparing reconfiguration, storage extent rebuild, or parity check.
- Report the capabilities of the Sparing implementation.

The physical resources on which a StorageExtent is realized are components that may result in data loss if they fail. If the physical resource is modeled, its storage shall be represented by a primordial StorageExtent. This profile requires that the physical resource on which a spare extent is realized be identifiable. As a consequence, if a StorageExtent is used as a spare, it

shall either be a primordial extent, or it shall have a ConcreteDependency association to one or more antecedent primordial StorageExtents.

The StorageRedundancySet class is used to group spares. There may be a single StorageRedundancySet per StorageVolume or LogicalDisk. Multiple StorageVolumes or LogicalDisks may share a single StorageRedundancySet. In the first case, the spares grouping can be said to be *dedicated* to that StorageVolume or LogicalDisk. In the second case, the spares grouping can be said to be *global*. That is, the spares will be used for all the StorageVolumes or LogicalDisks that are associated to a StorageRedundancySet. This is illustrated in Figure 62: "Variations of RS per Storage Element".

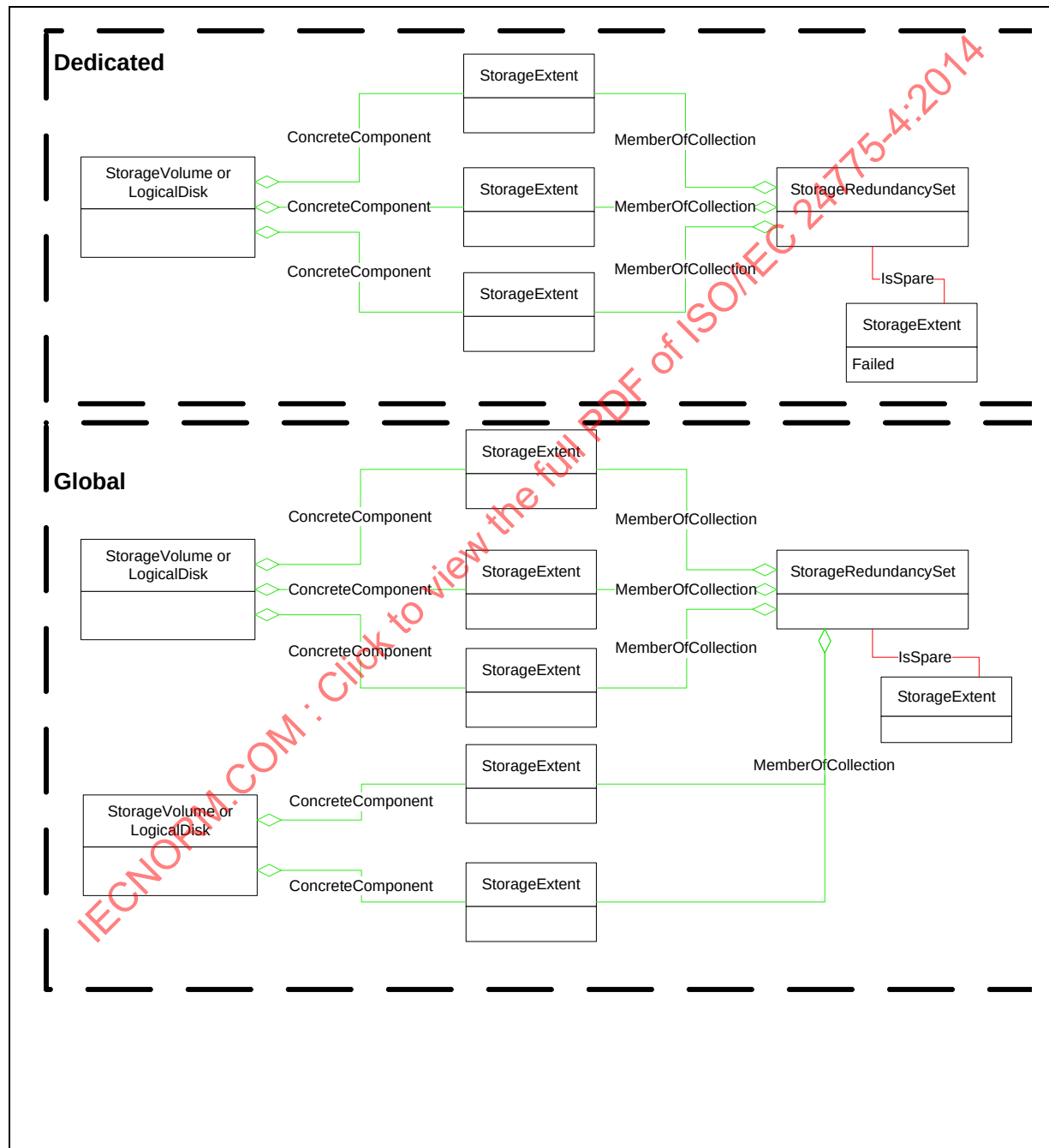


Figure 62 - Variations of RS per Storage Element

In the case where spares are not dedicated, the decision to group Extents with a given StorageRedundancySet depends on the rules of the implementation. Some implementations require particular types of spares to be used together. For example, some implementations may require that a DiskDrive is spared by another DiskDrive of the same size and/or type. This

profile does not model DiskDrives. To implement this case, the implementer would model the StorageExtent associated to the DiskDrive, a StorageRedundancySet, and associate StorageExtents to that StorageRedundancySet that share the characteristics, whatever they may be, that permit these StorageExtents to be used as spares. If an implementation supports such rules then a StorageRedundancySet shall be created per rule. When StorageVolumes or LogicalDisk are created or modified, the implementation can select the StorageRedundancySet to associate to the created or modified storage element using on the PackageRedundancy Goal. An implementation that supports *global* spares that supported both the Clause 6, "Block Services Package" and this subprofile, would match this Goal with StorageRedundancySet that had at least that number of spares.

A StoragePool, StorageVolume, or LogicalDisk may be have one or more StorageExtents that provide redundancy of its data. Storage elements for which this is the case shall participate in a ConcreteDependency association with the StorageExtents that form its redundancy. These StorageExtents shall participate MemberOfCollection associations to a RedundancySet. In turn, the reference RedundancySet shall indicate the status of the redundancy. The StorageExtents that be used to replace a StorageExtent whose realization has failed shall be associated to this StorageRedundancySet via an IsSpare association. Once the substitution of the failed StorageExtent for the spare StorageExtent started, the failed StorageExtent shall be associated to the spare StorageExtent via the Spared association. This shall be the case until the process of substitution has completed. After which, the failed StorageExtent shall participate in a MemberOfCollection with a FailoverStorageExtentsCollection but not participate in a MemberOfCollection association with a StorageRedundancySet nor in a ConcreteDependency association with any storage element. The failed StorageExtents are removed from the FEC when the failed component on which they are based is removed from the system through a means not defined in this profile, i.e., the drive FRU (Field Replaceable Unit) pulled from the array.

The FailoverExtentsCollection class is used to collect the spares that have failed. These are the components that need to be diagnosed, repaired, and, possibly, replaced or assigned to the primordial StoragePool.

The StorageConfigurationCapabilities class is used to report the capabilities of the implementation. Not all sparing functionality is required. This class is used to report what methods are implemented. The properties and methods of the class are specified later in this profile. Table 285 below lists the action names for the sparing methods. If a sparing method is supported synchronously, then the action name for the method shall be present in the SupportedSynchronousActions array. If a sparing method is supported asynchronously, then the action name for the method shall be present in the SupportedAsynchronousActions array.

Table 285 - Supported Methods to Method Mapping

Action	Method
Assign Spares	SpareConfigurationService.AssignSpares
Unassign Spares	SpareConfigurationService.UnassignSpares
Rebuild Storage Extent	SpareConfigurationService.RebuildStorageExtent
Check Parity Consistency	SpareConfigurationService.CheckParityConsistency
Repair Parity	SpareConfigurationService.RepairParity
Fail Over	StorageRedundancySet.Failover

13.1.4 Modeling Fail Over, Past and Present

This subclause illustrates the requirements for modeling spare fail over in three cases, before the failure, during the fail over, and after the fail over.

Figure 63: "Before Failure" shows a dedicated RedundancySet with a single spare.

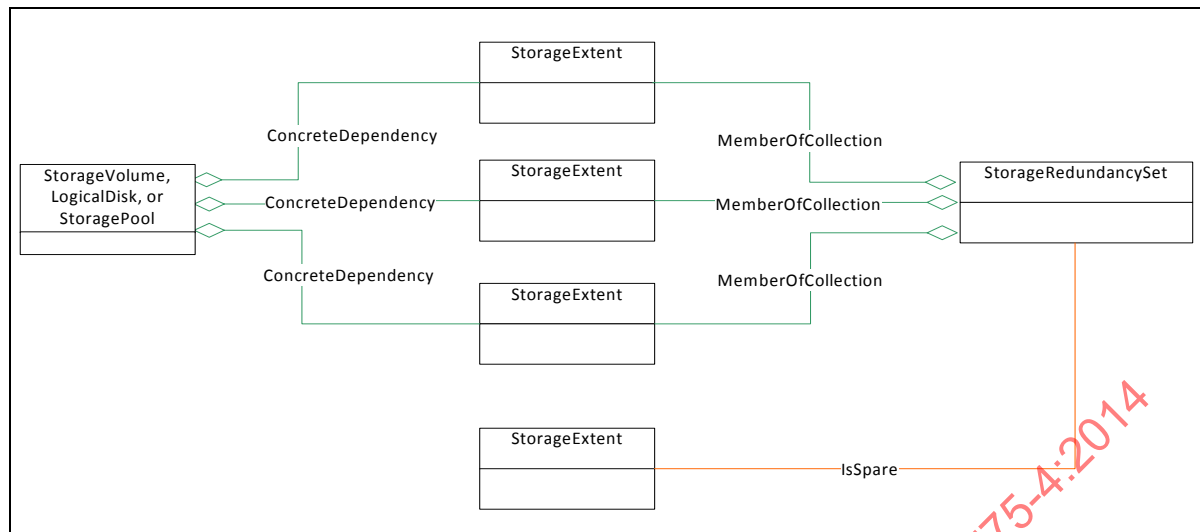


Figure 63 - Before Failure

Once the failure has occurred, a *ConcreteJob* is created to represent the fail over process, as shown in Figure 64: "During Failure".

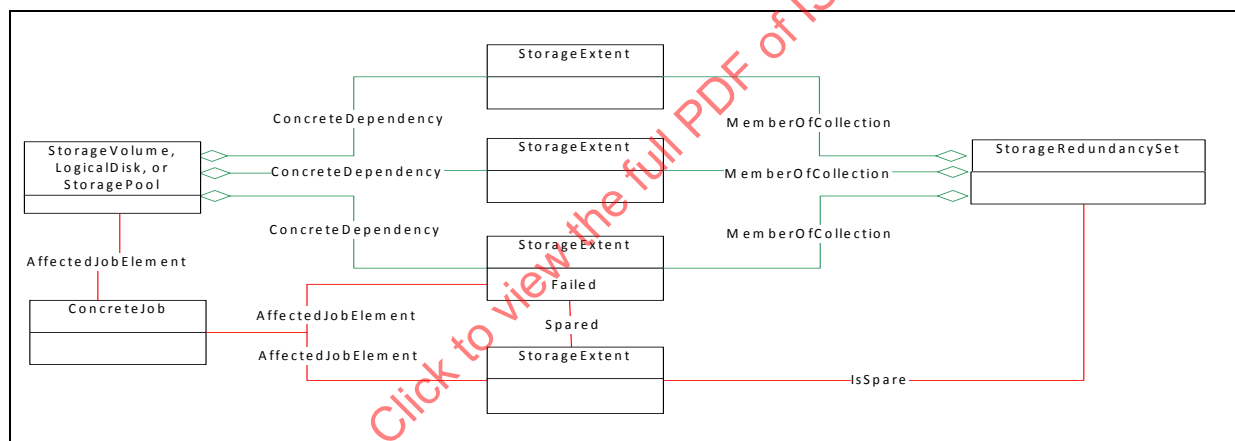


Figure 64 - During Failure

The *AffectJobElement* association shall associate the *LogicalDisk* or *StorageVolume* that is being failed over, the *StorageExtent* that has failed and is causing the fail over, and the spare *StorageExtent*. The associations shall remain for some period of time as per the rules in the *Information Technology – Storage Management, Part 3 Common Profiles*, Clause 27 "Job Control Subprofile". For these rules consider the two extents as Input values to the *StorageRedundancySet.Failover()* method.

This subprofile supports fail over initiated by the implementation or by the client. So that an observer can tell what this fail over *ConcreteJob* is doing, the implementation shall model the *ConcreteJob* as if another client initiated the fail over, even though the implementation did the initiation. In other words, the *ConcreteJob* shall be associated to the *StorageRedundancySet* associated to the two Extents in question via the *OwningJobElement* association. The *MethodResult* instance, as defined in *Information Technology – Storage Management, Part 3 Common Profiles*, Clause 27 "Job Control Subprofile", shall contain the *StorageRedundancySet.Failover()* method name and parameters.

Once the fail over is complete, the failed Extent shall no longer have a *ConcreteDependency* association to *StorageVolume* or *LogicalDisk* that was once based on it. The spare *StorageExtent* shall now participate in a *MemberOfCollection* associated to the *StorageRedundancySet* instead of the *IsSpare* association. The failed over Volume or

LogicalDisk shall now participate in a ConcreteDependency relationship with the spare Extent. The failed Extent may now participate in a MemberOfCollection association with the FailoverStorageExtentsCollection, illustrated in Figure 65: "After Failure".

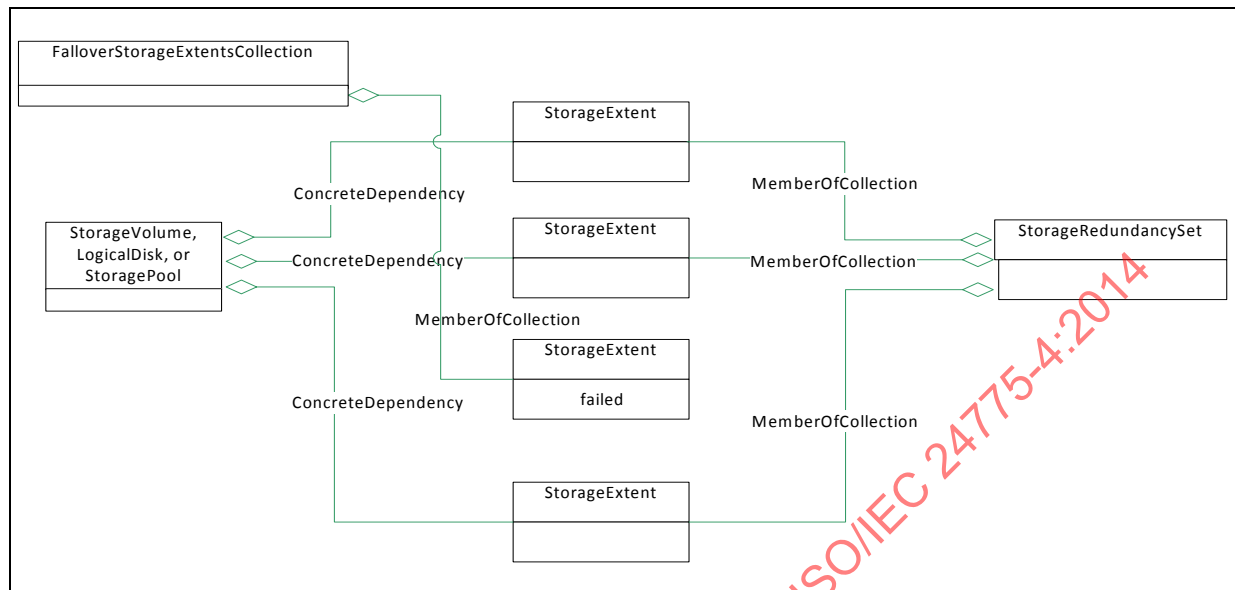


Figure 65 - After Failure

EXPERIMENTAL

13.1.5 Sparing Configuration and Control

All six methods defined or used in this subprofile, **AssignSpares**, **UnassignSpares**, **RebuildStorageExtent**, **CheckParityConsistency**, **CheckStorageElement**, and **RepairParity** can be initiated by the implementation or the client. If the method execution is not instantaneous, then information about what method invocation gave rise to the job follows the rules in *Information Technology – Storage Management, Part 3 Common Profiles*, Clause 27 "Job Control Subprofile". These methods can also be initiated by the implementation itself. The implementation shall represent the execution of the job, job name, and method parameters in said manner even it initiated the Job. If the implementation supports this functionality but does not allow the client to initiate the action, it shall still represent the execution of the functionality, as represented by a method execution, in said manner.

The purpose of these rules is to allow an observer to tell that, for example, a **RepairParity** task is executing.

EXPERIMENTAL

13.2 Health and Fault Management Considerations

One of the primary reasons for this subprofile is to allow a client to determine if the cause of performance degradation of a block server is caused by spare fail over, volume rebuild, or parity repair.

There are several failure cases possible with this subprofile:

- There may be failures of the several configuration and control methods of this subprofile for reasons other than the parameters provided by the client.

The **StorageExtents** used in the configuration and control methods may be invalid.

13.3 Cascading Conjunctions

Not defined in this International Standard.

13.4 Supported Subprofiles and Packages

Table 286 describes the supported profiles for Disk Sparing.

Table 286 - Supported Profiles for Disk Sparing

Profile Name	Organization	Version	Requirement	Description
Job Control	SNIA	1.5.0	Mandatory	

13.5 Methods of the Profile

EXPERIMENTAL

13.5.1 AssignSpares

```
uint32 AssignSpares(
    [Out] CIM_ConcreteJob REF Job
    [In] CIM_StoragePool REF InPool
    [In] CIM_StorageExtent REF InExtents[]
    [In] CIM_StorageRedundancySet REF
    RedundancySet)
```

This method is used to assign spares to a particular RedundancySet. If there is more than one StoragePool in this implementation, then the arguments to the method shall contain the references to StorageExtents and references to the primordial StoragePools of which they are components. This method shall not permit the assignment of spare from more than one StoragePool.

This method may return the following error codes. Many of the return codes are used widely and documented in CIM. The following documents the return codes that are unique to this method. This method shall not return vendor specific return codes.

```
ValueMap { "0", "1", "2", "3", "4", "5", "6", "..", "4096",
    "4097", "4098", "4099", "4100..32767", "32768..65535" },
Values { "Job Completed with No Error", "Not Supported",
    "Unknown", "Timeout", "Failed", "Invalid Parameter",
    "In Use", "DMTF Reserved",
    "Method Parameters Checked - Job Started",
    "Multiple StoragePools",
    "Spares Are Not Compatible",
    "StorageExtent is in use",
    "Method Reserved", "Vendor Specific" }
```

- 4097, "Multiple StoragePools", means the client passed Extents that are components of more than one Primordial StoragePool.
- 4098, "Spares Are Not Compatible", means the client pass Extents than may not be used together. There is no mechanism at this time to tell a client, through the model, what spares can be used together.
- 4099, "StorageExtent is in use", means that one or more of the Extents passes are already in use as a spare or as part of a StorageVolume or LogicalDisk.

13.5.2 UnassignSpares

```
uint32 UnassignSpares(
    [Out] CIM_ConcreteJob REF Job
    [In] CIM_StoragePool REF InPool
    [In] CIM_StorageExtent REF InExtents[])
```

This method is used to remove a spare from a StorageRedundancy and also unassign that Extent as a spare. The unassigned spare may end up as a member of the FailoverStorageExtentsCollection. The rules for the parameters and the same descriptions of assign spares are true for the parameters and return codes shared between the two method definitions. This method shall not return vendor specific return codes.

13.5.3 GetAvailableSpareExtents

```
uint32 GetAvailableExtents(
    [In] CIM_StoragePool REF InPool<
    [In] CIM_StorageRedundancySet REF
    RedundancySet,
    [Out] CIM_StorageExtent REF
    AvailableExtents[])
```

This method returns references of available StorageExtents that may be as spares for the given StorageRedundancySet and StoragePool. The referenced StorageRedundancySet shall provide redundancy for the referenced StoragePool.

The method may return error codes. Many of the return codes are used widely and documented in CIM. There are no return codes that are unique to this method. This method shall not return vendor specific return codes.

13.5.4 FailOver

```
uint32 Failover(
    [In] CIM_ManagedElement REF FailoverFrom
    [In] CIM_ManagedElement REF FailoverTo)
```

This method is used to force a failover between StorageExtents. The FailoverFrom reference shall be a reference to a StorageExtent that participates in a MemberOfCollection association with the StorageRedundancySet instance on which this method is called. The FailoverTo reference shall be a reference to a StorageExtent that participates in a IsSpare association with the StorageRedundancySet instance on which this method is called.

This method may return the follow error codes. Many of the return codes are used widely and documented in CIM. The following documents that return code semantics that are unique to this method.

```
ValueMap { "0", "1", "2", "3", "4", "..", "32768..65535" },
Values { "Completed with No Error", "Not
Supported",
"Unknown/Unspecified Error", "Busy/In Use",
"Parameter Error", "DMTF Reserved", "Vendor
Reserved" }]
```

- 3, "Unknown/Unspecified Error", means that the implementation failed to failover for some unspecified reason.
- 4, "Busy/In use", means that the failover between the reference StorageExtents is already in progress.

13.5.5 RebuildStorageExtent

```
uint32 RebuildStorageExtent(
    [Out] CIM_ConcreteJob REF Job
    [In] CIM_StorageExtent REF Target)
```

This method is used to rebuild the data distribution on the passed Extent with the other member Extents associated to a single StorageRedundancySet. If the Job execution fails, then

use `ConcreteJob.GetError()` to get the `CIM_Error` that states what the error was. In this case, the Target Extent shall report the appropriate, non "OK", `OperationalStatus`.

The method may return the following error codes. Many of the return codes are used widely and documented in CIM. The following documents the return codes that are unique to this method. This method shall not return vendor specific return codes.

```
ValueMap { "0", "1", "2", "3", "4", "5", "6", "..", "4096",
            "4097", "4098", "4099..32767", "32768..65535" },
Values { "Job Completed with No Error", "Not Supported",
         "Unknown", "Timeout", "Failed", "Invalid Parameter",
         "In Use", "DMTF Reserved",
         "Method Parameters Checked - Job Started",
         "Target is Not a Member of a StorageRedundancySet",
         "Rebuild already in Progress",
         "Method Reserved", "Vendor Specific" }
```

- 4097 "Target is Not a Member of a StorageRedundancySet", means that the Extent passed is not a member of StorageRedundancySet
- 4098 "Rebuild already in Progress", means that a rebuild of the data and/or parity on the passed Extent or one or more of the other member Extents of the same StorageRedundancySet is already in progress.

13.5.6 CheckParityConsistency

```
uint32 CheckParityConsistency(
    [Out] CIM_ConcreteJob REF Job
    [In] CIM_StorageExtent REF Target)
```

This method is used to check the consistency of the parity distribution on the passed Extent with the other member Extents associated to a single StorageRedundancySet. If the Job execution fails, then use `ConcreteJob.GetError()` to get the Error that states what the error was. In this case, the Target Extent shall report the appropriate, non "OK", `OperationalStatus`. If method execution determines that the parity is inconsistent, the ConcreteJob shall report successful completion and one of Operational Statuses of the passed Extent shall be 6 "Error".

The method may return the following error codes. Many of the return codes are used widely and documented in CIM. The following documents the return codes that are unique to this method. This method shall not return vendor-specific return codes.

```
ValueMap { "0", "1", "2", "3", "4", "5", "6", "..", "4096",
            "4097", "4098", "4099..32767", "32768..65535" },
Values { "Job Completed with No Error", "Not Supported",
         "Unknown", "Timeout", "Failed", "Invalid Parameter",
         "In Use", "DMTF Reserved",
         "Method Parameters Checked - Job Started",
         "Consistency Check Already in Progress",
         "No Parity to Check",
         "Method Reserved", "Vendor Specific" }
```

- 4097 "Consistency Check Already in Progress", means that a check and rebuild of the data parity on the passed Extent or one or more of the other member Extents of the same StorageRedundancySet is already in progress.
- 4098 "No Parity to Check", means that the member Extents of the StorageRedundancySet are not built with parity distribution. Recheck the Virtualization modeled.

13.5.7 RepairParity

```
uint32 RepairParity(
    [In] CIM_ConcreteJob REF Job,
    [Out] CIM_StorageExtent REF Target)
```

This method is used to rebuild of the parity distribution on the passed Extent with the other member Extents associated to a single StorageRedundancySet. The intent is that this method would be run after finding out that the CheckParityConsistency() reported that the Extent pair is inconsistent. If the Job execution fails, then use ConcreteJob.GetError() to get the Error that states what the error was. In this case, the Target Extents shall report the appropriate, non "OK", OperationalStatus and HealthState.

The method may return error codes. Many of the return codes are used widely and documented in CIM. There are no return codes that are unique to this method. This method shall not return vendor specific return codes.

13.5.8 CheckStorageElement

```
uint32 CheckStorageElement(
    [In
        Values {"Default", "Parity", "Bad Block",
        "Replication"}
        ValueMap{"1","2","3","4"}]
    uint16 CheckType,
    [In
        Values {"Run One Time", "Continuous"}
        ValueMap{"1","2"}]
    uint16 CheckMode,
    [In] CIM_LogicalElement REF TargetElement,
    [Out] CIM_ConcreteJob REF Job)
```

This method requests that the reference target element be checked with a given check type and with a given check mode. If a check mode of 1 "Run One Time" is requested, then the element check shall run once. If a check mode of 2 "Continuous", then the element shall be checked and checked again until the ConcreteJob instance, referenced by the Job parameter, is terminated.

The method may return the following error codes. Many of the return codes are used widely and documented in CIM. The following documents the return codes that are unique to this method. This method shall not return vendor specific return codes.

```
ValueMap { "0", "1", "2", "3", "4", "5", "6", "..", "4096",
    "4097", "4098", "4099..32767", "32768..65535" },
Values { "Job Completed with No Error", "Not Supported",
    "Unknown", "Timeout", "Failed", "Invalid Parameter",
    "In Use", "DMTF Reserved",
    "Method Parameters Checked - Job Started",
    "Storage Element Check Already in Progress",
    "Method Reserved", "Vendor Specific" }
```

- 4097 "Storage Element Check Already in Progress", means that a check on the passed Extent or one or more of the other member Extents of the same StorageRedundancySet is already in progress.

EXPERIMENTAL

13.6 Client Considerations and Recipes

13.6.1 General

The sparing implementation may cause the sparing configuration changes (i.e., jobs start and run) on its own in response to other clients.

The number of StorageRedundancySets may change over time because the physical components, realizing the spare StorageExtent, like disk drives are added or removed from the block server. Additionally, purely logical realizations of the spare StorageExtent may change as well. The StorageRedundancySets themselves once empty may remain in the model, but be empty, or may be removed from the model entirely for this or other reasons.

The sparing implementation shall report the correct RedundancyStatus, either 'Unknown' 0, 'Redundant' 1, or 'Redundancy Lost' 2. See property description (13.6.2) for details.

13.6.2 Determine if spare model is constructed correctly

```
// DESCRIPTION
// The goal of this recipe is to verify that the Sparing model
// is correctly instantiated.
// This type of instance traversal would be used by a client
// to determine if a particular storage element has spare
// coverage.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
// 1.A reference to a storage element, either a StorageVolume,
// a LogicalDisk, or a StoragePool, is previously defined in the
// $StorageElement-> variable

$SparedExtents->[] =
    AssociatorNames($StorageElement->,
        "CIM_ConcreteDependency",
        "CIM_StorageExtent",
        "Dependent", "Antecedent")
for i in $SparedExtents->[] {
    #RedundancySets->[] =
        AssociatorNames($SparedExtents->[#i],
            "CIM_MemberOfCollection",
            "CIM_StorageRedundancySet",
            "Member", "Collection")
    // We should find at least one RS per spared SE
    if(1 > #RedundancySets.length) {
        <ERROR! There should be at least one RedundancySet per spared
            StorageExtent>
    }
    for j in RedundancySets->[] {
        #SpareSEs->[] =
            AssociatorNames($RedundancySets->[#j],
                "CIM_IsSpare",
                "CIM_StorageExtent",
                "Dependent", "Antecedent") // SRE has the Dependent role
        if (0 < #SpareSEs->[]) {
            <EXIT: Successfully found at least one spare StorageExtent
        }
        else {
            <ERROR! The SRE associated to the subject StorageElement
                must have at least one Spare>
        }
    }
}
```

```

}
<ERROR! At least one Spared Extent MUST have been found.
If one or more was found, an successful exit would have occurred
before this point in the code.>

```

13.7 Registered Name and Version

Disk Sparing version 1.5.0 (Component Profile).

13.8 CIM Elements

13.8.1 General

Table 287 describes the CIM elements for Disk Sparing.

Table 287 - CIM Elements for Disk Sparing

Element Name	Requirement	Description
13.8.2 CIM_AssociatedComponentExtent (Spare to Storage Pool)	Conditional	Conditional requirement: Implementation of the Extent Composition profile.
13.8.3 CIM_ConcreteDependency (Extent to LogicalDisk)	Conditional	Conditional requirement: Referenced from Volume Management - LogicalDisk is mandatory. Represents the group of StorageExtents that form the redundancy of a LogicalDisk.
13.8.4 CIM_ConcreteDependency (Extent to Pool)	Mandatory	Represents the group of StorageExtents that form the redundancy of a StoragePool.
13.8.5 CIM_ConcreteDependency (Extent to StorageVolume)	Conditional	Conditional requirement: Referenced from Array - StorageVolume is mandatory or Referenced from Storage Virtualizer - StorageVolume is mandatory. Represents the group of StorageExtents that form the redundancy of a StorageVolume.
13.8.6 CIM_ElementCapabilities	Optional	Associates SpareConfigurationCapabilities with the Block Server's ComputerSystem instance.
13.8.7 CIM_HostedCollection (ComputerSystem to FailoverStorageExtentsCollection)	Optional	Associates FailoverStorageExtentsCollection with the Block Server's ComputerSystem instance.
13.8.8 CIM_HostedCollection (ComputerSystem to RedundancySet)	Mandatory	Associates StorageRedundancySet with the Block Server's ComputerSystem instance.
13.8.9 CIM_HostedService (ComputerSystem to SpareConfigurationService)	Optional	Associates SpareConfigurationService with the Block Server's ComputerSystem instance.
13.8.10 CIM_IsSpare	Mandatory	Represents the spare that may be used as a spare for any StorageExtents that is not a spare.
13.8.11 CIM_LogicalDisk	Conditional	Conditional requirement: Referenced from Volume Management - LogicalDisk is mandatory.
13.8.12 CIM_MemberOfCollection	Mandatory	Represents the relationship between the StorageExtents that form the redundancy of a StoragePool, StorageVolume, or LogicalDisk.
13.8.13 CIM_Spared	Mandatory	Represents the relationship between the spare and the StorageExtent that has failed and is being spared.
13.8.14 CIM_StorageExtent (Spare)	Mandatory	Represents the redundant or spare capacity.

Table 287 - CIM Elements for Disk Sparing (Continued)

Element Name	Requirement	Description
13.8.15 CIM_StoragePool	Mandatory	Elements to Primordial and Concrete Pools.
13.8.16 CIM_StorageRedundancySet	Mandatory	Represents the group of spare StorageExtents and StorageExtents that these spares will substitute for case of failure.
13.8.17 CIM_StorageVolume	Conditional	Conditional requirement: Referenced from Array - StorageVolume is mandatory or Referenced from Storage Virtualizer - StorageVolume is mandatory. Commonly known as a LUN but without the semantics of mapping to a host (which is covered by Masking and Mapping).
13.8.18 SNIA_FailoverStorageExtentsCollection	Optional	The collection of StorageExtents that have failed.
13.8.19 SNIA_SpareConfigurationCapabilities	Optional	Instances of this class define the behavior supported by this sparing implementation.
13.8.20 SNIA_SpareConfigurationService	Optional	This service manages sparing and validates the data and the parity for the StorageExtent. Not instantiating the service means that the service methods are supported.

13.8.2 CIM_AssociatedComponentExtent (Spare to Storage Pool)

The referenced spare StorageExtent represents capacity has not been allocated, is allocated in part, or is allocated in its entirety.

Requirement: Implementation of the Extent Composition profile.

Table 288 describes class CIM_AssociatedComponentExtent (Spare to Storage Pool).

Table 288 - SMI Referenced Properties/Methods for CIM_AssociatedComponentExtent (Spare to Storage Pool)

Properties	Flags	Requirement	Description and Notes
GroupComponent		Mandatory	The StoragePool.
PartComponent		Mandatory	The spare storage extent that is a component of the storage pool.

13.8.3 CIM_ConcreteDependency (Extent to LogicalDisk)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Referenced from Volume Management - LogicalDisk is mandatory.

Table 289 describes class CIM_ConcreteDependency (Extent to LogicalDisk).

Table 289 - SMI Referenced Properties/Methods for CIM_ConcreteDependency (Extent to LogicalDisk)

Properties	Flags	Requirement	Description and Notes
Antecedent		Mandatory	An underlying Storage Extent.
Dependent		Mandatory	A Logical Disk.

13.8.4 CIM_ConcreteDependency (Extent to Pool)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 290 describes class CIM_ConcreteDependency (Extent to Pool).

Table 290 - SMI Referenced Properties/Methods for CIM_ConcreteDependency (Extent to Pool)

Properties	Flags	Requirement	Description and Notes
Antecedent		Mandatory	
Dependent		Mandatory	

13.8.5 CIM_ConcreteDependency (Extent to StorageVolume)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Referenced from Array - StorageVolume is mandatory or Referenced from Storage Virtualizer - StorageVolume is mandatory.

Table 291 describes class CIM_ConcreteDependency (Extent to StorageVolume).

Table 291 - SMI Referenced Properties/Methods for CIM_ConcreteDependency (Extent to StorageVolume)

Properties	Flags	Requirement	Description and Notes
Antecedent		Mandatory	An underlying primordial Extent.
Dependent		Mandatory	A StorageVolume.

13.8.6 CIM_ElementCapabilities

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 292 describes class CIM_ElementCapabilities.

Table 292 - SMI Referenced Properties/Methods for CIM_ElementCapabilities

Properties	Flags	Requirement	Description and Notes
ManagedElement		Mandatory	The hosting System.
Capabilities		Mandatory	The support spare configuration capabilities.

13.8.7 CIM_HostedCollection (ComputerSystem to FailoverStorageExtentsCollection)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 293 describes class CIM_HostedCollection (ComputerSystem to FailoverStorageExtentsCollection).

**Table 293 - SMI Referenced Properties/Methods for CIM_HostedCollection
(ComputerSystem to FailoverStorageExtentsCollection)**

Properties	Flags	Requirement	Description and Notes
Antecedent		Mandatory	The hosting System.
Dependent		Mandatory	Indicates which FailoverStorageExtentsCollection are part of Disk Sparing implementation.

13.8.8 CIM_HostedCollection (ComputerSystem to RedundancySet)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 294 describes class CIM_HostedCollection (ComputerSystem to RedundancySet).

**Table 294 - SMI Referenced Properties/Methods for CIM_HostedCollection
(ComputerSystem to RedundancySet)**

Properties	Flags	Requirement	Description and Notes
Antecedent		Mandatory	The hosting System.
Dependent		Mandatory	Indicate which StorageRedundancySets are part of Disk Sparing implementation.

13.8.9 CIM_HostedService (ComputerSystem to SpareConfigurationService)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 295 describes class CIM_HostedService (ComputerSystem to SpareConfigurationService).

**Table 295 - SMI Referenced Properties/Methods for CIM_HostedService
(ComputerSystem to SpareConfigurationService)**

Properties	Flags	Requirement	Description and Notes
Antecedent		Mandatory	The hosting System.
Dependent		Mandatory	The support spare configuration service.

13.8.10 CIM_IsSpare

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 296 describes class CIM_IsSpare.

Table 296 - SMI Referenced Properties/Methods for CIM_IsSpare

Properties	Flags	Requirement	Description and Notes
SpareStatus		Mandatory	
FailoverSupported		Mandatory	
Antecedent		Mandatory	A Spare Storage Extent.
Dependent		Mandatory	

13.8.11 CIM_LogicalDisk

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Referenced from Volume Management - LogicalDisk is mandatory.

Table 297 describes class CIM_LogicalDisk.

Table 297 - SMI Referenced Properties/Methods for CIM_LogicalDisk

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	Opaque identifier.
ElementName		Optional	User friendly name.
Name		Mandatory	OS Device Name.
NameFormat		Mandatory	Format for name.
ExtentStatus		Mandatory	
OperationalStatus		Mandatory	
BlockSize		Mandatory	
NumberOfBlocks		Mandatory	The number of blocks that make of this LogicalDisk.
IsBasedOnUnderlyingRedundancy		Mandatory	
NoSinglePointOfFailure		Mandatory	
DataRedundancy		Mandatory	
PackageRedundancy		Mandatory	
DeltaReservation		Mandatory	
Primordial		Mandatory	Shall be false.

13.8.12 CIM_MemberOfCollection

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 298 describes class CIM_MemberOfCollection.

Table 298 - SMI Referenced Properties/Methods for CIM_MemberOfCollection

Properties	Flags	Requirement	Description and Notes
Member		Mandatory	
Collection		Mandatory	

13.8.13 CIM_Spared

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 299 describes class CIM_Spared.

Table 299 - SMI Referenced Properties/Methods for CIM_Spared

Properties	Flags	Requirement	Description and Notes
Antecedent		Mandatory	A reference to the StorageExtent that has replaced another StorageExtent.
Dependent		Mandatory	The StorageExtent that has failed and is being replaced.

13.8.14 CIM_StorageExtent (Spare)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 300 describes class CIM_StorageExtent (Spare).

Table 300 - SMI Referenced Properties/Methods for CIM_StorageExtent (Spare)

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	
HealthState		Mandatory	Reports the state of the StorageExtents underlying component.
OperationalStatus		Mandatory	Reports the operational status of the StorageExtent.
Primordial		Mandatory	A boolean that identifies whether the spare is primordial or concrete.

13.8.15 CIM_StoragePool

Requirement: Mandatory.

Table 301 describes class CIM_StoragePool.

Table 301 - SMI Referenced Properties/Methods for CIM_StoragePool

Properties	Flags	Requirement	Description and Notes
InstanceID		Mandatory	
ElementName		Mandatory	
PoolID		Mandatory	A unique name in the context of this system that identifies this Pool.

13.8.16 CIM_StorageRedundancySet

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 302 describes class CIM_StorageRedundancySet.

Table 302 - SMI Referenced Properties/Methods for CIM_StorageRedundancySet

Properties	Flags	Requirement	Description and Notes
InstanceID		Mandatory	
RedundancyStatus		Mandatory	The redundancy status shall be either 'Unknown' 0, 'Redundant' 2, or 'Redundancy Lost' 3. The implementation should report 2 or 3 most of the time, although it may report 0 sometimes. It should report 2 when there is at least one spare per the StorageRedundancySet. It should report 3 when there are no more spares (via IsSpare association) per the StorageRedundancySet.
TypeOfSet		Mandatory	'Limited Sparing', 5, is the type of sparing supported in the subprofile.
MinNumberNeeded		Mandatory	
MaxNumberSupported		Mandatory	
Failover()		Optional	For block servers that do not do automatically fail over failed components, this method is used to cause the fail over to occur. More commonly, block server implementations automatically maintain the availability of their capacity. In this case, the method would only be used to cause fail back to occur, if that also does not occur automatically.

13.8.17 CIM_StorageVolume

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Referenced from Array - StorageVolume is mandatory or Referenced from Storage Virtualizer - StorageVolume is mandatory.

Table 303 describes class CIM_StorageVolume.

Table 303 - SMI Referenced Properties/Methods for CIM_StorageVolume

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	Opaque identifier.
ElementName		Optional	User friendly name.
Name		Mandatory	VPD 83 identifier for this volume (ideally a LUN WWN).
NameFormat		Mandatory	Format for name.
ExtentStatus		Mandatory	
OperationalStatus		Mandatory	
Primordial		Mandatory	Shall be false.

13.8.18 SNIA_FailoverStorageExtentsCollection

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 304 describes class SNIA_FailoverStorageExtentsCollection.

Table 304 - SMI Referenced Properties/Methods for SNIA_FailoverStorageExtentsCollection

Properties	Flags	Requirement	Description and Notes
InstanceID		Mandatory	
ElementName		Mandatory	User friendly name.

13.8.19 SNIA_SpareConfigurationCapabilities

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 305 describes class SNIA_SpareConfigurationCapabilities.

Table 305 - SMI Referenced Properties/Methods for SNIA_SpareConfigurationCapabilities

Properties	Flags	Requirement	Description and Notes
InstanceID		Mandatory	
ElementName		Mandatory	User friendly name.
SupportedAsynchronousActions	N	Mandatory	Enumeration indicating what operations will be executed as asynchronous jobs. If an operation is included in both this and SupportedSynchronousActions then the underlying implementation is indicating that it may or may not create.

Table 305 - SMI Referenced Properties/Methods for SNIA_SpareConfigurationCapabilities (Con-

Properties	Flags	Requirement	Description and Notes
SupportedSynchronousActions	N	Mandatory	Enumeration indicating what operations will be executed without the creation of a job. If an operation is included in both this and SupportedAsynchronousActions then the underlying instrumentation is indicating that it may or may not create a job.
SystemConfiguredSpares		Mandatory	Set to true if this storage system automatically configures spares. If set to false, the client shall use the extrinsic methods AssignSpares and UnassignSpares.
AutomaticFailOver		Mandatory	Set to true if this storage system automatically fails over. If set to false, the client shall use the FailOver extrinsic method, although that method may not be supported.
MaximumSpareStorageExtents		Mandatory	States the maximum number of StorageExtents that can be configured as spares for the entire block server. A 0 means that all primordial StorageExtents can be configured as spares.

13.8.20SNIA_SpareConfigurationService

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 306 describes class SNIA_SpareConfigurationService.

Table 306 - SMI Referenced Properties/Methods for SNIA_SpareConfigurationService

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
Name		Mandatory	Opaque identifier.
AssignSpares()		Mandatory	
UnassignSpares()		Mandatory	
GetAvailableSpareExtents()		Mandatory	
RebuildStorageExtent()		Optional	
CheckParityConsistency()		Optional	
RepairParity()		Optional	
CheckStorageElement()		Optional	

IMPLEMENTED

EXPERIMENTAL

14 Erasure Profile

14.1 Description

14.1.1 General

The Erasure Profile describes how data on a storage element (StorageVolume, LogicalDisk, or primordial StorageExtent) may be erased. As data is replicated, migrated and archived throughout its lifecycle, there is a need to ensure that residual and superseded copies or versions of the data that remain on storage media are destroyed in line with business policies for privacy, confidentiality and security.

Erasure will be required whenever it is deemed that the data on a storage element is sufficiently sensitive or of competitive value that the media cannot be reused, redeployed or made redundant without ensuring that the data is destroyed.

As part of the data lifecycle, data will potentially be replicated and migrated several times throughout their life before final destruction, as a result of media and technology change or management policies.

Common situations would include:

- Migration to secondary or tertiary archive storage followed by deletion of the source data.
- Movement of data from a failing device to a spare.
- Migration and cut-over to new target media, retaining the source media for a "fall back" for some period then reuse (or resale) of the source media.

14.1.2 Existing Erasure standards

There are numerous erasure standards in the industry. These techniques generally involve writing a bit pattern to the storage media and in most cases require multiple passes of overwriting of these bit patterns. The following is an incomplete list of erasure techniques to illustrate the variety that exists today.

- HMG Infosec Standard 5, The Baseline Standard.
- HMG Infosec Standard 5, The Enhanced Standard.
- Peter Gutmann's algorithm.
- U.S.Department of Defense Sanitizing (DOD 5220.22-M)
- Bruce Schneier's algorithm.
- Navy Staff Office Publication (NAVSO P-5239-26) for RLL.
- The National Computer Security Center (NCSC-TG-025).
- Air Force System Security Instruction 5020.
- US Army AR380-19.
- German Standard VSIT
- OPNAVINST 5239.1A.

Because there is such a wide variety of techniques, this subprofile does not dictate which technique shall be used. The instrumentation shall tell the client which methods are supported. Since erasure of data on a volume may be a lengthy process and will most likely be a

background task, the volume may provide the status of the erasure and may provide notification via an Indication of the erasure completion.

To support this profile, instrumentation shall provide a list of supported erasure methods in the ErasureCapabilities.SupportedErasureMethods property. If the instrumentation supports erasing a volume upon return to a storage pool, then the ErasureCapabilities.CanEraseOnReturnToStoragePool property shall be set to true. If the instrumentation does not support this capability, then the value shall be false (the default value). The ErasureCapabilities shall be associated to the ErasureService via the ElementCapabilities association.

If CanEraseOnReturnToStoragePool is true, then the ErasureCapabilities.DefaultErasureMethod shall be used to erase StorageVolume or LogicalDisk elements, unless the ErasureSetting.ErasureMethod is non-NULL. The instrumentation may provide a default value for this property. A client may be able to change the ErasureCapabilities.DefaultErasureMethod and ErasureSetting.ErasureMethod:

The erasure of StorageExtents is restricted to primordial extents only and shall be accomplished by calling ErasureService.Erase explicitly.

The CanEraseOnReturnToStoragePool shall only be used for StorageVolumes and LogicalDisks.

To erase the volume explicitly, the user shall call the ErasureService.Erase method, passing in the volume to erase and the erasure method to use. The erasure method shall be one of the erasure methods the instrumentation supports. A NULL may be passed in as the ErasureMethod, in which case, the instrumentation shall use the DefaultErasureMethod from the capabilities as the erasure method. To erase a volume implicitly, it is required that the CanEraseOnReturnToStoragePool shall be true and that the ErasureSetting associated to the volume has the EraseOnReturnToPool value set to true. If these conditions are met, then when the user calls the ReturnToStoragePool method, the volume shall be erased before being returned to the pool.

If a ConcreteJob has been started as a result of the erasure (either from calling Erase or ReturnToStoragePool), then the ConcreteJob shall have an AffectedJobElement association to the StorageVolume being erased.

Figure 66 shows the new properties and method introduced by this subprofile. While a StorageVolume is shown, the same shall apply to LogicalDisk.

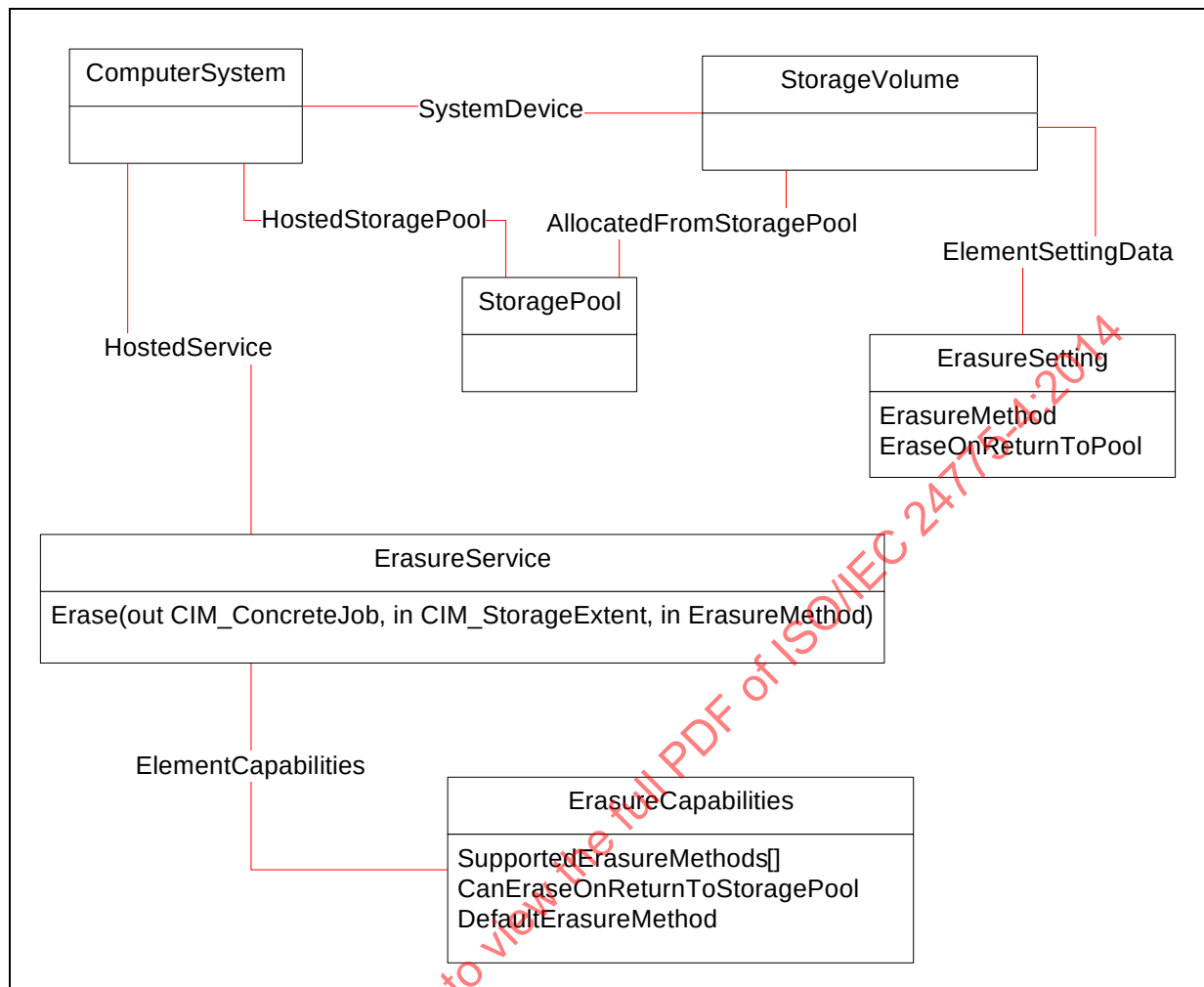


Figure 66 - Model Elements

14.2 Health and Fault Management Considerations

Not defined in this International Standard.

14.3 Cascading Considerations

Not applicable.

14.4 Supported Profiles, Subprofiles, and Packages

Not defined in this International Standard.

14.5 Methods of the Profile

The Erase method (see table 307) in the ErasureService shall erase the contents of the volume using the specified erasure method. The erasure methods that the instrumentation supports shall be found in the ErasureCapabilities.SupportedErasureMethods property.

Table 307 - Erase Method

Method: Erase	
Return Values:	
Value	Description
0: Job completed	Job completed with no error
1: Not supported	Method not supported

2: Unspecified Error			
3: Timeout			
4: Failed	Refer to instance of CIM_Error		
5: Invalid parameter	Refer to instance of CIM_Error		
6: In Use			
7..4095	DMTF Reserved		
4096: Job started	REF returned to started ConcreteJob		
Errors:			
(status):registry:MessageID	ErrorName:MessageArguments		
Parameters:			
Qualifiers	Name	Type	Description/Values
OUT	Job	CIM_ConcreteJob REF	Returned if job started.
IN, REQ	Extent	CIM_StorageExtent REF	Extent (volume) to erase
IN, REQ	Type	uint16	Type of extent (StorageVolume, LogicalDisk, or primordial StorageExtent)
IN, REQ	EraseMethod	uint32	Erase method to use

14.6 Client Considerations and Recipes

14.6.1 General

These cases can be generalized into the explicit case of "Volume Erasure" and the implicit case of "Volume Deletion".

14.6.2 Recipe 1: Volume Erasure

This is the case where it is determined that the contents of a storage volume shall be erased. This requires the client to call the ErasureService method Erase() to specify the StorageVolume and the ErasureMethod.

```
// DESCRIPTION:
//
// Erase a volume
//
// PRE-EXISTING CONDITIONS AND ASSUMPTIONS
//
// 1. The ErasureService has been found and the object path
//    value is stored in $ErasureService->
// 2. The ErasureCapabilities associated to the
//    ErasureService has been found and the instance stored
//    in $ControllerCapabilities
// 3. The StorageVolume to use has been identified and the object path
//    values are stored in $Volume->
// 4. The erasure method to use has been determined and it's value
//    stored in #EraseMethod

// Determine if there is a job created by method
```



```

// and wait for the job to complete
// Input:
//   #ReturnCode : The return code of the method
//   $ConcreteJob-> :The output parameter that may have a ConcreteJob
//                   REF.
// This method will return control if the recipe was not exited because
// of error
sub void WaitForJob(#ReturnCode, $ConcreteJob->) {
  if (4096 == #ReturnCode) {
    if ($ConcreteJob-> != null) {
      /*Wait until the completion of the job using $ConcreteJob-> as
      a filter Verify that the OperationalStatus contains 2 ("OK"),
      or 17 ("Completed") */
      $JobInstance = GetInstance($ConcreteJob->,
        false, false, false, null)
      if ($JobInstance.JobStatus != 7) { // 7 - Completed
        <ERROR! Job failed! >
      }
    } else {
      <ERROR! Missing Job reference>
    }
  }
}

// Step 1. Erase the volume

if ( (#ErasureMethod != NULL) &&
  (contains(#ErasureMethod,
    $ControllerCapabilities.SupportedErasureMethods[]) == false) ) {
  <ERROR! Invalid Erasure method>
}

%InputArguments["Extent"]      = { $Volume-> }
%InputArguments["Type"]        = 1 // StorageVolume
%InputArguments["ErasureMethod"] = #ErasureMethod

#ReturnCode = InvokeMethod($ErasureService->,
  "Erase",
  %InputArguments,
  %OutputArguments)

// 0 is "Success" and 4096 is "Method Parameters Checked - Job Started"
if (#ReturnCode != 0 || #ReturnCode != 4096) {
  <ERROR! Method failure>
}

$Job-> = %OutputArguments["Job"]
if ($Job-> != null) {
  // Wait until job is finished
  &WaitForJob(#ReturnCode, $Job->)
}

```

14.6.3 Recipe 2: Volume Deletion

This case is where a volume is being returned to the storage pool, and it needs to be erased. The client needs to check the CanEraseOnReturnToStoragePool property to see if this is possible. If it is, then the client looks for an ErasureSetting associated to the volume, creating one if necessary. The client sets the ErasureSetting.EraseMethod and ErasureSetting.EraseOnReturnToStoragePool for the setting associated to the volume, then calls ReturnToStoragePool.

```
// DESCRIPTION:
//
// Erase a volume as a byproduct of being deleted
//
// PRE-EXISTING CONDITIONS AND ASSUMPTIONS
//
// 1. The ErasureService has been found and the object path
//    value is stored in $ErasureService->
// 2. The ErasureCapabilities associated to the
//    ErasureService has been found and the instance stored
//    in $ControllerCapabilities
// 3. The StorageConfigurationService has been found and the object path
//    value is stored in $StorageConfigService->
// 4. The StorageVolume to use has been identified and the object path
//    values are stored in $Volume->
// 5. The erasure method to use has been determined and it's value
//    stored in #ErasureMethod
//
// Determine if there is a job created by method
// and wait for the job to complete
// Input:
// #ReturnCode : The return code of the method
// $ConcreteJob-> :The output parameter that may have a ConcreteJob
//                  REF.
// This method will return control if the recipe was not exited because
//                  of error
sub void WaitForJob(#ReturnCode, $ConcreteJob->) {
    if (4096 == #ReturnCode) {
        if ($ConcreteJob-> != null) {
            /*Wait until the completion of the job using $ConcreteJob-> as
            a filter Verify that the OperationalStatus contains 2 ("OK"),
            or 17 ("Completed") */
            $JobInstance = GetInstance($ConcreteJob->,
                false, false, false, null)
            if ($JobInstance.JobStatus != 7) { // 7 - Completed
                <ERROR! Job failed! >
            }
        }
    } else {
        <ERROR! Missing Job reference>
    }
}
```

```

}

// Step 1. Check capabilities

if ( $ControllerCapabilities.CanEraseOnReturnToStoragePool == false) {
    <ERROR! Implicit erasure not supported. Use Erase() method >
}

// Step 2. Find/create setting
$Setting[] = Associators($Volume->,
    "CIM_ElementSettingData",
    "SNIA_EraseSetting",
    null, null,
    false, false, null)
if ($Setting[].length == 0) {
    // Create setting
    $TheSetting = newInstance("SNIA_EraseSetting")
    $TheSetting.InstanceID = "SNIA:0001" // create unique ID
    $TheSetting.EraseMethod = #EraseMethod
    $TheSetting.EraseOnReturnToStoragePool = true
    $instance-> = CreateInstance($TheSetting)
}
else {
    $Setting[0].EraseMethod = #EraseMethod
    $Setting[0].EraseOnReturnToStoragePool = true
    ModifyInstance($Setting[0])
}

// Step 3 Delete the volume
%InArguments["TheElement"] = $Volume->

#ReturnCode = InvokeMethod($StorageService->,
    "ReturnToStoragePool",
    %InArguments,
    %OutArguments)

// 0 is "Success" and 4096 is "Method Parameters Checked - Job Started"
if (#ReturnCode != 0 || #ReturnCode != 4096) {
    <ERROR! Method failure>
}

$Job-> = %OutputArguments["Job"]
if ($Job-> != null) {
    // Wait until job is finished
    &WaitForJob(#ReturnCode, $Job->)
}

```

14.7 Registered Name and Version

Erasure version 1.2.0 (Component Profile).

14.8 CIM Elements

14.8.1 General

Table 308 describes the CIM elements for Erasure.

Table 308 - CIM Elements for Erasure

Element Name	Requirement	Description
14.8.2 CIM_AllocatedFromStoragePool	Mandatory	AllocationFromStoragePool as defined in the Array Profile.
14.8.3 CIM_LogicalDisk	Conditional	Conditional requirement: Conditional
14.8.4 CIM_StoragePool	Mandatory	
14.8.5 CIM_StorageVolume	Conditional	Conditional requirement: Conditional
14.8.6 SNIA_ErasureCapabilities	Mandatory	
14.8.7 SNIA_ErasureService	Mandatory	
14.8.8 SNIA_ErasureSetting	Mandatory	

14.8.2 CIM_AllocatedFromStoragePool

Created By: External

Modified By: External

Deleted By: External

Requirement: Mandatory.

Table 309 describes class CIM_AllocatedFromStoragePool.

Table 309 - SMI Referenced Properties/Methods for CIM_AllocatedFromStoragePool

Properties	Flags	Requirement	Description and Notes
SpaceConsumed		Mandatory	
Antecedent		Mandatory	
Dependent		Mandatory	

14.8.3 CIM_LogicalDisk

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: null.

Table 310 describes class CIM_LogicalDisk.

Table 310 - SMI Referenced Properties/Methods for CIM_LogicalDisk

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

14.8.4 CIM_StoragePool

Created By: External

Modified By: External

Deleted By: External

Requirement: Mandatory.

Table 311 describes class CIM_StoragePool.

Table 311 - SMI Referenced Properties/Methods for CIM_StoragePool

Properties	Flags	Requirement	Description and Notes
Primordial		Mandatory	
TotalManagedSpace		Mandatory	
RemainingManagedSpace		Mandatory	

14.8.5 CIM_StorageVolume

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: null.

Table 312 describes class CIM_StorageVolume.

Table 312 - SMI Referenced Properties/Methods for CIM_StorageVolume

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

14.8.6 SNIA_EraseCapabilities

Created By: Static

Requirement: Mandatory.

Table 313 describes class SNIA_EraseCapabilities.

Table 313 - SMI Referenced Properties/Methods for SNIA_EraseCapabilities

Properties	Flags	Requirement	Description and Notes
ElementName		Mandatory	User friendly name for this instance of Capabilities.
InstanceID		Mandatory	Unique identifier for the instance.
EraseMethods		Mandatory	Indicates erasure methods supported.

Table 313 - SMI Referenced Properties/Methods for SNIA_EraseCapabilities

Properties	Flags	Requirement	Description and Notes
DefaultEraseMethod		Mandatory	Erase method to use if none specified in the volume's setting.
CanEraseOnReturnToStoragePool		Mandatory	Indicates that the volume can be erased when deleted.
ElementTypesSupported		Mandatory	Supported element types for the Erase method. Valid values are StorageVolume, LogicalDisk, and StorageExtent.

14.8.7 SNIA_EraseService

Created By: Static

Requirement: Mandatory

Table 314 describes class SNIA_EraseService.

Table 314 - SMI Referenced Properties/Methods for SNIA_EraseService

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	The scoping System CreationClassName.
SystemName		Mandatory	The scoping System Name.
CreationClassName		Mandatory	The name of the concrete subclass.
Name		Mandatory	Unique identifier for the Service.
Erase()		Mandatory	This service contains the Erase method used to erase storage elements.

14.8.8 SNIA_EraseSetting

Created By: External

Modified By: External

Deleted By: External

Requirement: Mandatory.

Table 315 describes class SNIA_EraseSetting.

Table 315 - SMI Referenced Properties/Methods for SNIA_EraseSetting

Properties	Flags	Requirement	Description and Notes
EraseMethod		Mandatory	Erase method to use. Shall be one of the erasure methods supported by the instrumentation.
EraseOnReturnToPool		Mandatory	Indicates if this volume should be erased when deleted. Default is false.

EXPERIMENTAL

STABLE**15 Extent Composition Subprofile****15.1 Description****15.1.1 General**

The Extent Composition Subprofile allows an implementation that supports the Block Services package to optionally provide an abstraction of how it virtualizes exposable block storage elements from the underlying Primordial storage pool. The abstraction is presented to the client as a representative hierarchy of extents. These extents are instances of CompositeExtents and StorageExtents linked by a combination of CompositeExtentBasedOn and BasedOn associations. The foundation of the hierarchy is a set of Primordial extents.

This subprofile is used optionally with the Array, Virtualization, Self-Contained NAS, NAS Head, and Volume Management profiles.

A Primordial storage extent can represent a Disk Drive in the Array or Self-contained NAS, a downstream virtualized Volume used by the Virtualizer or NAS Head Profiles, or a OS Logical Disk in the Volume Management Profile.

An exposable block storage element as used in this subprofile is defined as a Storage Volume or a Logical Disk.

In the presented hierarchy each extent (the dependent) is formed from those that it "precede" it (the antecedents) by a process of either decomposition or composition.

15.1.2 Decomposition

Decomposition is used to allocate space from an antecedent extent, in order to form a new dependent extent. This allocation may be partial or complete consumption. Complete consumption is the degenerate case in which all space in the antecedent extent is used. In this case the decomposed dependent extent may be either modeled even though it is one to one with the antecedent extent or omitted and the antecedent extent used in its stead.

15.1.3 Composition

Composition is used to form an a dependent extent from antecedent extents for the purpose of either concatenating the antecedent blocks to achieve a size goal, or to achieve a Quality Of Service goal such as mirroring the antecedent extents for redundancy, striping the antecedent extents for performance, or striping the antecedent extents with the addition of parity to achieve redundancy.

These extent "productions" can be assembled in a multi-layer hierarchy.

15.1.4 Model Element Summary

This subprofile uses the following CIM Classes:

LogicalDisk and StorageVolume - These are used to model the exposable block storage element. These are as defined in the Block Services Package. The StorageVolume may also be a Constituent Volume as defined by the Pools From Volumes Profile.

StorageExtent (Intermediate or Pool Component) - Used to represent the decomposition (partial allocation) of an Antecedent extent.

StorageExtent (Remaining) - Used to represent the unused portion of an antecedent StorageExtent (Pool Component).

CompositeExtent (Composite Intermediate or Composite Pool Component) - Used to represent the composition of several antecedent extents into a virtualized set of blocks with desired size and Quality-Of-Service.

BasedOn - Used to associate a Dependent and Antecedent extent in the subprofile hierarchy for both composition and decomposition. It is also used in one special case as a one-to-one (neither composing or decomposing), always associating the StorageVolume or LogicalDisk to the antecedent CompositeExtent. This is because, as a sibling of StorageExtent and LogicalDisk, CompositeExtent cannot be exposed directly.

CompositeExtentBasedOn - A subclass of BasedOn that is used in a composition production when the Dependent is a CompositeExtent which is describing striping. It contains Stripe Depth information. Stripe Depth is the number of blocks written to an Antecedent extent before moving on to the next extent. Although this property is on the association class, its values shall be the same for each instance of the association with the same Dependent CompositeExtent.

DEPRECATED

ConcreteComponent - Used to associate extents (Pool Component and Remaining) to their parent StoragePool (see 15.1.5.2).

DEPRECATED

AssociatedComponentExtent - Used to associate extents (Pool Component or Composite Pool Component) to their parent StoragePool (see 15.1.5.2).

StoragePool and AllocatedStoragePool are shown in instance diagrams for context but are part of the Block Service package Read Only sub-package.

Refer to 15.8 "CIM Elements" for detailed class descriptions.

15.1.5 Relation to other Packages and Subprofiles

15.1.5.1 Block Services StoragePool hierarchy.

The Block Services package defines the model for the hierarchy of pools from the exposable storage element to the Primordial Pool. The hierarchy defined in this subprofile parallels that pool hierarchy and is layered so that the virtualization can be presented within the pool level in which it actually takes place.

15.1.5.2 Component Extents

Component Extents of a pool are the most dependent extents in the pool. They are also the only extents that are directly *manageable* by the methods in the Block Services Package. They are also the only extents that figure into the reconciliation of managed space in the pool (see 15.1.5.3).

Although a given implementation may choose a low level (i.e., detailed) or high-level presentation of how it virtualizes a storage element from a pool, or how space in a pool is itself virtualized, the Pool Component extents that are part of an exposable block storage element's hierarchy shall be modeled along with their associations to the parent pool.

15.1.5.3 Block Services Extent Conservation

The Block Services package describes the concept of Extent Conservation, which describes the result of allocating storage from Pool Component extents using "Remain Space Extents". These extents are not modeled by the Extent Composition Subprofile, they are discoverable by the GetAvailableExtents method in Block Services.

15.1.5.4 Block Services Common RAID Levels

The Block Services Package describes a set of RAID Levels and in addition, properties on StorageSetting such as ExtentStripeLength and UserDataStripeDepth which allow creation of a subset of those RAID levels, using CreateOrModifyElementFromElements.

15.1.6 Remaining Extents

[illegible]

Figure 67 shows two Remaining StorageExtents. Building from the bottom, there is a Pool Component StorageExtent allocated from the Primordial StorageExtent. But this StorageExtent does not use all space on the primordial extent. So a Remaining StorageExtent is shown to represent the unallocated space on the primordial extent. The Remaining StorageExtent has a BaseOn association to the primordial extent to indicate that it is unallocated space from the primordial extent. The Remaining Extent also has an AssociatedRemainingExtent association to the same primordial StoragePool that the primordial StorageExtent has its AssociatedComponentExtent association.

The Pool Component extent above the primordial storage extent also has a StorageExtent allocated from it that is also a partial allocation. So, it too has a Remaining StorageExtent to represent the unallocated space on the Pool Component StorageExtent. This Remaining StorageExtent has a BasedOn association to the Pool Component StorageExtent and an AssociatedRemainingExtent association to the same Concrete StoragePool that the Pool Component StorageExtent has its AssociatedComponentExtent association.

For more information and detail on the use and application of Remaining StorageExtents, see 6.1.15 for extent conservation provisions.

15.1.7 Scenarios

15.1.7.1 General

The following example scenarios are common abstractions of the use-cases that were used when this subprofile was being defined. The scenarios are not intended to cover all possible variations of the use of Extent Composition.

15.1.7.2 Volume Composition

Figure 68: "Volume Composition from General QOS Pool" shows extent composition when a single RAID QOS/Level is applied directly to the construction of a StorageVolume. The Storage Volume or Logical Disk and the underlying CompositeExtent represent the same virtual extent and range of blocks. The initial BasedOn association between them is a one-to-one "dummy" association. The Storage Volume and Logical Disk classes do not have the necessary properties to describe the RAID information and the CompositeExtent which is a sibling class of StorageVolume and LogicalDisk, cannot be directly exposed. This Based on association does not represent composition or decomposition, but the main recipe (see 15.6.1) for this subprofile makes use of the decomposition function (i.e., complete consumption) to make this initial traversal.

IECNORM.COM : Click to view the full PDF of ISO/IEC 24775-4:2014

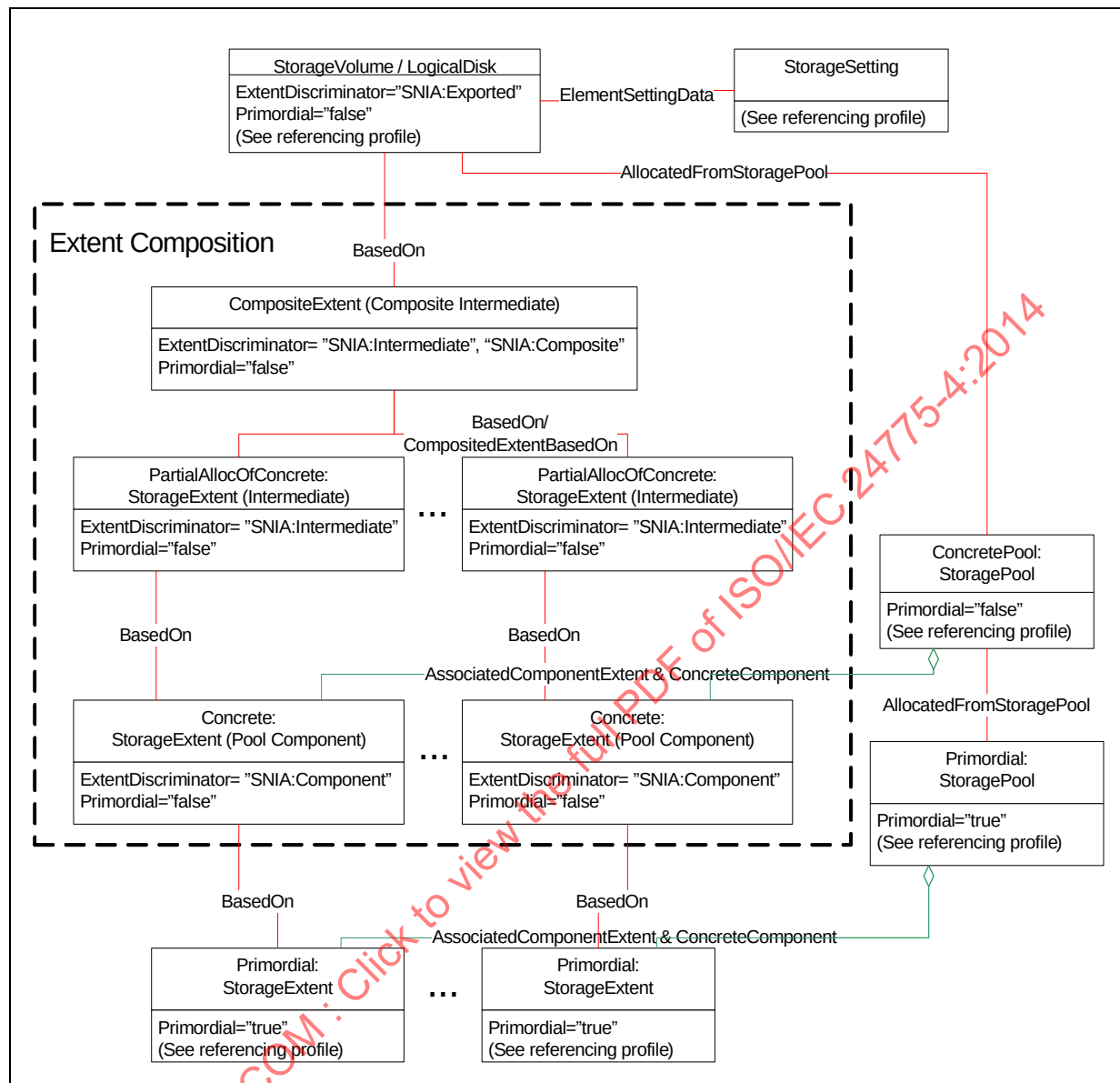


Figure 68 - Volume Composition from General QOS Pool

Figure 69: "Single QOS Pool Composition (RAID Groups)" shows a single composition (such as a RAID5 or RAID1). Not shown is the scenario where there may be two or more such back to back productions (such as a RAID10). Also not shown is the scenario where the two productions may be in different concrete pools in the hierarchy. A RAID10 Volume may be constructed as a RAID0 composition from a concrete pool that is itself a RAID1 pool (see 15.1.7.3).

In this scenario, note that the extents below the StorageVolume and the Component Extents are not part of the pool, but allocated from it.

In fact this StorageVolume and its companion CompositeExtent could be composed from member extents (labeled PartialAllocOfConcrete in the diagram) from different pools.

15.1.7.3 Pool Composition

Certain pools can be created or modified to contain one or more extents each with a single specific quality of service. These extents are known as Raid Groups. The bound space in each of these RAID Groups is represented by this subprofile as a single CompositeExtent at the top

of an extent sub-hierarchy in that pool. Volumes created from this type of Pool are partially allocated (decomposed) from the CompositeExtent playing the role of the RAIDGroup. Figure 69 shows the Single QOS Pool Composition (RAID Groups).

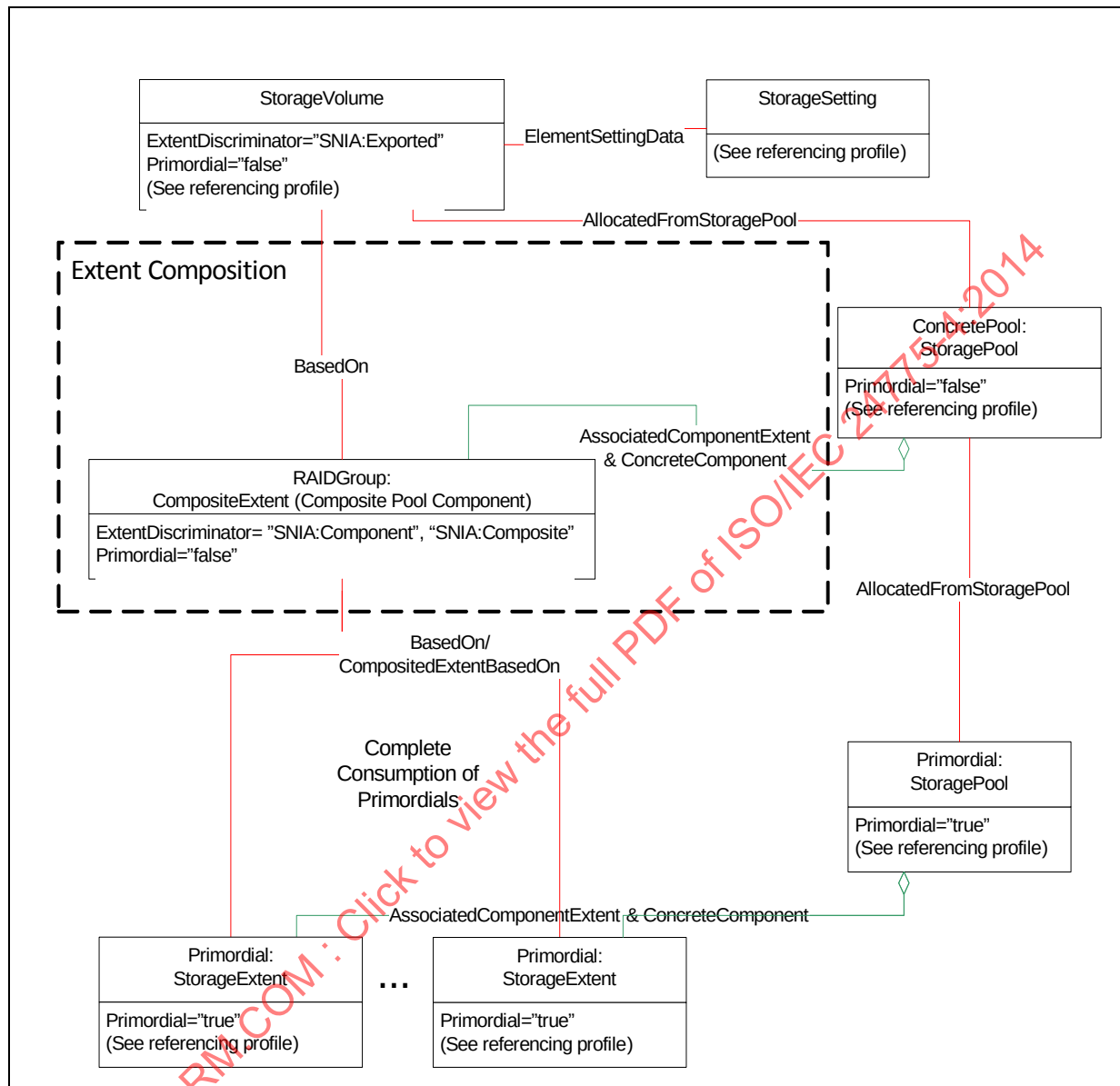


Figure 69 - Single QOS Pool Composition (RAID Groups)

Figure 70: "Single QOS Pool Composition - Two Concretes" extends this scenario by allocating a child concrete pool from the RAID Group instead of a Volume and then allocating the Volume from the child concrete. In this example the child pool contains a single component extent that has a single Quality of Service (that of the parent RAID Group concrete pool). The

Storage Volume or Logical Disk is allocated or decomposed directly from the pool component extent.

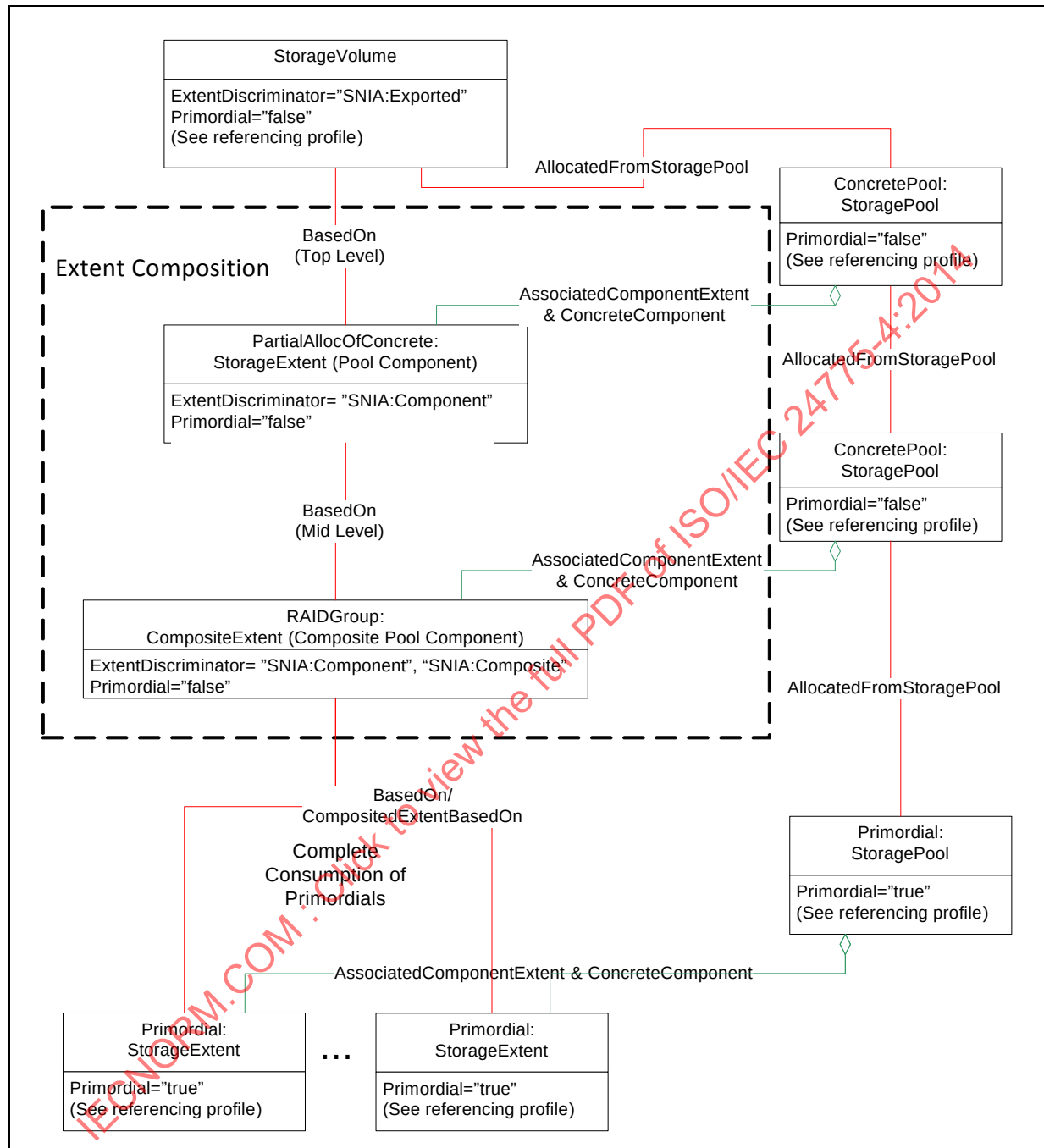


Figure 70 - Single QOS Pool Composition - Two Concretes

15.1.7.4 Example RAID Compositions from Block Services

15.1.7.4.1 General

Table 316 is an abridged version of the RAID Mapping table in Block Services. The table describes the RAID levels commonly used at the time this version of SMI-S was released. Table 316 lists the subset of those RAID Levels that can be modeled by using the Extent Composition Subprofile, and the properties used to distinguish them.

Following Table 316 are some example instance diagrams, showing the use of CompositeExtent, StorageExtent, BasedOn and CompositeExtentBasedOn to represent the

construction of many of the RAID levels. In these cases there will be at most, two levels of CompositeExtent and CompositeExtentBasedOn/BasedOn.

In complex compositions, such as RAID10, there is no intermediate decomposition modeled. Each extent Antecedent to the top level CompositeExtent is itself a CompositeExtent.

Table 316 - Supported Common RAID Levels

RAID Level	Package Redundancy	Data Redundancy	Extent Stripe Length	User Data Stripe Depth
JBOD	0	1	1	Null
0 (Striping)	0	1	2 - n	Vendor Dependent
1	1	2 - n	1	Null
10	1	2 - n	2 - n	Vendor Dependent
0+1	1	2 - n	2 - n	Vendor Dependent
3 or 4	1	1	3 - n	Vendor Dependent
4DP	2	1	4 - n	Vendor Dependent
5 (3/5)	1	1	3 - n	Vendor Dependent
6, 5DP	2	1	4 - n	Vendor Dependent
15	2	2 - n	3 - n	Vendor Dependent
50	1	1	3 - n	Vendor Dependent
51	2	2 - n	3 - n	Vendor Dependent

15.1.7.4.2 JBOD (Concatenation)

Figure 71: "Concatenation Composition" shows a partial instance diagram for a JBOD Volume or Pool, in which the Antecedent Extents are concatenated.

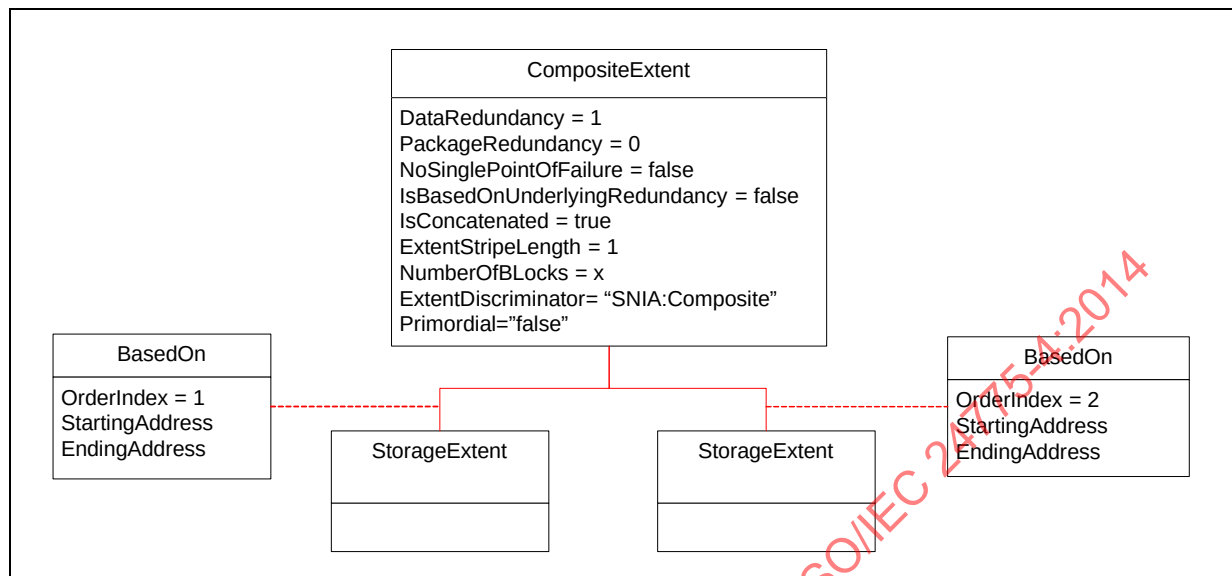


Figure 71 - Concatenation Composition

15.1.7.4.3 RAID0 (Striping)

Figure 72: "RAID0 Composition" shows a partial instance diagram for a RAID0 Volume or Pool.

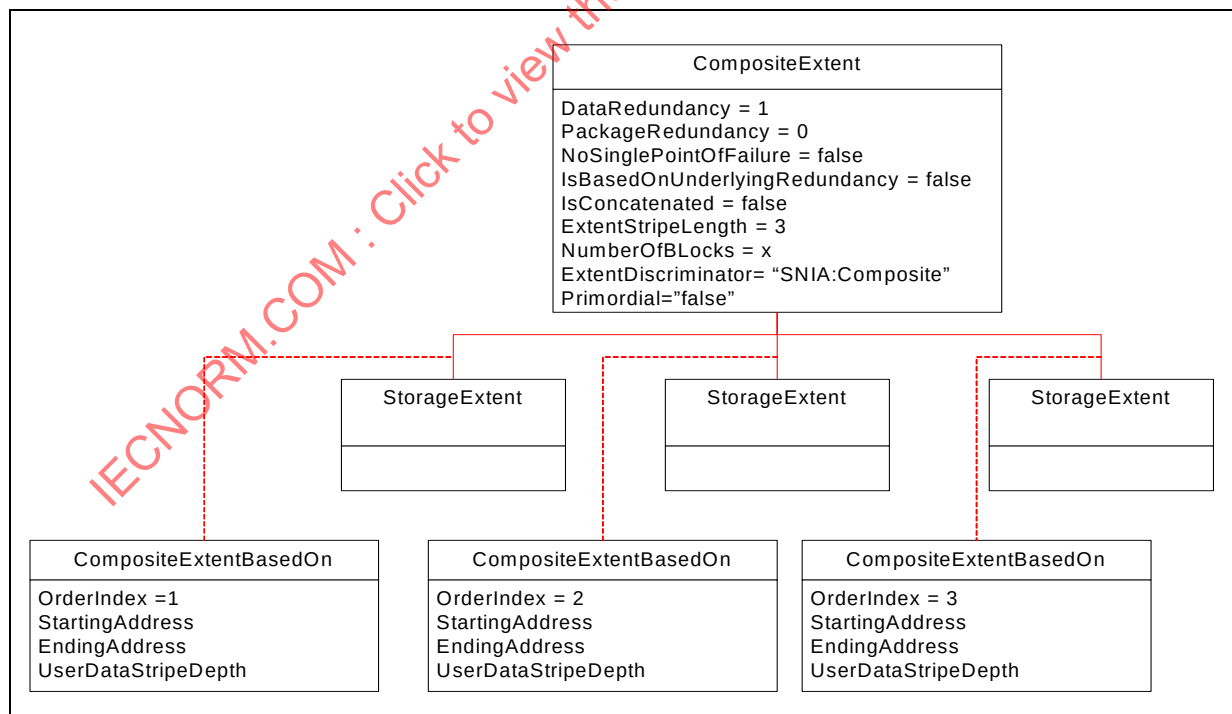


Figure 72 - RAID0 Composition

15.1.7.4.4 RAID1

Figure 73: "RAID1 Composition" shows a partial instance diagram for a RAID1 Volume or Pool.

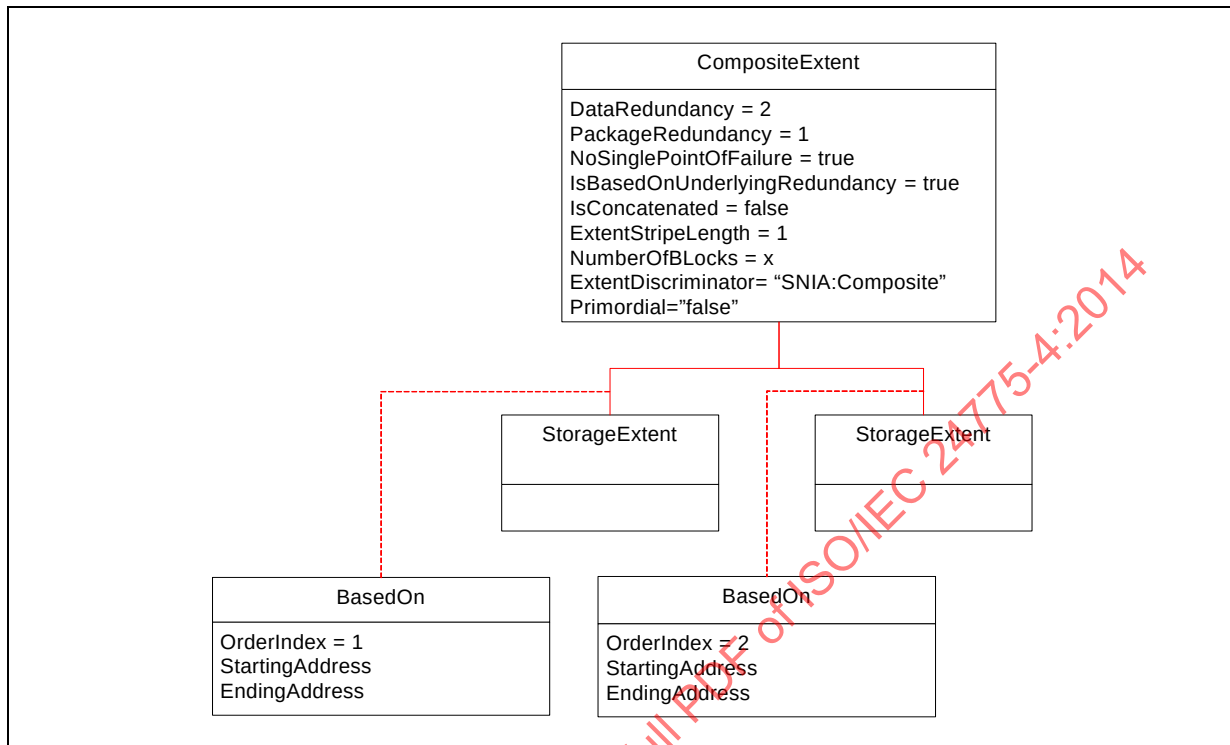


Figure 73 - RAID1 Composition

15.1.7.4.5 RAID10

Figure 74: "RAID10 Composition" shows a partial instance diagram for a RAID10 Volume or Pool. In this example the Data and Package Redundancy reflect the Quality of Service of the combined RAID Level, not just the top level composition which by itself is a non-redundant stripeset. That is, the top level is a RAID0, but the `DataRedundancy` value for the corresponding **CompositeExtent** is 2, reflecting two complete copies of the data.

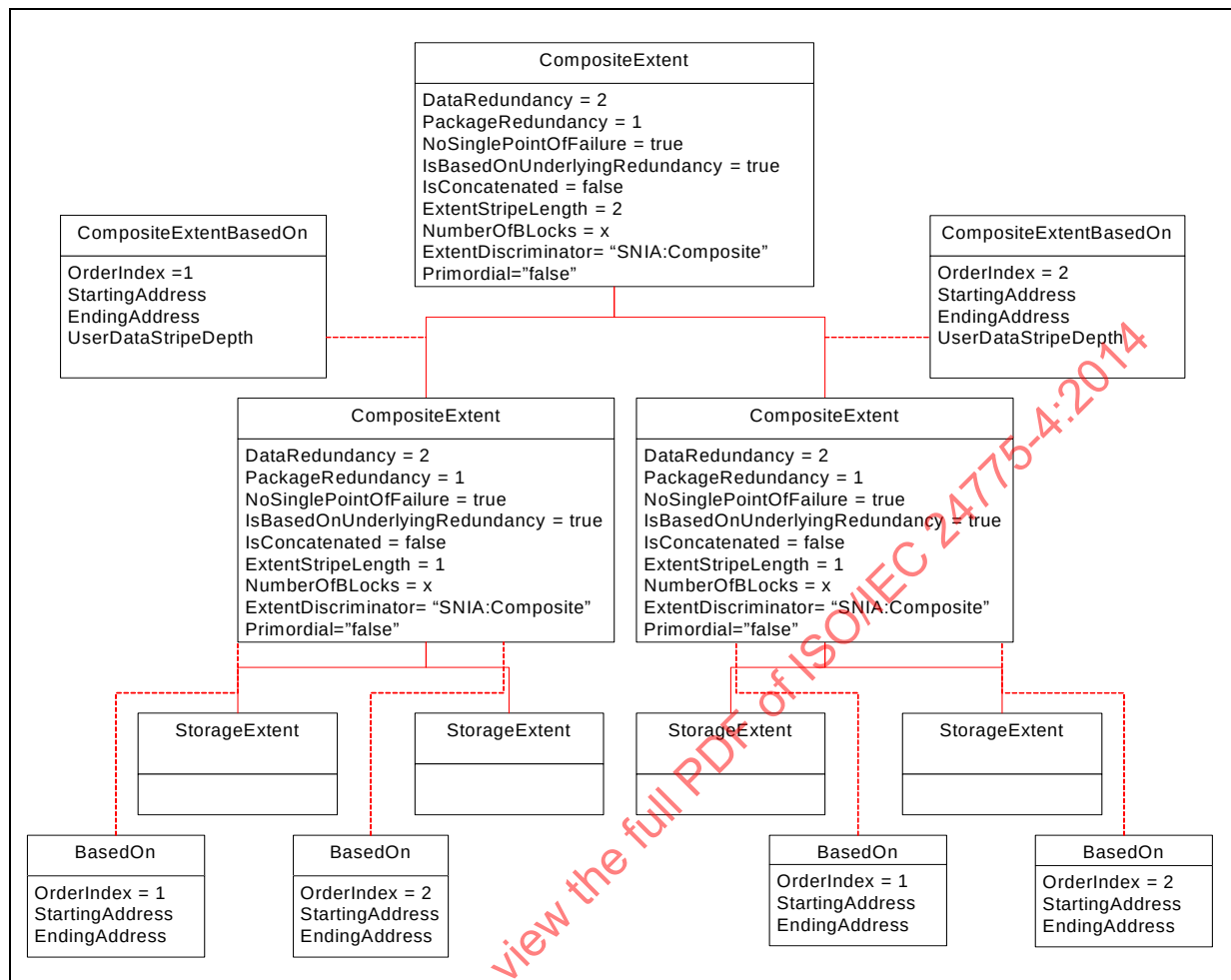


Figure 74 - RAID10 Composition

15.1.7.4.6 RAID0+1

Figure 75: "RAID0+1 Composition" shows a partial instance diagram for a RAID0+1 Volume or Pool.

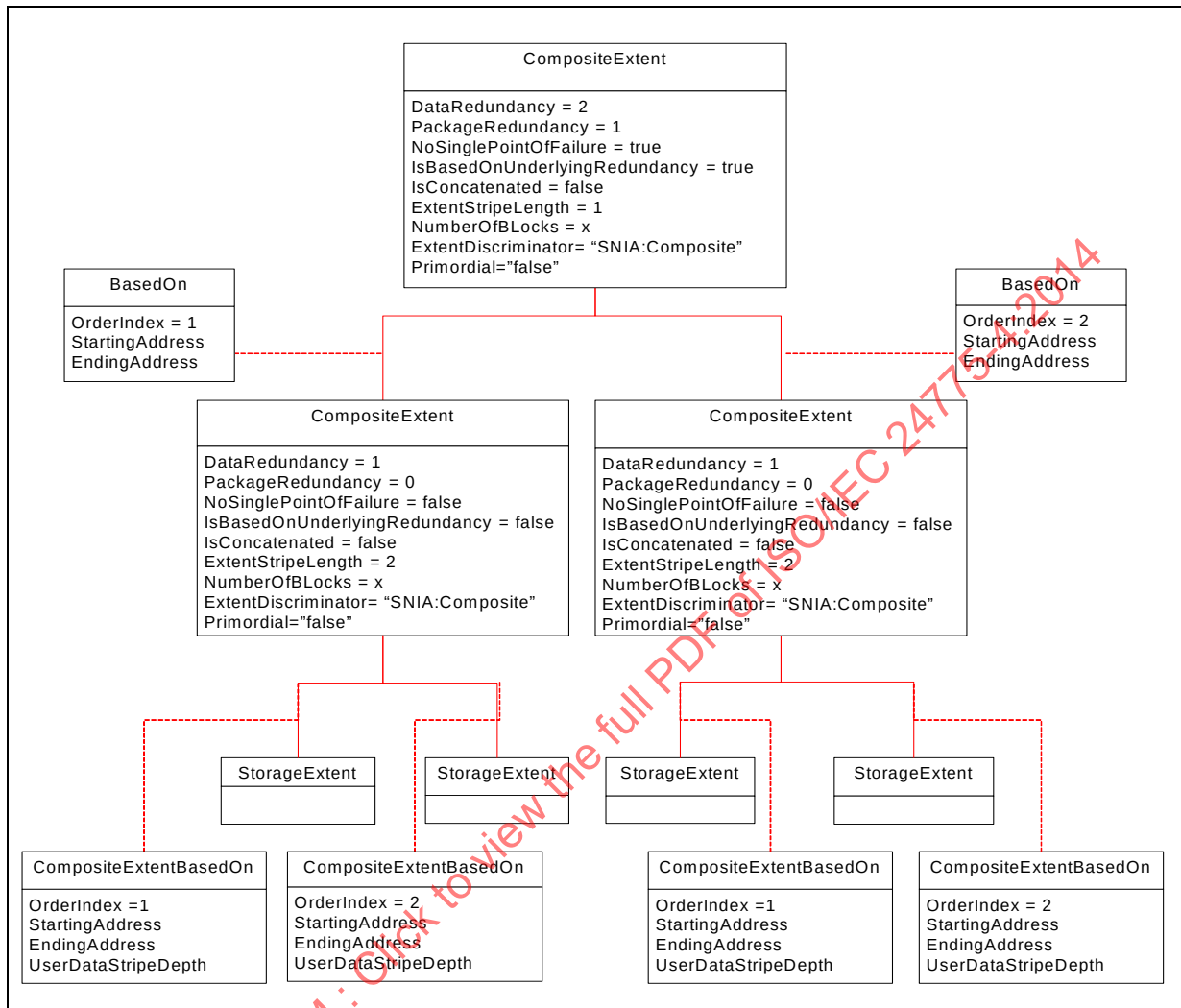


Figure 75 - RAID0+1 Composition

15.1.7.4.7 RAID4 or 5

Figure 76: "RAID4, 5 Composition" shows a partial instance diagram for a RAID4 or 5 Volume or Pool.

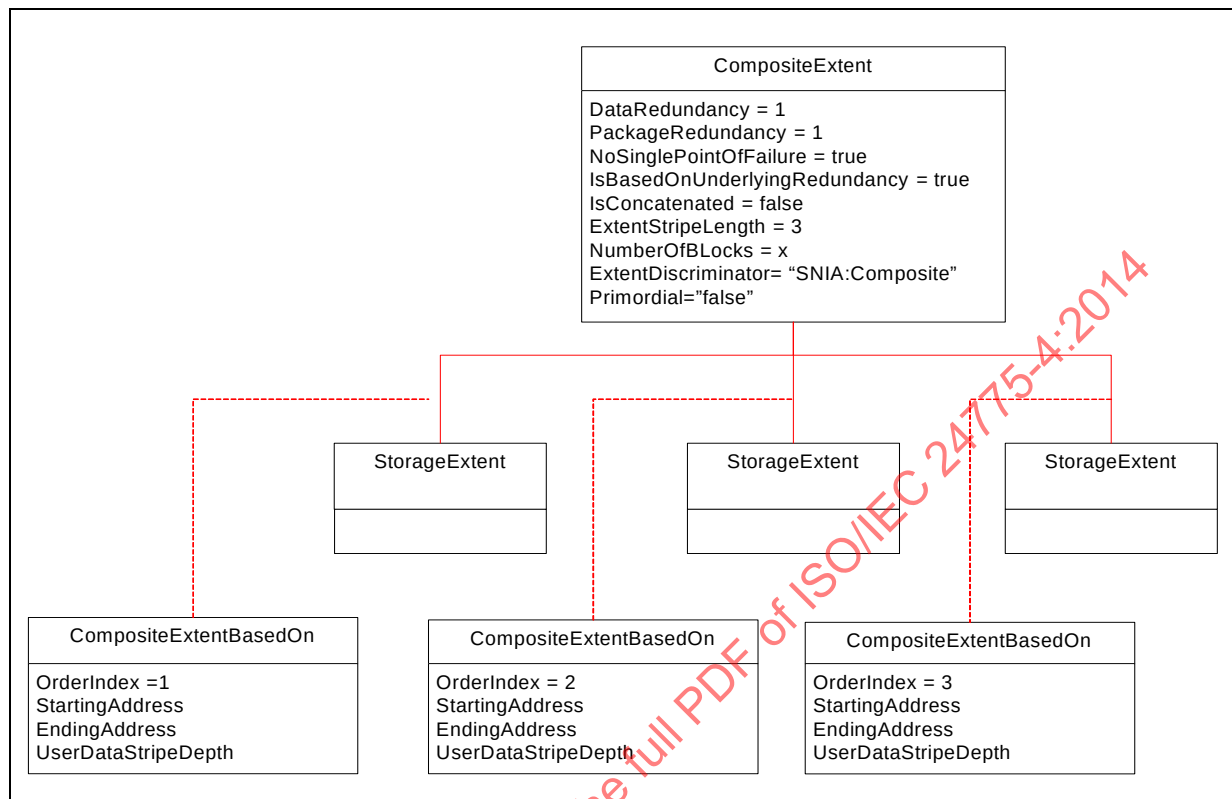


Figure 76 - RAID4, 5 Composition

15.1.7.4.8 RAID6, 5DP, and 4DP

Figure 77: "RAID 6, 5DP, 4DP" shows a partial instance diagram for a RAID6, 5DP, or 4DP Volume or Pool. Note that the PackageRedundancy is 2, indicating that two of the antecedent extents can fail simultaneously without loss of data. Four extents are shown, the minimum required for these double parity RAID organizations.

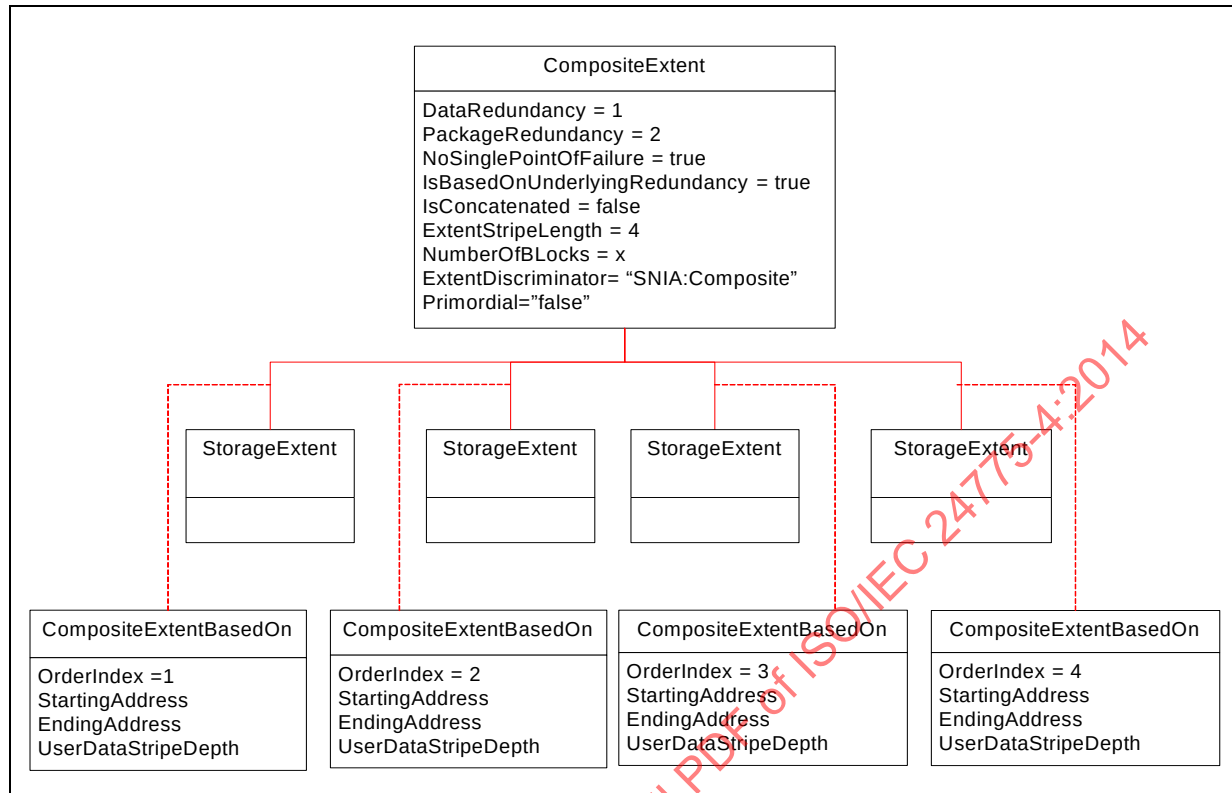


Figure 77 - RAID 6, 5DP, 4DP

15.1.7.4.9 RAID 15

Figure 78: "RAID15 Composition" shows a partial instance diagram for a RAID15 Volume or Pool. In this example the Data and Package Redundancy reflect the Quality of Service of the combined RAID Level, not just the top level composition which by itself is a simple RAID5.

NOTE Only CompositeExtent members 1 and 3 of the Raid 5 layer are shown.

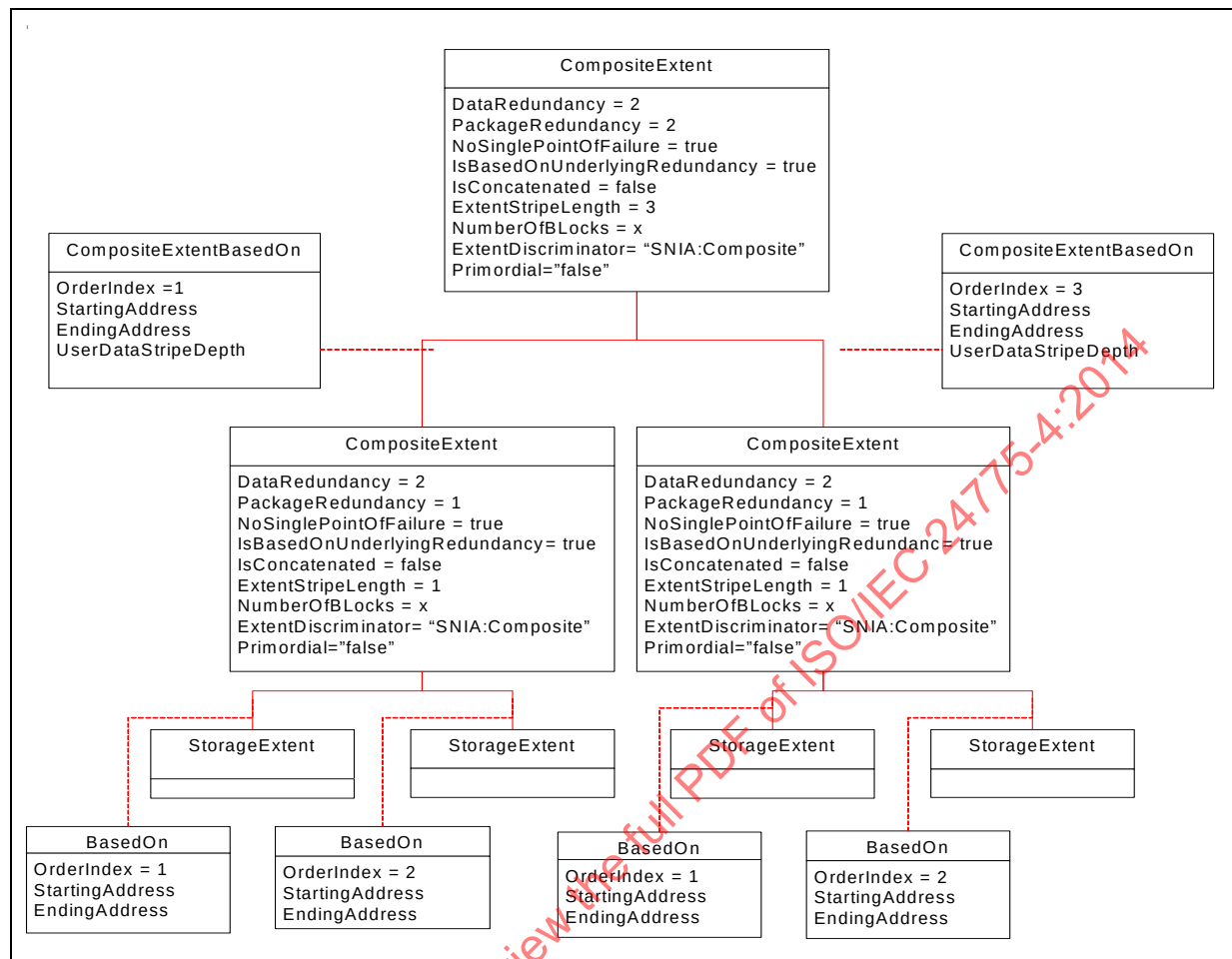


Figure 78 - RAID15 Composition

15.1.7.4.10 RAID50

Figure 79: "RAID50 Composition" shows a partial instance diagram for a RAID50 Volume or Pool. In this example the Data and Package Redundancy reflect the Quality of Service of the combined RAID Level, not just the top level composition which by itself is a non-redundant stripeset.

NOTE In the Raid 5 layer, CompositeExtent member 2 in each stripe member is not shown.

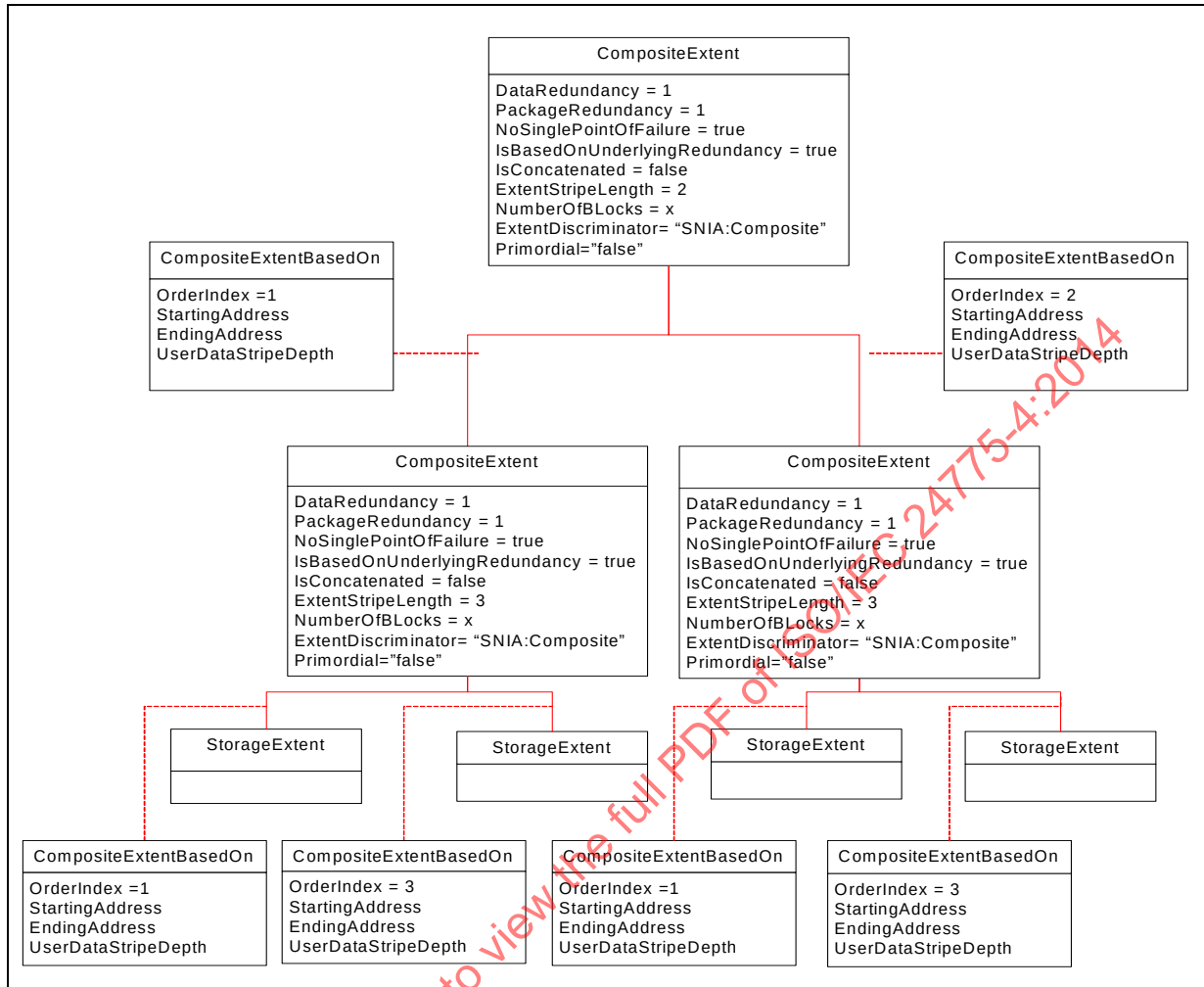


Figure 79 - RAID50 Composition

15.1.7.4.11 RAID51

Figure 80: "RAID51 Composition" shows a partial instance diagram for a RAID51 Volume or Pool. In this example the Data and Package Redundancy reflect the Quality of Service of the combined RAID Level, not just the top level composition which by itself is a simple mirror. That is, the top level is a RAID1, but the PackageRedundancy is 2, indicating the QOS for the entire hierarchy.

NOTE In the Raid 5 layer, CompositeExtent member 2 in each mirror is not shown.

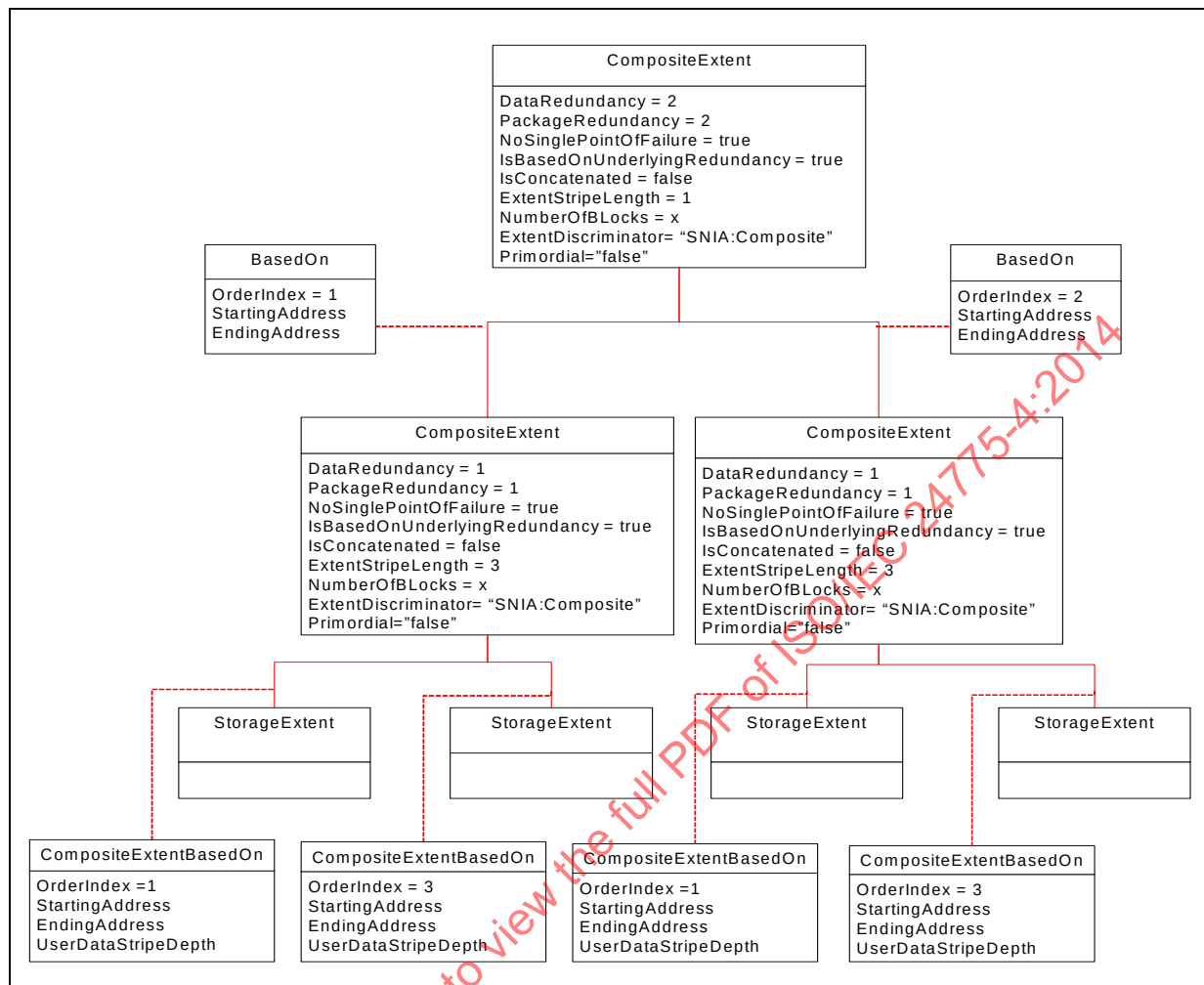


Figure 80 - RAID51 Composition

15.2 Health and Fault Management Considerations

Not defined in this International Standard.

15.3 Cascading Considerations

None.

15.4 Supported Subprofiles and Packages

Related Profiles for Extent Composition: Not defined in this International Standard.

15.5 Methods of the Profile

None.

15.6 Client Considerations and Recipes

15.6.1 Traverse the virtualization hierarchy of a StorageVolume or LogicalDisk

```
// DESCRIPTION
//
// This recipes defines a mechanism for traversing the extent hierarchy
// between
// the Exposable Block Storage Element and the Primordial Extents it
// makes use
// of, determining the RAID level structure, Concrete and Primordial pool
// membership.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTIONS
// 1. The instance name for an exposable block storage element (e.g.
// StorageVolume, LogicalDisk) of interest has been previously identified
// as
// $BlockElement->.

// This function determines if an Extent is a Primary(non-remaining)
// Component
// of a Pool.
//
sub boolean IsPrimaryComponent(REF $TargetExtent->) {
    $Pools->[] = AssociatorNames($TargetExtent->,
        "CIM_ConcreteComponent",
        "CIM_StoragePool",
        "PartComponent",
        "GroupComponent")

    if ($Pools->[] != null && $Pools->[].length == 1) {
        // This Extent is a Component Extent of either a Concrete
        // or Primordial pool
        return true
    }
    else
        return false
}

// This function determines the RAID Level or Quality of Service of a
// CompositeExtent and then recursively traverses the hierarchy beneath
// it.
//
sub void traverseComposition(REF $Composite->) {

    // See if this composite is a Primary(non-remaining) Component
    // Extent of a Pool (for information only.)
```



```

#PrimaryComponent = &IsPrimaryComponent($Composite->)

// Get the instances of the associations in which this Extent is the
// Dependent reference. The association instances retrieved should be
// either BasedOn or CompositeExtentBasedOn.
$Associations[] = References($Composite->,
    NULL,
    "Dependent",
    false,
    false,
    NULL)

// Now get the underlying extents
$TargetExtents->[] = AssociatorNames($Composite->,
    Associations[0].getClassName(),
    NULL,
    "Dependent",
    "Antecedent")

// Examine the QOS of the current level's Composite Extent
$CompositeExtent = GetInstance($Composite->,
    false,
    false,
    false,
    {"IsConcatenated", "ExtentStripeLength",
    "IsBasedOnUnderlyingRedundancy"})

if (($Associations[0] ISA CIM_CompositeExtentBasedOn)
    && ($CompositeExtent.IsConcatenated == false)
    && ($CompositeExtent.ExtentStripeLength > 1)) {

    // The TargetExtents are striped together. Get the Stripe Depth
    // from
    // the first association. The assumption here is that this
    // property is
    // the same for each association instance.
    #StripeDepth = $Associations[0].UserDataStripeDepth

    // Inspect the RAID level.
    #RAID = 0
    if ($CompositeExtent.IsBasedOnUnderlyingRedundancy) {
        #RAID = 5
    }
} else {
    // Associations are CIM_BasedOn, So this is either a Mirror or
    // a Concatenation
    if (($CompositeExtent.IsBasedOnUnderlyingRedundancy == true)
        && ($CompositeExtent.IsConcatenated == false)
        && ($CompositeExtent.ExtentStripeLength == 1)) {
        // The TargetExtents are mirrored together,
        // This level is a RAID 1
    }
}

```

```

    } else if (($CompositeExtent.IsBasedOnUnderlyingRedundancy ==
        false)
        && ($CompositeExtent.IsConcatenated == true)
        && ($CompositeExtent.ExtentStripeLength == 1)) {
        // The TargetExtents are concatenated together,
        // This level is a JBOD.
    } else {
        <ERROR! Illegal combination of property values; does not
            correspond to supported composition type.>
    }
}

// Now for each underlying extent at this level, traverse the sub-
// tree
// it is the sub-root of. If the extent is a CompositeExtent, then
// this
// is part of a complex RAID level; recursively invoke the
// Composition
// Algorithm. Otherwise it is just a regular StorageExtent and thus
// either a Primordial or decomposed from an Antecedent, so invoke
// the
// recursive Decomposition Algorithm.
for (#i in $TargetExtents->[]) {
    if ($TargetExtents->[#i] ISA CIM_CompositeExtent) {
        &traverseComposition($TargetExtents->[#i])
    } else {
        &traverseDecomposition($TargetExtents->[#i])
    }
}

}

// This function recursively traverses the hierarchy below a non-
// Composite
// Storage Extent.
sub void traverseDecomposition(REF $SubjectExtent->) {
    // See if this extent is a Primary(non-remaining) Component
    // Extent of a Pool (for information only.)
    #PrimaryComponent = &IsPrimaryComponent($SubjectExtent->)

    // Check here to see if we have reached the leaves of the hierarchy
    $SubjectExtent = GetInstance($SubjectExtent->,
        false,
        false,
        false,
        {"Primordial"})

    if ($SubjectExtent.Primordial == true) {
        // Recursion ends with each Primordial Extent.
        <EXIT: Recursion ends with each Primordial Extent.>
    } else {

```

```

// The Subject Extent is allocated partially or in full from the
// Antecedent Extent, so a single BasedOn is expected.
$TargetExtents[] = Associators($SubjectExtent->,
    "CIM_BasedOn",
    "CIM_StorageExtent",
    "Dependent",
    "Antecedent",
    false,
    false,
    {"Primordial"})

// Since the Subject Extent is allocated from the Antecedent,
// there can
// only be one Antecedent.
if ($TargetExtents[] == null || $TargetExtents[].length != 1) {
    <ERROR! Extent allocated from multiple Antecedents>
}
$TargetExtent = $TargetExtents[0]

if ($TargetExtent ISA CIM_CompositeExtent) {
    // This is a Composite Extent representing a RAID Level.
    // Since we
    // encountered the Composite in a decomposition, the
    // Dependent/Antecedent relationship falls into one of the
    // following scenarios:
    //
    // o The Subject Extent is a StorageVolume that is one-to-
    //   one with
    //   the Target Composite Extent.
    //
    // o The Subject Extent is a StorageVolume partially
    //   allocated from
    //   the Target Composite Extent, where the Composite is a
    //   RAID Group.
    //
    // o The Subject Extent is a ComponentExtent of a Concrete
    //   pool and is
    //   partially allocated from the Target Composite Extent
    //   where the
    //   Composite is a RAID Group.
    //
    // Call the (recursive) function to analyze the sub-hierarchy
    // composed by the Target Extent.
    //
    &traverseComposition($TargetExtent.getObjectPath())
} else {
    // The Antecedent is a regular StorageExtent and was not
    // Primordial, so it must be in turn a dependent decomposed
    // from an Antecedent, so invoke
    // ourselves recursively.
    &traverseDecomposition($TargetExtent.getObjectPath())
}

```

```
    }  
  
  }  
}  
  
// MAIN  
// Since the exposable block element is either one-to-one with the  
//      initial  
// CompositeExtent, or a partial allocation of it (in the case of a RAID  
//      Group),  
// decompose the block hierarchy.  
//  
&traverseDecomposition($BlockElement->)
```

15.6.2 Find the Primordial Extents used by a Storage Volume or Logical Disk

A storage administrator may want the information provided by this recipe for several reasons:

Failure Exposure: To understand what Drive or virtualized Volume failures may affect the health of a block storage element, or conversely what block storage elements are affected by a given Drive failure.

Performance and Loading: To avoid locating frequently accessed Volumes on the same Disk Drive.

Utilization: To avoid locating portions of too many volumes on the same Drive while leaving other drives under utilized.

```
// DESCRIPTION  
//  
// This recipe defines a mechanism for finding the Primordial Storage  
//      Extents  
// used by a Storage Volume in an Array or Virtualizer, or a LogicalDisk  
//      in  
// a Volume Manager or NAS system.  
//  
// PRE-EXISTING CONDITIONS AND ASSUMPTIONS  
// 1. The instance name for an exposable block storage element (e.g.  
// StorageVolume, LogicalDisk) of interest has been previously identified  
//      as  
// $BlockElement->.  
  
// This function recursively searches for the Primordial Storage Extents  
//      that  
// comprise the specified block storage element.  
sub $PrimordialExtents[] findPrimordials(REF $SubjectExtent->) {  
  
  // Get the Extents that are Antecedent to the specified Extent.  
  //  
  $TargetExtents[] = Associators($SubjectExtent->,  
    "CIM_BasedOn",  
    "CIM_StorageExtent",  
    "Dependent",  
    "Antecedent",  
    false,
```

```

        false,
        {"Primordial"})

// Examine each Extent at the next level to determine if its
// Primordial.

#i = 0
for (#j in $TargetExtents[]) {
    if ($TargetExtents[#j].Primordial == true) {
        // The Extent is Primordial, the recursion ends here. Add it to
        // the group of Primordials gathered at this level or below.
        $PrimordialExtents[#i++] = TargetExtents[j]
    } else {
        // The Extent is not Primordial, but it must be based on a
        // sub-hierarchy in which each leaf is a Primordial, so call
        // this
        // function Recursively.
        $SubordinatePrimordialExtents[] =
            &findPrimordials(TargetExtents[#j].getObjectPath())
        if ($SubordinatePrimordialExtents[] == null
            || $SubordinatePrimordialExtents[].length == 0) {
            <ERROR! Found a Leaf Extent that is not a Primordial>
        }

        for (#k in $SubordinatePrimordialExtents[]) {
            // The recursion delivers the bottom for each branch
            // These need to be collected and added into the whole
            $PrimordialExtents[#i++] = SubordinatePrimordialExtents[#k]
        }
    }
}
return ($PrimordialExtents[])
}

// MAIN
// Make initial call to the recursive function.
$PrimordialExtents[] = &findPrimordials($BlockElement->)
if ($PrimordialExtents[] == null || $PrimordialExtents[].length == 0) {
    <ERROR! No Primordials Found>
} else {
    <EXIT: Primordial Extents accumulated>
}

```

15.7 Registered Name and Version

Extent Composition version 1.5.0 (Component Profile).

15.8 CIM Elements

15.8.1 General

Table 317 describes the CIM elements for Extent Composition.

Table 317 - CIM Elements for Extent Composition

Element Name	Requirement	Description
15.8.2 CIM_AssociatedComponentExtent (Pool Component to Concrete Pool)	Mandatory	
15.8.3 CIM_AssociatedRemainingExtent (Pool to its remaining extents)	Mandatory	
15.8.4 CIM_BasedOn (Mid level BasedOn)	Optional	Associates a Storage Extent (Pool Component or Intermediate) to underlying Storage Extents it is based on.
15.8.5 CIM_BasedOn (Top level BasedOn)	Mandatory	Associates a StorageVolume (or LogicalDisk) to the underlying Storage Extent it is based on.
15.8.6 CIM_CompositeExtent (Composite Intermediate)	Optional	Represents a Concrete StorageExtent that is a composite and does not have an AssociatedComponentExtent association to a Concrete StoragePool.
15.8.7 CIM_CompositeExtent (Composite Pool Component)	Optional	Represents a Concrete StorageExtent that is a composite and has an AssociatedComponentExtent association to a Concrete StoragePool.
15.8.8 CIM_CompositeExtentBasedOn	Optional	Associates a Composite Extent representing a striping simple RAID organization such as RAID 0 or RAID 5 to the underlying Storage Extents that it virtualizes.
15.8.9 CIM_ConcreteComponent (Pool Component to Concrete Pool)	Mandatory	Deprecated. Associate the extents that are playing the Pool Component role to their aggregating StoragePool.
15.8.10 CIM_ConcreteComponent (Remaining Extent to Pool)	Mandatory	Deprecated. Associate a remaining extent to the StoragePool for which it represents unused space.
15.8.11 CIM_FilterCollection (Extent Composition Predefined FilterCollection)	Conditional	Experimental. Conditional requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='5' (Predefined Filter Collections). This is a collection of predefined IndicationFilters to which a client may subscribe.
15.8.12 CIM_HostedCollection (System to predefined IndicationFilters)	Conditional	Experimental. Conditional requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='5' (Predefined Filter Collections).
15.8.13 CIM_MemberOfCollection (Extent Composition Filter Collection to FilterCollection)	Conditional	Experimental. Conditional requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='5' (Predefined Filter Collections). This associates the Extent Composition predefined FilterCollection to the FilterCollection for the autonomous profile (e.g., the Array FilterCollection).
15.8.14 CIM_MemberOfCollection (Predefined Filter Collection to Extent Composition Filters)	Optional	Experimental. This associates the Extent Composition predefined FilterCollection to the predefined Filters supported by the implementation.

Table 317 - CIM Elements for Extent Composition

Element Name	Requirement	Description
15.8.15 CIM_StorageExtent (Intermediate)	Optional	Represents a Concrete StorageExtent that is not a composite and does not have an AssociatedComponentExtent association to a Concrete StoragePool.
15.8.16 CIM_StorageExtent (Pool Component)	Optional	Represents a Concrete StorageExtent that is not a composite and has an AssociatedComponentExtent association to a Concrete StoragePool.
15.8.17 CIM_StorageExtent (Remaining)	Optional	Represents a Concrete StorageExtent that identifies unused space in a Concrete StoragePool and has an AssociatedRemainingExtent association to that Concrete StoragePool.
15.8.18 CIM_SystemDevice (Composite Extent System)	Optional	Associates a CompositeExtent to a hosting computer system.
15.8.19 CIM_SystemDevice (Storage Extent System)	Optional	Associates a StorageExtent to a hosting computer system.

15.8.2 CIM_AssociatedComponentExtent (Pool Component to Concrete Pool)

The referenced StorageExtent represents capacity has not been allocated, is allocated in part, or is allocated in its entirety.

Requirement: Mandatory.

Table 318 describes class CIM_AssociatedComponentExtent (Pool Component to Concrete Pool).

Table 318 - SMI Referenced Properties/Methods for CIM_AssociatedComponentExtent (Pool Component to Concrete Pool)

Properties	Flags	Requirement	Description and Notes
GroupComponent		Mandatory	The (non-empty) Concrete StoragePool.
PartComponent		Mandatory	The storage extent or composite extent that is a component of the concrete storage pool.

15.8.3 CIM_AssociatedRemainingExtent (Pool to its remaining extents)

The referenced StorageExtent represents the capacity of the StorageExtent on which it is based that was not used in resource allocation.

Requirement: Mandatory.

Table 319 describes class CIM_AssociatedRemainingExtent (Pool to its remaining extents).

Table 319 - SMI Referenced Properties/Methods for CIM_AssociatedRemainingExtent (Pool to its remaining extents)

Properties	Flags	Requirement	Description and Notes
GroupComponent		Mandatory	The (non-empty, Concrete or Primordial) StoragePool.
PartComponent		Mandatory	The storage extent that represents free space in the concrete storage pool.

15.8.4 CIM_BasedOn (Mid level BasedOn)

Created By: External

Modified By: External

Deleted By: External

Requirement: Optional.

Table 320 describes class CIM_BasedOn (Mid level BasedOn).

Table 320 - SMI Referenced Properties/Methods for CIM_BasedOn (Mid level BasedOn)

Properties	Flags	Requirement	Description and Notes
StartingAddress		Optional	
EndingAddress		Optional	
Dependent		Mandatory	The Storage Extent (Pool Component, Intermediate, Composite Intermediate, Composite Pool Component or Remaining) that is based on underlying extents.
Antecedent		Mandatory	The underlying extents. They may be intermediate or Pool Components and they may be composite or uncomposed.

15.8.5 CIM_BasedOn (Top level BasedOn)

Created By: External

Modified By: External

Deleted By: External

Requirement: Mandatory.

Table 321 describes class CIM_BasedOn (Top level BasedOn).

Table 321 - SMI Referenced Properties/Methods for CIM_BasedOn (Top level BasedOn)

Properties	Flags	Requirement	Description and Notes
StartingAddress		Optional	
EndingAddress		Optional	
Dependent		Mandatory	The Storage Volume or Logical Disk that depends on the associated extent.
Antecedent		Mandatory	The extent on which the storage volume or logical disk is based.

15.8.6 CIM_CompositeExtent (Composite Intermediate)

Instances of this class with the discriminator of 'SNIA:Intermediate' and 'SNIA:Composite' are Concrete StorageExtents that are a composite and do not have an AssociatedComponentExtent association to a Concrete StoragePool.

Created By: External

Modified By: External

Deleted By: External

Requirement: Optional.

Table 322 describes class CIM_CompositeExtent (Composite Intermediate).

Table 322 - SMI Referenced Properties/Methods for CIM_CompositeExtent (Composite Intermediate)

Properties	Flags	Requirement	Description and Notes
Name	CD	Mandatory	
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	
ExtentStatus		Mandatory	
DataRedundancy		Mandatory	
PackageRedundancy		Mandatory	
NoSinglePointOfFailure		Mandatory	
IsBasedOnUnderlyingRedundancy		Mandatory	
IsConcatenated		Mandatory	
ExtentStripeLength		Mandatory	
NumberOfBlocks		Mandatory	
ConsumableBlocks		Mandatory	The number of usable blocks.
BlockSize		Mandatory	
Primordial		Mandatory	This shall be 'false' for extents instantiated in Extent Composition.
ExtentDiscriminator		Mandatory	Experimental. This is array of values that shall contain 'SNIA:Intermediate' and 'SNIA:Composite'.

15.8.7 CIM_CompositeExtent (Composite Pool Component)

Instances of this class with the discriminator of 'SNIA:Pool Component' and 'SNIA:Composite' are Concrete StorageExtents that are a composite and have an AssociatedComponentExtent association to a Concrete StoragePool.

Created By: External

Modified By: External

Deleted By: External

Requirement: Optional.

Table 323 describes class CIM_CompositeExtent (Composite Pool Component).

Table 323 - SMI Referenced Properties/Methods for CIM_CompositeExtent (Composite Pool Component)

Properties	Flags	Requirement	Description and Notes
Name	CD	Mandatory	
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	

**Table 323 - SMI Referenced Properties/Methods for CIM_CompositeExtent
(Composite Pool Component)**

Properties	Flags	Requirement	Description and Notes
ExtentStatus		Mandatory	
DataRedundancy		Mandatory	
PackageRedundancy		Mandatory	
NoSinglePointOfFailure		Mandatory	
IsBasedOnUnderlyingRedundancy		Mandatory	
IsConcatenated		Mandatory	
ExtentStripeLength		Mandatory	
NumberOfBlocks		Mandatory	
ConsumableBlocks		Mandatory	The number of usable blocks.
BlockSize		Mandatory	
Primordial		Mandatory	This shall be 'false' for extents instantiated in Extent Composition.
ExtentDiscriminator		Mandatory	Experimental. This is array of values that shall contain 'SNIA:Pool Component' and 'SNIA:Composite'.

15.8.8 CIM_CompositeExtentBasedOn

Created By: External

Modified By: External

Deleted By: External

Requirement: Optional.

Table 324 describes class CIM_CompositeExtentBasedOn.

Table 324 - SMI Referenced Properties/Methods for CIM_CompositeExtentBasedOn

Properties	Flags	Requirement	Description and Notes
StartingAddress		Optional	
EndingAddress		Optional	
OrderIndex		Mandatory	Indicates the order in which the antecedent extents have blocks striped onto them.
UserDataStripeDepth		Mandatory	The number of blocks written to an Antecedent extent before moving on to the next extent. Although this property is on the association class, its values shall be the same for each instance of the association with the same Dependent CompositeExtent.
Dependent		Mandatory	The composite extent that is based on underlying extents.
Antecedent		Mandatory	The extents on which the composite extent is based. They may be intermediate or pool component extents and they may be either other composite extents or uncomposed extents.

15.8.9 CIM_ConcreteComponent (Pool Component to Concrete Pool)

Deprecated. Associate the extents that are playing the Pool Component role to their aggregating StoragePool. This is Deprecated since its function is better covered by AssociatedComponentExtent.

Created By: External

Modified By: External

Deleted By: External

Requirement: Mandatory.

Table 325 describes class CIM_ConcreteComponent (Pool Component to Concrete Pool).

**Table 325 - SMI Referenced Properties/Methods for CIM_ConcreteComponent
(Pool Component to Concrete Pool)**

Properties	Flags	Requirement	Description and Notes
GroupComponent		Mandatory	The (non-empty) Concrete StoragePool.
PartComponent		Mandatory	The storage extent or composite extent that is a component of the concrete storage pool.

15.8.10 CIM_ConcreteComponent (Remaining Extent to Pool)

Deprecated.

Created By: External

Modified By: External

Deleted By: External

Requirement: Mandatory.

Table 326 describes class CIM_ConcreteComponent (Remaining Extent to Pool).

**Table 326 - SMI Referenced Properties/Methods for CIM_ConcreteComponent
(Remaining Extent to Pool)**

Properties	Flags	Requirement	Description and Notes
GroupComponent		Mandatory	The (non-empty) StoragePool.
PartComponent		Mandatory	The storage extent that represents unused space in the storage pool.

15.8.11 CIM_FilterCollection (Extent Composition Predefined FilterCollection)

Experimental. This is a collection of predefined IndicationFilters to which a client may subscribe. An Extent Composition implementation shall indicate support for predefined FilterCollections by the SNIA_IndicationConfigurationCapabilities.FeaturesSupported = '5' (Predefined Filter Collections).

Requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='5' (Predefined Filter Collections).

Table 327 describes class CIM_FilterCollection (Extent Composition Predefined FilterCollection).

**Table 327 - SMI Referenced Properties/Methods for CIM_FilterCollection
(Extent Composition Predefined FilterCollection)**

Properties	Flags	Requirement	Description and Notes
InstanceID		Mandatory	Shall specify the unique identifier for an instance of this class within the Implementation namespace.
CollectionName		Mandatory	The value of CollectionName shall be 'SNIA:Extent Composition'.

15.8.12 CIM_HostedCollection (System to predefined IndicationFilters)

Experimental.

Requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='5' (Predefined Filter Collections).

Table 328 describes class CIM_HostedCollection (System to predefined IndicationFilters).

**Table 328 - SMI Referenced Properties/Methods for CIM_HostedCollection
(System to predefined IndicationFilters)**

Properties	Flags	Requirement	Description and Notes
Dependent		Mandatory	Reference to the predefined FilterCollection for Extent Composition.
Antecedent		Mandatory	Reference to the 'Top level' System.

15.8.13 CIM_MemberOfCollection (Extent Composition Filter Collection to FilterCollection)

Experimental. This associates the Extent Composition predefined FilterCollection to the FilterCollection for the autonomous profile (e.g., the Array FilterCollection).

Requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='5' (Predefined Filter Collections).

Table 329 describes class CIM_MemberOfCollection (Extent Composition Filter Collection to FilterCollection).

**Table 329 - SMI Referenced Properties/Methods for CIM_MemberOfCollection
(Extent Composition Filter Collection to FilterCollection)**

Properties	Flags	Requirement	Description and Notes
Collection		Mandatory	Reference to the Extent Composition predefined FilterCollection.
Member		Mandatory	Reference to the Extent Composition predefined FilterCollection.

15.8.14 CIM_MemberOfCollection (Predefined Filter Collection to Extent Composition Filters)

Experimental. This associates the Extent Composition predefined FilterCollection to the predefined Filters supported by the implementation.

Requirement: Optional.

Table 330 describes class CIM_MemberOfCollection (Predefined Filter Collection to Extent Composition Filters).

**Table 330 - SMI Referenced Properties/Methods for CIM_MemberOfCollection
(Predefined Filter Collection to Extent Composition Filters)**

Properties	Flags	Requirement	Description and Notes
Collection		Mandatory	Reference to the Extent Composition predefined FilterCollection.
Member		Mandatory	Reference to the predefined IndicationFilters of the Extent Composition implementation.

15.8.15 CIM_StorageExtent (Intermediate)

Instances of this class with the discriminator of 'SNIA:Intermediate' are Concrete StorageExtents that are not a composite and do not have an AssociatedComponentExtent association to a Concrete StoragePool.

Created By: External

Modified By: External

Deleted By: External

Requirement: Optional.

Table 331 describes class CIM_StorageExtent (Intermediate).

Table 331 - SMI Referenced Properties/Methods for CIM_StorageExtent (Intermediate)

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	
ExtentStatus		Mandatory	
NumberOfBlocks		Mandatory	
ConsumableBlocks		Mandatory	The number of usable blocks.
BlockSize		Mandatory	
Primordial		Mandatory	This shall be 'false' for extents instantiated in Extent Composition.
ExtentDiscriminator		Mandatory	Experimental. This is array of values that shall contain 'SNIA:Intermediate'.

15.8.16 CIM_StorageExtent (Pool Component)

Instances of this class with the discriminator of 'SNIA:Pool Component' are Concrete StorageExtents that are not a composite and have an AssociatedComponentExtent association to a Concrete StoragePool.

Created By: External

Modified By: External

Deleted By: External

Requirement: Optional.

Table 332 describes class CIM_StorageExtent (Pool Component).

Table 332 - SMI Referenced Properties/Methods for CIM_StorageExtent (Pool Component)

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	
ExtentStatus		Mandatory	
NumberOfBlocks		Mandatory	
ConsumableBlocks		Mandatory	The number of usable blocks.
BlockSize		Mandatory	
Primordial		Mandatory	This shall be 'false' for extents instantiated in Extent Composition.
ExtentDiscriminator		Mandatory	Experimental. This is an array of values that shall contain 'SNIA:Pool Component'.

15.8.17 CIM_StorageExtent (Remaining)

Instances of this class with the discriminator of 'SNIA:Remaining' are Concrete StorageExtents that are not a composite and have an AssociatedRemainingExtent association to the Concrete StoragePool for which they represent free space.

Created By: External

Modified By: External

Deleted By: External

Requirement: Optional.

Table 333 describes class CIM_StorageExtent (Remaining).

Table 333 - SMI Referenced Properties/Methods for CIM_StorageExtent (Remaining)

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	
ExtentStatus		Mandatory	
NumberOfBlocks		Mandatory	
ConsumableBlocks		Mandatory	The number of usable blocks.
BlockSize		Mandatory	
Primordial		Mandatory	This shall be 'false' for extents instantiated in Extent Composition.
ExtentDiscriminator		Mandatory	Experimental. This is an array of values that shall contain 'SNIA:Remaining'.

15.8.18 CIM_SystemDevice (Composite Extent System)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 334 describes class CIM_SystemDevice (Composite Extent System).

**Table 334 - SMI Referenced Properties/Methods for CIM_SystemDevice
(Composite Extent System)**

Properties	Flags	Requirement	Description and Notes
GroupComponent		Mandatory	A reference to an instance of Computer System.
PartComponent		Mandatory	A reference to an instance of CIM_CompositeExtent (Composite Intermediate or Composite Pool Component) used in this profile.

15.8.19 CIM_SystemDevice (Storage Extent System)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 335 describes class CIM_SystemDevice (Storage Extent System).

**Table 335 - SMI Referenced Properties/Methods for CIM_SystemDevice
(Storage Extent System)**

Properties	Flags	Requirement	Description and Notes
GroupComponent		Mandatory	A reference to an instance of Computer System.
PartComponent		Mandatory	A reference to an instance of CIM_StorageExtent (Intermediate, Pool Component or Remaining) used in this profile.

STABLE

DEPRECATED

16 LUN Creation Subprofile

The functionality of the LUN Creation and Pool Manipulation Capabilities, and Settings Subprofiles have been subsumed by Clause 6 Block Services Package.

The LUN Creation Subprofile is defined in 7.3.3.11 of SMI-S 1.0.2.

DEPRECATED

IECNORM.COM : Click to view the full PDF of ISO/IEC 24775-4:2014

DEPRECATED

17 Extent Mapping Subprofile

The functionality of the Extent Mapping Subprofile (see 7.3.3.5 of SMI-S 1.0.2) has been subsumed by the Extent Composition Subprofile (clause 15).

DEPRECATED

IECNORM.COM : Click to view the full PDF of ISO/IEC 24775-4:2014

DEPRECATED

18 LUN Mapping and Masking Subprofile

18.1 General

The LUN Mapping and Masking Subprofile (see 7.3.3.14 in SMI-S 1.0.2) has been replaced by Clause 19 Masking and Mapping Subprofile.

18.2 Compatibility with SMI-S 1.0 clients

Problems with the functionality and complexity of the LUN Mapping and Masking Subprofile in SMI-S 1.0 required some changes that may not be backwards compatible in the 1.1.0 version. The Mapping and Masking Subprofile now reduces the complexity by replacing the 1.0.2 extrinsic methods and severely constraining the valid combinations of parameters. Additionally, changes made to support non-FC transports and non-SCSI protocols also affect backwards compatibility. Specifically, associating the SCSIProtocolController to a SCSIProtocolEndpoint instead of LogicalPort. SCSIProtocolEndpoint is associated to the LogicalPort. Separating the port from the protocol allows the port to be used with non-SCSI protocols such as IP. Most of the model is identical, but new classes, properties, and methods have been added to simplify its operation. Some of the old methods are still used in 1.1.0.

Class and association changes to the model for 1.1.0:

- SAPAvailableForElement replaces the ProtocolControllerForPort association
- SCSIProtocolEndpoint replaces LogicalPort
- LogicalPort is associated to SCSIProtocolEndpoint via PortImplementsEndpoint (see Clause 18 LUN Mapping and Masking Subprofile in *Information Technology – Storage Management, Part 3 Common Profiles*.)
- AuthorizedPrivilege associations to SystemSpecificCollection via AuthorizedSubject associations are no longer allowed.

Instrumentation may be able to provide 1.0.2 and 1.0 compliant implementations in a single namespace, if the following conditions are met:

- ProtocolControllerMaskingCapabilities.ProtocolControllerSupportsCollections is false (StorageHardwareID instances are referenced directly by AuthorizedSubject associations).
- There is exactly a 1-1-1 relationship between instance of AuthorizedSubject, AuthorizedPrivilege, and AuthorizedTarget. In other words, Privilege instances cannot be shared.

If these criteria are not met, instrumentation could provide separate 1.0.2 and 1.1.0 implementations in separate CIM namespaces.

DEPRECATED

STABLE

19 Masking and Mapping Subprofile

19.1 Description

19.1.1 General

NOTE See 18.2 for notes on compatibility with the LUN Mapping and Masking Subprofile in SMI-S 1.0.2.

Many disk arrays provide an interface for the administrator to specify which initiators can access what volumes through which target ports. 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*.)

Given a storage system with no LUN masking or mapping, all hosts/initiators see the same elements when they discover a storage system. In a storage system supporting LUN Masking, logical units are masked (hidden) from SCSI initiators (Host Bus Adaptors) by default. The administrator uses the Masking and Mapping Subprofile to determine which logical units are visible (exposed) to specific initiators through which target ports. The 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 subsets - each subset 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 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.

19.1.2 Views and Paths

The key concepts for Device Masking and Mapping are view and path. A “view” is a list of logical units exposed to a list of initiators through a list of target ports, modeled as SCSIProtocolController (SPC) with associated LogicalDevices, StorageHardwareIDs, and SCSIProtocolEndpoints. The logical devices have logical unit numbers and access permissions relative to the view, modeled as DeviceNumber and DeviceAccess properties of the ProtocolControllerForUnit association. A full “path” is a combination of one of each logical unit, initiator port, and target port - the concept of path is independent from a CIM model, but a view expresses a combination of paths that comply with SCSI rules. In essence, an SPC serves as a collection of paths - each initiator ID is granted access to each logical unit through each target port.

In addition, there are partial and invalid states. A partial path is a path missing associations to instances of logical unit, initiator port, or target port. In practice, some arrays do not support partial paths and other arrays support some, but not all, configurations with partial paths. An SPC lacking associations to logical units, initiator ports, or target ports - as required by the underlying implementation - is in an invalid partial path state.

An invalid view state is a combination of classes and associations in the provider that does not map to a committed configuration of the underlying implementation. The 1.0 LUN Masking and Mapping interfaces required clients to perform multiple transactions to achieve a valid view, forcing providers to maintain invalid view states while waiting for the client to complete a sequence of transactions. This created non-interoperability when the providers only supported

transactions in a certain order, and when a second client looked at the model before a sequence of transactions was completed.

An SPC with no instances of one type of association (to initiators, targets, or LUs) with support from the instrumentation is in a valid partial path state. The result is that the SPC does not expose any valid SCSI paths. Instrumentation may support these states as convenience to clients - allowing a client to quickly activate/deactivate a configuration by adding/removing associations - or as an intermediate state between multiple ExposePath or HidePath requests. It is not mandatory in SMI-S to support these partial path states, but clients need to understand which partial path states are and are not valid.

19.1.3 Model Elements

The model uses three basic types of objects:

LogicalDevice, the superclass of volumes and tape drives representing SCSI logical units.

SCSIProtocolController - models the "view" described above.

SCSIProtocolEndpoint - models the SCSI protocol aspects of a port. A SCSIProtocolEndpoint is associated to one or more ports (modeled as subclasses of LogicalPort). SCSIProtocolEndpoint and classes (such as FCPort) representing ports are part of target port subprofiles.

These objects are related by two associations:

ProtocolControllerForUnit associates a SCSIProtocolController with its LogicalDevices. The controller-relative address (such as a SCSI Logical Unit Number) is modeled as the DeviceNumber property of ProtocolControllerForUnit.

SAPAvailableForElement associates a SCSIProtocolController to one or more SCSIProtocolEndpoints.

In this subprofile, the existence of a ControllerConfigurationService with a ConcreteDependency association to a SCSIProtocolController governs the high-level device mapping and masking policy for that protocol controller.

If the service does not exist, then regardless of host port, the policy is that **SAPAvailableForElement** associates SCSIProtocolController to all SCSIProtocolEndpoints that represent SCSI target behavior (that is, have Role property set to "Target").

If the service is present, then for a particular host port, the policy is that SAPAvailableForElement connects a SCSIProtocolController to a SCSIProtocolEndpoint only when access is explicitly granted.

Figure 81: "Generic System with no Configuration Service" and Figure 82: "Generic System with ControllerConfigurationService" depict an instance diagram of a generic storage system with dual-port access to four logical devices and an implementation with no device mapping and masking services. All of the LogicalDevices are exposed to all initiators with the same

DeviceNumber. Figure 81: "Generic System with no Configuration Service" depicts a configuration with no LUN Masking capabilities.

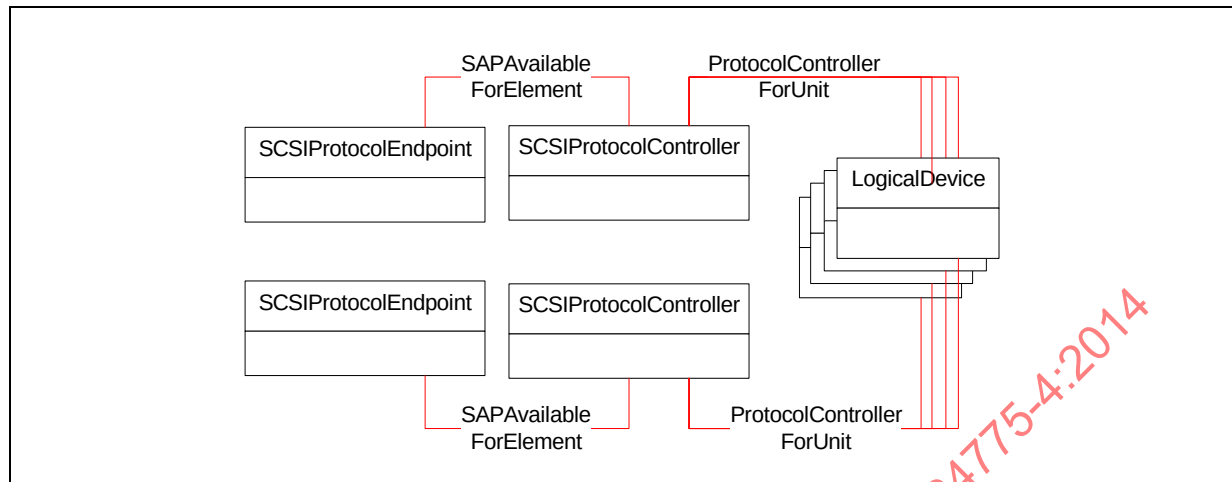


Figure 81 - Generic System with no Configuration Service

Figure 82: "Generic System with ControllerConfigurationService" 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.

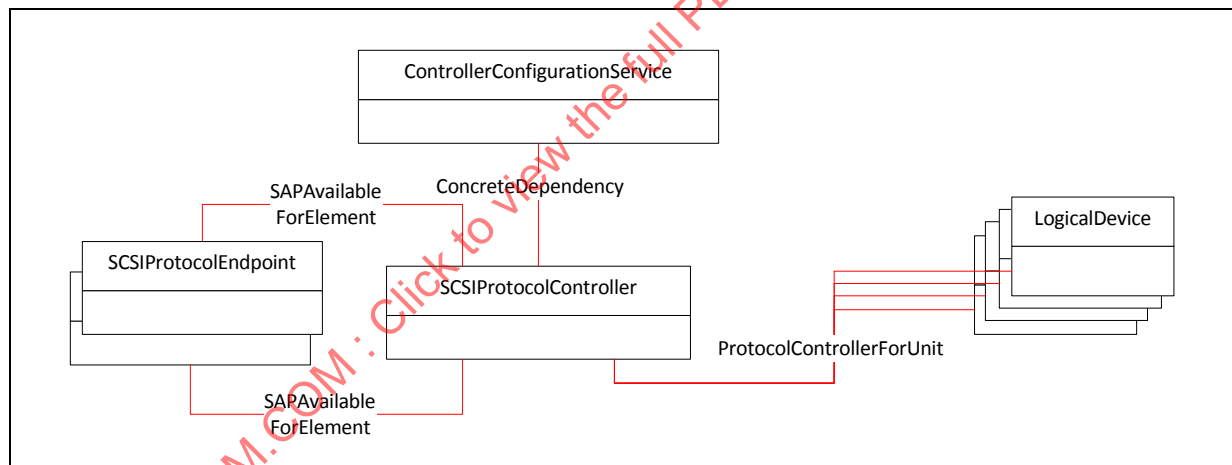


Figure 82 - Generic System with ControllerConfigurationService

The means to grant access is discussed in 19.5.1 "ExposePaths".

19.1.4 SCSIProtocolController Views

Device Masking limits the devices seen by particular host initiators (such as HBAs). For example, when a host discovers a device (using SCSI Report LUNs and Inquiry commands), it 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.

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 (SCSIProtocolController) through a target port. This view includes LogicalDevices explicitly exposed to specified initiators and "default access" 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 subset. In particular, a view:

- is associated with a set of LogicalDevices;
- may be exposed to zero or more host ports;
- is associated with one or more target device ports;
- 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).

For systems where access is granted through all or no target ports (where ProtocolControllerMaskingCapabilities.PortsPerView is set to “All Ports share the same View”), this rule is simpler – an initiator StorageHardwareID shall not be associated with more than one view (SCSIProtocolController).

- each LogicalDevice in a view shall have a unique DeviceNumber (SCSI logical unit number);
- a LogicalDevice may be in multiple views, and in each may be assigned the same or different DeviceNumbers (Logical Units).

The device uses the initiator port identifier to authorize access and to determine the view to present to the HBA. The initiator ID (such as FC Port WWN) is modeled as a subclass of Identity called StorageHardwareID. As used in this subprofile, AuthorizedSubject associates a AuthorizedPrivilege with a StorageHardwareID. As used in this subprofile, AuthorizedTarget associates an AuthorizedPrivilege with a SCSIProtocolController.

In this version of the subprofile, there is exactly a one-to-one-to-one relationship between AuthorizedSubject, AuthorizedPrivilege, and AuthorizedTarget. In other words, for each StorageHardwareID associated to a SCSIProtocolController, there will be unique instances of AuthorizedSubject, AuthorizedPrivilege, and AuthorizedTarget

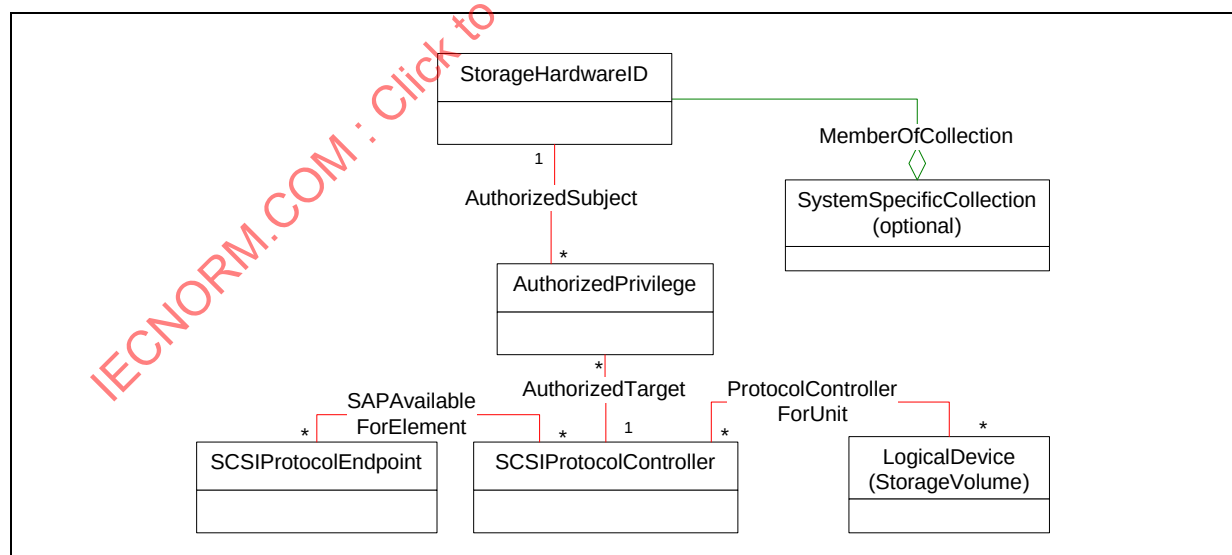


Figure 83 - Relationship of Initiator IDs, Endpoints, and Logical Units

19.1.5 Initiator ID Collections

An implementation may optionally model collections of Initiator IDs. This is modeled as depicted in Figure 83: "Relationship of Initiator IDs, Endpoints, and Logical Units". If the implementation supports collection of initiator IDs, the instrumentation shall set ProtocolControllerMaskingCapabilities.ProtocolControllerSupportsCollections to True.

19.1.6 Default View / Default Logical Unit Access

An implementation may expose some logical units to all initiators while restricting access to others. A default LUN exposes the same SCSI logical unit to all initiators, so adding a default LUN requires that the instrumentation assure that no existing logical-unit-view map uses that same logical unit address. Whenever a new SCSIProtocolController is created, it is automatically attached to all default LUNs.

This is modeled with a SCSIProtocolController that is associated via AuthorizedTarget to a AuthorizedPrivilege that is associated via AuthorizedSubject to a StorageHardwareID with an Name property set to null (not the zero-length string ""). These are known as **default protocol controllers** - exposing a view that is granted by default to all initiators, regardless of masking rules. If the implementation supports default protocol controllers, the instrumentation shall instantiate at least one default protocol controller when the instrumentation starts. The instrumentation shall reject any client attempt to delete a default protocol controller.

Only one null-name StorageHardwareID is allowed. It is associated to all default SPCs. No other StorageHardwareIDs may be associated to default SPCs. A target port can be associated with at most one default SPC.

If ProtocolControllerMaskingCapabilities.PortsPerView is not set to "All Ports share the same View", the instrumentation may support multiple default protocol controllers, but a target port shall not be associated to more than one default protocol controller.

A client requests a logical unit be given default access by associating with the default protocol controller using ExposeDefaultLUs method. The instrumentation shall ensure that the requested unit number is not used in any SCSIProtocolController connected to target ports associated with the default protocol controller. If the unit number is available, the logical unit is attached to the default protocol controller and all the other protocol controllers that share its target ports. Similarly, a client requests default access be removed from a logical unit by calling HideDefaultLUs, passing in a reference to the default protocol controller and the logical unit's ID.

19.1.7 Arbitrary Logical Units

If the implementation supports logical units for management (rather than storage), they shall be modeled with SCSIArbitraryLogicalUnit. If these management units are exposed regardless of masking access then they shall be associated to the default protocol controller.

19.1.8 Read-only versus Read-Write access

ExposePaths includes a DeviceAccess parameter that is used to set the DeviceAccess property of ProtocolControllerForUnit association.

19.1.9 Read-Only Volumes

An implementation may model a volume that is readable, but not writable to any initiator by setting StorageVolume.Access to "Readable" (1).

19.1.10 Finding Volumes that are not Mapped

A StorageVolume is considered mapped if it is exposed to an initiator. Instrumentation shall inform clients whether a volume is or is not mapped using the "In-Band Access Granted" value in StorageVolume.ExtentStatus array property. If a volume is associated with one or more protocol controllers and one of the associated protocol controllers is associated with one or more StorageHardwareIDs, the instrumentation shall set "In-Band Access Granted" in ExtentStatus. Otherwise, "In-Band Access Granted" shall not be set.

19.1.11 Limits on Map counts per Logical Unit

ProtocolControllerMaskingCapabilities.MaximumMapCount is the maximum number of times the underlying implementation allows a logical unit to be mapped (in other words, the maximum

number of ProtocolControllerForUnit associations that can be associated to the logical unit represented by the LogicalDevice subclass. The instrumentation sets this to 0 if it has no limit.

19.1.12 Deactivated Logical Units

Instrumentation may describe inaccessibility of a logical unit through a path using ProtocolControllerForUnit.AccessState. This property may be read, but not written by clients. Possible values are Active, Inactive, "Replication In Progress", and "Mapping Inconsistency".

Since default protocol controllers were not defined in SMI-S 1.0, a client could have created a configuration that does not comply with the SMI-S 1.1.0 semantics (which are intended to mimic SCSIs). Similarly, a non-compliant configuration could have been created using non-SMI-S interfaces. Instrumentation may set AccessState to "Mapping Inconsistency" to express these states. A client request to set a valid mapping configuration using ExposePaths should clear this state and reset AccessState to Active.

19.1.13 SCSIProtocolController Properties

Table 336 - SCSIProtocolController Property Description

Property	Description	Impact on ExposePaths (see 1)	Impact on HidePaths
SPCAllowsNoLUs	It is valid to have no LogicalDevices associated with an SPC	If true, LUNames, DeviceNumbers, and DeviceAccess may be null. If false, LUNames and DeviceAccesses shall be non-null. DeviceNumbers depend on ClientSelectableDeviceNumbers	If true, then all associated LogicalDevices may be specified in LUNames. If false and client specifies names of all associated LUs in LUNames, then see 2.
SPCAllowsNoTargets	It is valid to have no target ports associated with an SPC	If true, TargetPortIDs may be null. If false, TargetPortIDs shall be non-null.	If true, then all associated target ports may be specified in TargetPortIDs. If false, and client specifies names of all associated target ports in TargetPortIDs, then see 2.
SPCAllowsNoInitiators	It is valid to have no initiator port IDs associated with an SPC	If true, InitiatorPortIDs may be null. If false, InitiatorPortIDs shall be non-null.	If true, then all associated initiator port IDs may be specified in InitiatorPortIDs. If false, and client specifies names of all associated initiator port IDs in InitiatorPortIDs, then see 2.
1. This only applies to the "Create a new view" use case for ExposePaths. 2. The result of this HidePaths request would be an invalid partial path state. Therefore, the instrumentation shall delete the SPC and all its associations.			

There are two clarifications to the property descriptions in Table 336. If the implementation supports partial path SPCs, the intrinsic DeleteInstance is used to delete an SPC with no full paths. If DeleteInstance is called to delete an SPC with full paths, the instrumentation shall return CIM Error with CIM_ERR_FAILED status code.

19.1.14 Initiator Setting Data

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). Being able to identify the OS-specific operating mode ("host mode") of an element (i.e., FCPort or SCSIProtocolController) is essential because there are variances in SCSI communications between different operating systems or even different

versions of the same operating system, and having the incorrect "host mode" will cause operations to have degraded performance or even fail. This information is modeled as `StorageClientSettingData`. `StorageClientSettingData.ClientTypes[]` is an array of OS names. This array property allows a single `StorageClientSettingData` instance to apply to multiple OS Types. The `StorageClientSettingData` instances shall be scoped to a particular `ComputerSystem` because a CIM server hosting multiple devices will need to distinguish the valid `StorageClientSettingData` instances for one array from another.

The instrumentation should 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 `StorageHardwareID`s 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 system element (such as a Port, a `SCSIProtocolController`, or a `StorageVolume`) to request that this element exhibit the setting-specific behavior. This is done by creating a new `ElementSettingData` association from the element to the `StorageClientSettingData` instance using the intrinsic `CreateInstance` method. If any `ElementSettingData` association between the element and a `StorageClientSettingData` instance already exists, it shall be deleted by the client before calling `CreateInstance`. Figure 84: "StorageClientSettingData Model" provides an example.

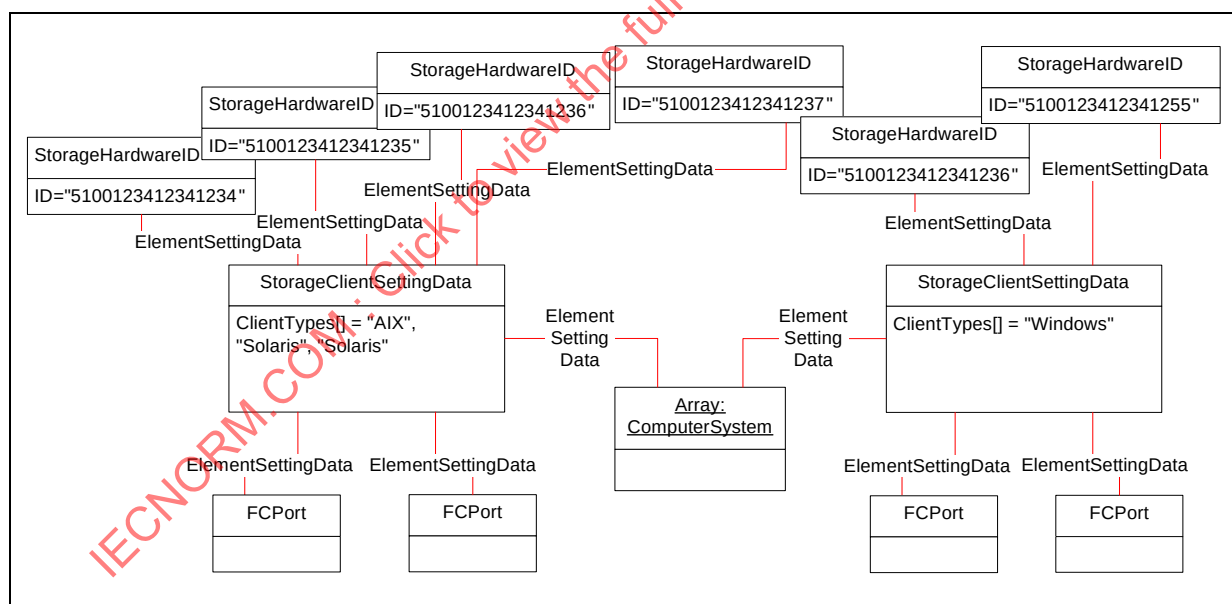


Figure 84 - StorageClientSettingData Model

Figure 85: "Entire Model" depicts the entire model.

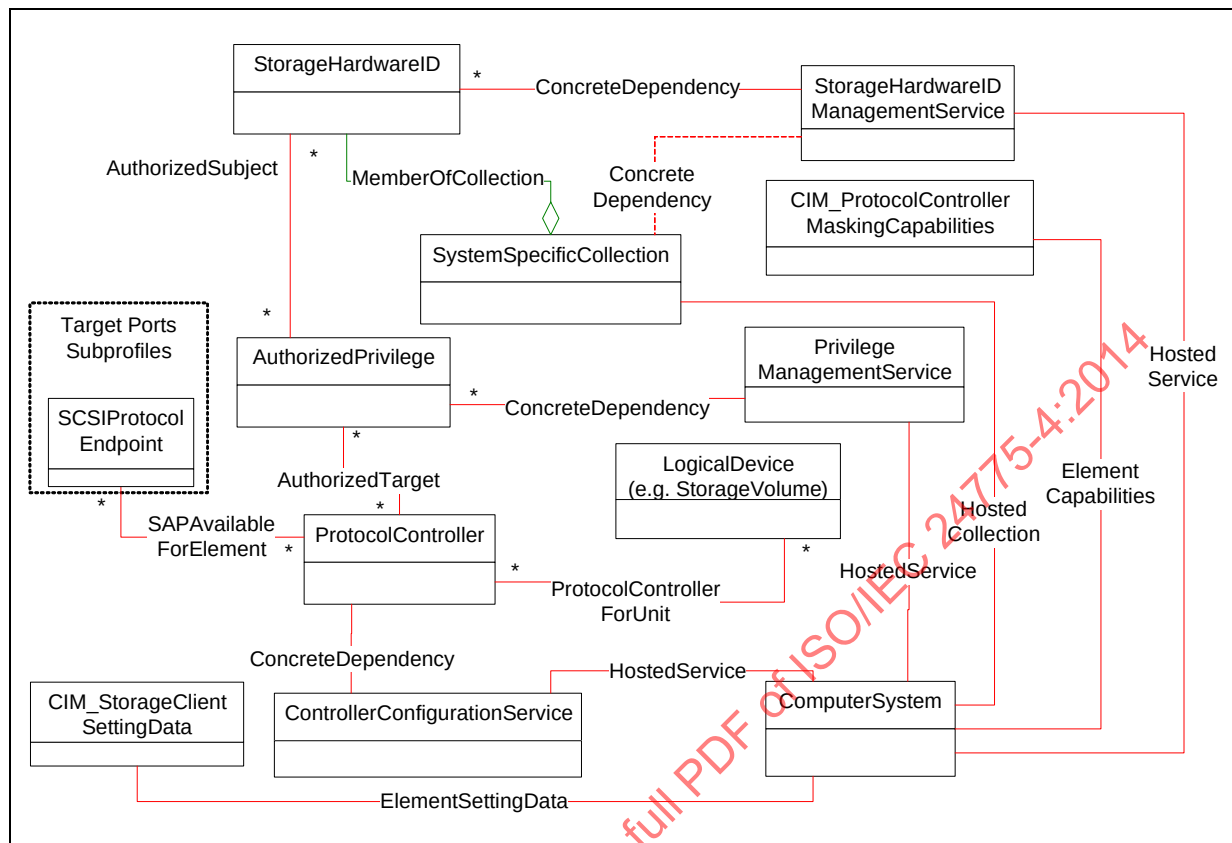


Figure 85 – Entire Model

19.1.15 Durable Names and Correlatable IDs of the Profile

The Masking and Mapping Subprofile uses the durable names/correlatable ID for logical devices as defined by the parent profile.

19.1.16 Instrumentation Requirements

If a PrivilegeManagementService is not present, then all access is assumed. If an PrivilegeManagementService is present, then access shall be specifically granted.

A LogicalDevice may have ProtocolControllerForUnit associations to multiple SCSIProtocolControllers - this models a device shared by different subject sets.

Clients may need to know the range of possible unit numbers supported by a storage system. The agent should set SCSIProtocolController.MaxUnitsControlled.

EXPERIMENTAL

The two CIM_ProtocolControllerMaskingCapabilities properties (SupportedSynchronousMethods and SupportAsynchronousMethods) describe the methods that are supported by the instrumentation. These enumerations indicate what operations will be executed as asynchronous jobs or synchronously. If an operation is included in both, then the underlying implementation is indicating that it may or may not create a job. If an operation is not included in either, then the instrumentation does not implement that method. If an instrumentation does not support all of the methods as defined by this subprofile, these properties can help a client determine if there is sufficient support to manage masking and mapping. Any instrumentation that does not support the required methods of this subprofile shall not be considered compliant even if these properties are supported.

19.1.17 Element Naming

The name of a ProtocolController, StorageHardwareID, GatewayPathID, or SystemSpecificCollection may be changed. The existence of the EnabledLogicalElementCapabilities instance associated to the element indicates that the element can be named. If ElementNameEditSupported is set to TRUE, then the ElementName of the associated element name may be modified. The ElementNameMask property provides the regular expression that expresses the limits of the name; see 19.8.20 for the class definition for EnabledLogicalElementCapabilities for details for this property.

However, this model does not indicate which element names can be used in the creation or modification of the element through the corresponding service methods if there are no elements of a given type. To do this, the instrumentation shall support a single EnabledLogicalElementCapabilities for each element type. There shall be a single mask for each storage element type as well. Each of these instances shall be associated to the appropriate service as described in Table 337.

Table 337 - Element to Service Mapping

Element	Service	Created by Service Method
GatewayPathID	StorageHardwareIDManagementService	CreateGatewayPathID
StorageHardwareID	StorageHardwareIDManagementService	CreateStorageHardwareID
StorageHardwareID	ControllerConfigurationService	ExposePaths
SystemSpecificCollection	StorageHardwareIDManagementService	CreateHardwareIDCollection
ProtocolController	ControllerConfigurationService	ExposePaths

The EnabledLogicalElementCapabilities associated to an element shall have the ElementName property set as per Table 338.

Table 338 - Element to Element Name Mapping

Element	EnabledLogicalElementCapabilities.ElementName
GatewayPathID	GatewayPathID Enabled Capabilities
StorageHardwareID	StorageHardware Enabled Capabilities
SystemSpecificCollection	SystemSpecificCollection Enabled Capabilities
ProtocolController	ProtocolController Enabled Capabilities

If the implementation supports the creation or modification of a given element type and the modification of the name of the storage element, then it shall produce an EnabledLogicalElementCapabilities instance for each of those elements.

If an storage element's name is modifiable through once of the aforementioned service methods, it shall also be modifiable through instance modification. However, a storage element's name may be modifiable through instance modification but storage element modification may not be allowed through these service methods.

EXPERIMENTAL

19.2 Health and Fault Management Considerations

None.

19.3 Cascading Considerations

None.

19.4 Supported Subprofiles, and Packages

None.

19.5 Methods of the Profile

19.5.1 ExposePaths

19.5.1.1 General

ExposePaths is used in place of the AssignAccess and AttachDevice methods used in 1.0. The problem with these methods was that they required the clients to perform multiple transactions to achieve a valid view. This forced providers to maintain invalid view states while waiting for the client to complete a sequence of transactions. This also created non-interoperability when the providers only supported transactions in a certain order, and when a second client looked at the model before a sequence of transactions was completed.

ExposePaths performs the mapping and masking operation in one method call. It exposes a list of SCSI logical units (such as RAID volumes or tape drives) to a list of initiators through a list of target ports, through one or more SCSIProtocolControllers (SPCs). Support for the 1.0 equivalent functionality is available by passing in an existing SCSIProtocolController.

There are two modes of operation, create and modify. If a NULL value is passed in for the SPC, then the instrumentation will create at least one SPC that satisfies the request. Depending upon the instrumentation capabilities, more than one SPC may be created. (e.g. if ProtocolControllerMaskingCapabilities.OneHardwareIDPerView is true and more than one initiatorID was passed in, then one SPC per initiatorID will be created). If an SPC is passed in, then the instrumentation attempts to add the new paths to the existing SPC. Depending upon the instrumentation capabilities, this may result in the creation of additional SPCs. The instrumentation shall return an error if honoring this request would violate SCSI semantics.

For creating an SPC, the parameters that need to be specified are dependent upon the SPCAllows* properties in ProtocolControllerMaskingCapabilities. If SPCAllowsNoLUs is false, the caller shall specify a list of LUNames. If it is true, the caller may specify a list of LUNames or may pass in null. If SPCAllowsNoTargets is false and PortsPerView is not 'All Ports share the same view' the caller shall specify a list of TargetPortIDs. If it is true, the caller may specify a list of TargetPortIDs or may pass in null. If SPCAllowsNoInitiators is false, the caller shall specify a list of InitiatorPortIDs. If it is true, the caller may specify a list of InitiatorPortIDs or may pass in null. If LUNames is not null, the caller shall specify the DeviceAccess for each logical unit. If the provider's ProtocolControllerMaskingCapabilities.ClientSelectableDeviceNumbers property is TRUE then the client shall either provide a list of device numbers (LUNs) to use for the paths to be created or pass in NULL. If it is false, the client shall pass in NULL for this parameter.

The LUNames, DeviceNumbers, and DeviceAccesses parameters are mutually indexed arrays - any element in DeviceNumbers or DeviceAccesses will set a property relative to the LogicalDevice instance named in the corresponding element of LUNames. LUNames and DeviceAccesses shall have the same number of elements. DeviceNumbers shall be null (asking the instrumentation to assign numbers) or have the same number of elements as LUNames. If these conditions are not met, the instrumentation shall return a 'Invalid Parameter' status.

For modifying an SPC, there are three specific use cases identified. The instrumentation shall support these use cases. Other permutations are allowed, but are vendor-specific. The use cases are: Add LUs to a view, Add initiator IDs to a view, and Add target port IDs to a view.

Add LUs to a view requires that the LUNames parameter not be null and that the InitiatorIDs and TargetPortIDs parameters be null. DeviceNumbers may be null if ClientSelectableDeviceNumbers is false. DeviceAccess shall be specified.

Add initiator IDs to a view requires that the LUNames parameter be null, that the InitiatorIDs not be null, and that the TargetPortIDs parameters be null. DeviceNumbers and DeviceAccess shall be null.

Add target port IDs to a view requires that the LUNames and InitiatorPortIDs parameters be null, and only possible if PortsPerView is 'Multiple Ports Per View'. DeviceNumbers and DeviceAccess shall also be null.

If a client calls ExposePaths specifying logical units already associated to the SPC and specifies different DeviceNumber or DeviceAccess values, the instrumentation shall change these properties in the appropriate ProtocolControllerForUnit instance(s).

When calling ExposePaths where an entry (e.g., LogicalDevice) does not exist, then ExposePaths shall fail and report an error.

There are four valid use cases for ExposePaths - create plus the three modify use cases above. These four use cases and the requirements for parameters are summarized in Table 339.

Table 339 - ExposePath Use Cases

parameters/ use cases	LUNames	InitiatorPortIDs	TargetPortIDs	DeviceNumbers	DeviceAccesses	Protocol- Controllers (on input)
Create a new view	See 1)	See 1)	See 1) See 2)	See 3)	Mandatory, see 4)	NULL
Add LUs to a view	Mandatory	NULL	NULL	See 3)	Mandatory, see 4)	contains a single SPC ref
Add initiator IDs to a view (see 5)	NULL	Mandatory	NULL	NULL	NULL	contains a single SPC ref
Add target port IDs to a view (see 6)	NULL	NULL	Mandatory	NULL	NULL	contains a single SPC ref
Vendor-specific	As long as all the previous use cases are implemented, the instrumentation may support other vendor-specific combinations of parameters.					
1. Dependent on values of new SPCAllowsNo* capability properties described below. 2. If PortsPerView is "All ports share same view", TargetPortIDs parameter shall be null. 3. If ClientSelectableDeviceNumbers is true, shall either be null or have same number of elements as LUNames. If ClientSelectableDeviceNumbers is false, shall be null. 4. They shall have same number of elements as LUNames. 5. Only valid if OneHardwareIDPerView is false. 6. Only valid if PortsPerView is "Multiple Ports per View".						

The relevant rules of SCSI semantics are:

- an SPC shall not be exposed through a particular host/target port pair that is in use by another SPC. (In other words, an SPC and its associated logical units and ports together correspond to the logical unit inventory provided by SCSI REPORT LUNS and INQUIRY commands).

- each LogicalDevice associated to an SPC shall have a unique ProtocolControllerForUnit DeviceNumber (logical unit number).

The instrumentation shall report an error if the client request would violate one of these rules.

If the instrumentation provides PrivilegeManagementService, the results of setting DeviceAccesses shall be synchronized with PrivilegeManagementService as described in the ProtocolControllerForUnit DeviceAccess description (see 18.8.27 "CIM_ProtocolControllerForUnit").

19.5.1.2 Uint32 ExposePaths

OUT CIM_ConcreteJob REF Job

Reference to the job (may be null if no job started)

IN string LUNames[]

An array of IDs of logical unit instances. The LU instances need to already exist. The members of this array shall match the Name property of LogicalDevice instances that represent SCSI logical units. See Table 339, "ExposePath Use Cases" for situations where this parameter may be null.

IN string InitiatorPortIDs[]

IDs of initiator ports. If existing StorageHardwareID instances exist, they shall be used. If no StorageHardwareID instance matches, then one is implicitly created. See Table 339, "ExposePath Use Cases" for situations where this parameter may be null.

IN string TargetPortIDs[]

IDs of target ports. See Table 339, "ExposePath Use Cases" for situations where this parameter may be null.

IN string DeviceNumber[]

A list of logical unit numbers to assign to the corresponding logical unit in the LUNames parameter. (Within the context of the elements specified in the other parameters.) If the LUNames parameter is null, then this parameter shall be null. Otherwise, if this parameter is null, all LU numbers are assigned by the hardware or instrumentation. This shall be formatted as unseparated uppercase hexadecimal digits, with no leading "0x".

IN uint16 DeviceAccess[]

A list of permissions to assign to the corresponding logical unit in the LUNames parameter. This specifies the permission to assign within the context of the elements specified in the other parameters. Setting this to 'No Access' assigns the DeviceNumber for the associated initiators, but does not grant read or write access. If the LUNames parameter is not null then this parameter shall be specified.

IN/OUT CIM_SCSIProtocolController REF ProtocolControllers[]

An array of references to SCSIProtocolControllers (SPCs). On input, this can be null, or contain exactly one element; if null on input, the instrumentation will create one or more new SPC instances.

On output, this will be either null (if a job was created) or the set of SPCs affected (those created or modified). Or those having some part of the 'view' modified, e.g. such as association being created or an AuthorizedPrivilege being created.) If a job was started, references to the SPCs affected will be found by following the AffectedJobElement association from the job.

19.5.2 HidePaths

19.5.2.1 General

HidePaths is used in place of the HideAccess and DetachDevice methods used in SMI-S 1.0. The problem with these methods is the same as AssignAccess and AttachDevice, in that they required the clients to perform multiple transactions to achieve a valid view. This forced providers to maintain invalid view states while waiting for the client to complete a sequence of transactions. This also created non-interoperability when the providers only supported transactions in a certain order, and when a second client looked at the model before a sequence of transactions was completed.

HidePaths is the inverse of ExposePaths. It hides a list of SCSI logical units (such as RAID volumes or tape drives) from a list of initiators through a list of target ports, through one or more SCSIProtocolControllers (SPCs). Support the 1.0 equivalent functionality is available by passing in an existing SCSIProtocolController.

When hiding logical units, there are three specific use cases identified. The instrumentation shall support these use cases. Other permutations are allowed, but are vendor-specific. The use cases are: Remove LUs from a view, Remove initiator IDs from a view, and Remove target port IDs from a view.

Remove LUs from a view requires that the LUNames parameter not be null and that the InitiatorIDs and TargetPortIDs parameters be null.

Remove initiator IDs from a view requires that the LUNames parameter be null, that the InitiatorIDs not be null, and that the TargetPortIDs parameters be null.

Remove target port IDs from a view requires that the LUNames and InitiatorPortIDs parameters be null.

The disposition of the SPC when the last logical unit, initiator ID, or target port ID is removed depends upon the ProtocolControllerMaskingCapabilites SPCAllowsNo* properties. If SPCAllowsNoLUs is false, then the SPC is automatically deleted when the last logical unit is removed. If SPCAllowsNoTargets is false, then the SPC is automatically deleted when the last target port ID is removed. If SPCAllowsNoInitiators is false, then the SPC is automatically deleted when the last initiator port ID is removed. In all other cases, the SPC needs to be explicitly deleted via the DeleteInstance intrinsic function or via the DeleteProtocolController method. The use cases for HidePaths() are summarized in Table 340.

Table 340 - HidePaths Use Cases

Parameters/use cases	LUNames	InitiatorPortIDs	TargetPortIDs	ProtocolController (on input) see 1
Remove LUs from a view	Mandatory	NULL	NULL	contains a single SPC ref
Remove initiator IDs from a view	NULL	Mandatory	NULL	contains a single SPC ref
Remove target ports from a view (see 2)	NULL	NULL	Mandatory	contains a single SPC ref
Hide full paths from a view	Mandatory	Mandatory	Mandatory	contains a single SPC ref
Vendor-specific	As long as all the previous use cases are implemented, the instrumentation may support other vendor-specific combinations of parameters.			
1. On output, the provider returns a list of refs to SPCs that have been affected (those created or modified or those having some part of the 'view' modified, e.g. such as association being created or deleted an AuthorizedPrivilege being created or deleted). Will be NULL if the SPC is automatically deleted as a result of one or more of the SPCAllowsNoLUs, SPCAllowsNoTargets, or SPCAllowsNoInitiators conditions being met as a result of the HidePaths operation. 2. Only valid if PortsPerView is "Multiple Ports per View".				

When calling HidePaths where the Port, SPC, StorageHardwareID, or StorageVolume exist, but the association(s) that are being modified don't exist (e.g. calling HidePaths for a volume that is not currently exposed), then HidePaths may return success. The rationale for returning success is the net result of the operation is the same whether or not the association exists, so it is not necessarily considered an error.

However, when calling HidePaths where an entry (e.g. Port) does not exist, then HidePaths shall return an error. The difference between this and the above case is that the above has just a connection between instances missing, while this case has an actual instance missing. The net result of the HidePaths operation would be different because HidePaths does not delete the instance (with the exception of the AuthorizedPrivilege), just the association between instances.

19.5.2.2 uint32 HidePaths

OUT CIM_ConcreteJob REF Job

Reference to the job (may be null if no job started)

IN string LUNames[]

An array of IDs of logical unit instances. The LU instances need to already exist. See Table 340, "HidePaths Use Cases" for situations where this parameter may be null.

IN string InitiatorPortIDs[]

IDs of initiator ports. See Table 340, "HidePaths Use Cases" for situations where this parameter may be null.

IN string TargetPortIDs[]

IDs of target ports. See Table 340, "HidePaths Use Cases" for situations where this parameter may be null.

IN/OUT CIM_SCSIProtocolController REF ProtocolControllers[]

An array of references to SCSIProtocolControllers (SPCs). On input, this can be null, or contain exactly one element. The instrumentation will attempt to remove associations (LUNames, InitiatorPortIDs, or TargetPortIDs) from this SPC. Depending upon the specific implementation, the instrumentation may need to create new SPCs with a subset of the remaining associations.

On output, this will be either null (if a job was created or if the SPC was automatically removed per the SPCAllowsNo* rules) or the set of SPCs affected (those created or modified). If a job was started, references to the SPCs affected will be found by following the AffectedJobElement association from the job.

19.5.3 ExposeDefaultLUs

19.5.3.1 General

ExposeDefaultLUs is similar to ExposePaths, except ExposeDefaultLUs works with 'default view' SPCs. The 'default view' SPC exposes logical units to all initiators. This SPC is identified by an association to a StorageHardwareID with Name property set to the empty string. ExposeDefaultLUs exposes a list of SCSI logical units (such as RAID volumes or tape drives) through a 'default view' SCSIProtocolController (SPC) through a list of target ports.

As with ExposePaths, there are two modes of operation, create and modify. If a NULL value is passed in for the SPC, then the instrumentation will attempt to create a new default view. If PortsPerView is 'All Ports share the same view', then there is at most one default view SPC. If PortsPerView is not 'All Ports share the same view', then there may be multiple default view SPCs as long as different ports are associated with each. If an SPC is passed in, then the instrumentation adds the new paths to the existing SPC. The instrumentation may return an error if honoring this request would violate SCSI semantics.

For creating a default view SPC, the parameters that need to be specified are dependent upon the SPCAllows* properties in ProtocolControllerMaskingCapabilities. If SPCAllowsNoLUs is false, the caller shall specify a list of LUNames. If it is true, the caller may specify a list of LUNames or may pass in null. If SPCAllowsNoTargets is false, the caller shall specify a list of TargetPortIDs. If it is true, the caller may specify a list of TargetPortIDs or may pass in null. If LUNames is not null, the caller shall specify the DeviceAccess for each logical unit. If the provider's ProtocolControllerMaskingCapabilities ClientSelectableDeviceNumbers property is TRUE then the client shall either provide a list of device numbers (LUNs) to use for the paths to be created or pass in NULL. If it is false, the client shall pass in NULL for this parameter.

The LUNames, DeviceNumbers, and DeviceAccesses parameters are mutually indexed arrays - any element in DeviceNumbers or DeviceAccesses will set a property relative to the LogicalDevice instance named in the corresponding element of LUNames. LUNames and

DeviceAccesses shall have the same number of elements. DeviceNumbers shall be null (asking the instrumentation to assign numbers) or have the same number of elements as LUNames. If these conditions are not met, the instrumentation shall return a 'Invalid Parameter' status.

For modifying an SPC, there are two specific use cases identified. The instrumentation shall support one and the other is required depending on how a property is set. Other permutations are allowed, but are vendor-specific.

The required use case is - Add LUs to a default view. Add LUs to a default view requires that the LUNames parameter not be null and that the TargetPortIDs parameters be null. DeviceNumbers may be null if ClientSelectableDeviceNumbers is false. DeviceAccess shall be specified.

Add target port IDs to a default view is only valid if PortsPerView is set to 'Multiple Ports per View'. It requires that the LUNames, DeviceNumbers, and DeviceAccess shall also be null. The use cases for ExposeDefaultLUs() are summarized in Table 341.

Table 341 - Use Cases for ExposeDefaultLUs

Parameters / use cases	LUNames	TargetPortIDs	DeviceNumbers	DeviceAccesses	ProtocolControllers (on input)
Create a new default view (see 1)	See 2)	See 2)	See 3)	Mandatory, see 4)	Shall be null
Add LUs to a view	Mandatory	Shall be null	See 3)	Mandatory, see 4)	Shall contain a single SPC ref
Add target port IDs to a view (see 5)	Shall be null	Mandatory	Shall be null	Shall be null	Shall contain a single SPC ref
Vendor-Specific	As long as all the previous use cases are implemented, the instrumentation may support other vendor-specific combinations of parameters.				
1. Only valid if PortsPerView is not "All Ports share the same View" 2. Dependent on values of SPCAllows* capability properties described above 3. If ClientSelectableDeviceNumbers is true, shall either be null or have same number of elements as LUNames. If ClientSelectableDeviceNumbers is false, shall be null. 4. Shall have same number of elements as LUNames 5. Only valid if PortsPerView is "Multiple Ports per View"					

The relevant rules of SCSI semantics are:

- an SPC shall be exposed through a particular host/target port pair that is in use by another SPC. (In other words, an SPC and its associated logical units and ports together correspond to the logical unit inventory provided by SCSI REPORT LUNS and INQUIRY commands)
- each LogicalDevice associated to an SPC shall have a unique ProtocolControllerForUnit DeviceNumber (logical unit number)

The instrumentation shall report an error if the client request would violate one of these rules.

If the instrumentation provides PrivilegeManagementService, the results of setting DeviceAccesses shall be synchronized with PrivilegeManagementService as described in the ProtocolControllerForUnit DeviceAccess description (18.8.27 "CIM_ProtocolControllerForUnit").

If the instrumentation supports ExposeDefaultLUs then it shall also support HideDefaultLUs.

19.5.3.2 uint32 ExposeDefaultLUs

OUT CIM_ConcreteJob REF Job

Reference to the job (may be null if no job started)

IN string LUNames[]

An array of IDs of logical unit instances. The LU instances shall already exist. The members of this array shall match the Name property of LogicalDevice instances that represent SCSI logical units. See Table 341, "Use Cases for ExposeDefaultLUs" for situations where this parameter may be null.

IN string TargetPortIDs[]

IDs of target ports. See Table 341, "Use Cases for ExposeDefaultLUs" for situations where this parameter may be null.

IN string DeviceNumber[]

A list of logical unit numbers to assign to the corresponding logical unit in the LUNames parameter. (Within the context of the elements specified in the other parameters.) If the LUNames parameter is null, then this parameter shall be null. Otherwise, if this parameter is null, all LU numbers are assigned by the hardware or instrumentation. Each element shall be formatted as unseparated uppercase hexadecimal digits, with no leading "0x".

IN uint16 DeviceAccess[]

A list of permissions to assign to the corresponding logical unit in the LUNames parameter. This specifies the permission to assign within the context of the elements specified in the other parameters. Setting this to 'No Access' assigns the DeviceNumber for the associated initiators, but does not grant read or write access. If the LUNames parameter is not null then this parameter shall be specified.

IN/OUT CIM_SCSIProtocolController REF ProtocolControllers[]

An array of references to SCSIProtocolControllers (SPCs). On input, this can be null, or contain exactly one element. There may be multiple references on output. If null on input, the instrumentation will create one or more new SPC instances.

On output, this will be either null (if a job was created) or the set of SPCs affected (those created or modified). If a job was started, references to the SPCs affected will be found by following the AffectedJobElement association from the job.

19.5.4 HideDefaultLUs

19.5.4.1 General

HideDefaultLUs is similar to HidePaths, except HideDefaultLUs works with 'default view' SPCs. The 'default view' SPC exposes logical units to all initiators. This SPC is identified by an association to a StorageHardwareID with Name property set to the empty string. HideDefaultLUs hides a list of SCSI logical units (such as RAID volumes or tape drives) through a 'default view' SCSIProtocolController (SPC) through a list of target ports.

HideDefaultLUs is the inverse of ExposeDefaultLUs. It hides a list of SCSI logical units (such as RAID volumes or tape drives) from a list of initiators through a list of target ports, through one or more SCSIProtocolControllers (SPCs).

When hiding logical units, there are two specific use cases identified. The use cases are: Remove LUs from a default view and Remove target port IDs from a default view. Remove LUs from a default view requires that the LUNames parameter not be null and that the TargetPortIDs parameter be null. Remove target port IDs from a default view is required if PortsPerView is Multiple Ports per view. It requires that the LUNames parameter be null.

The instrumentation shall support the Remove LUs case and shall support the remove target port IDs if PortsPerView is set to 'Multiple Ports per View'. Other permutations are allowed, but are vendor-specific.

If both LUNames and TargetIDs parameters are non-null and ProtocolControllerMaskingCapabilities.MaximumMapCount is 0, then the instrumentation shall create new SPCs and change associations as necessary to meet the client request and maintain the relevant rules of SCSI in the ExposeDefaultLUs description. If both LUNames and TargetIDs parameters are non-null and ProtocolControllerMaskingCapabilities.MaximumMapCount is greater than 0, then any client that cannot be honored by changing associations to the specified SPC shall receive a 'Maximum Map Count Error' response. The use cases for HideDefaultLUs are summarized in Table 342.

Table 342 - Use Cases for HideDefaultLUs

parameters/ use cases	LUNames	TargetPortIDs	ProtocolController (on input)
Remove LUs from a default view	Mandatory	Shall be null	Mandatory
Remove target ports from a view (see 1)	Shall be null	Mandatory	Mandatory
Vendor-specific	As long as all the previous usecases are implemented, the instrumentation may support other vendor-specific combinations of parameters.		
1. Only valid if PortsPerView is "Multiple Ports per View".			

The disposition of the SPC when the last logical unit or target port ID is removed depends upon the ProtocolControllerMaskingCapabilities.SPCAllows* properties. If SPCAllowsNoLUs is false, then the SPC is automatically deleted when the last logical unit is removed. If SPCAllowsNoTargets is false, then the SPC is automatically deleted when the last target port ID is removed. In all other cases, the SPC shall be explicitly deleted via the DeleteInstance intrinsic function.

If the instrumentation supports HideDefaultLUs then it shall also support ExposeDefaultLUs.

19.5.4.2 uint32 HideDefaultLUs

OUT CIM_ConcreteJob REF Job

Reference to the job (may be null if no job started)

IN string LUNames[]

An array of IDs of logical unit instances. The LU instances shall already exist. See Table 342, "Use Cases for HideDefaultLUs" for situations where this parameter may be null.

IN string TargetPortIDs[]

IDs of target ports. See Table 342, "Use Cases for HideDefaultLUs" for situations where this parameter may be null.

IN/OUT CIM_SCSIProtocolController REF ProtocolControllers[]

An array of references to SCSIProtocolControllers (SPCs). On input, this shall contain exactly one element. The instrumentation will attempt to remove associations (LUNames or TargetPortIDs) from this SPC. Depending upon the specific implementation, the instrumentation may need to create new SPCs with a subset of the remaining associations.

On output, this will be either null (if a job was created or if the SPC was automatically removed per the SPCAllowsNo* rules) or the set of SPCs affected (those created or modified). If a job

was started, references to the SPCs affected will be found by following the AffectedJobElement association from the job.

19.5.5 CreateStorageHardwareID

19.5.5.1 General

CreateStorageHardwareID creates a StorageHardwareID and the ConcreteDependency association between this service and the new StorageHardwareID.

19.5.5.2 UInt32 CreateStorageHardwareID

IN string ElementName

The ElementName of the new StorageHardwareID instance.

IN string StorageID

StorageID is the value used by the SecurityService to represent identity - in this case, a hardware worldwide unique name.

IN UInt16 IDType

The type of the StorageID property.

IN string OtherIDType

The type of the storage ID, when IDType is 'Other'.

IN CIM_StorageClientSettingData REF Setting

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.

IN CIM_StorageHardwareID REF HardwareID

REF to the new StorageHardwareID instance.

19.5.6 DeleteStorageHardwareID

19.5.6.1 General

DeleteStorageHardwareID deletes a StorageHardwareID and the ConcreteDependency association between the ID and the service. If the StorageHardwareID still has associations to AuthorizedPrivilege instances (and thus to ProtocolControllers), then this method shall return an error. The reason is that deleting it without deleting the associations would cause an invalid model. Deleting the Association and AuthorizedPrivilege and SPC would be a very unexpected side effect. The client shall call HidePaths() first to delete these associations.

19.5.6.2 UInt32 DeleteStorageHardwareID

IN CIM_StorageHardwareID REF HardwareID

REF to the StorageHardwareID to delete.

19.5.7 CreateHardwareIDCollection

19.5.7.1 General

Create a group of StorageHardwareIDs as a new instance of SystemSpecificCollection. This is useful to define a set of authorized subjects that can access volumes in a disk array. This method allows the client to make a request of a specific Service instance to create the collection and provide the appropriate class name. When these capabilities are standardized in CIM/WBEM, this method can be deprecated and intrinsic methods used. In addition to creating the collection, this method causes the creation of the HostedCollection association (to this

service's scoping system) and MemberOfCollection association to members of the IDs parameter.

19.5.7.2 uint32 CreateHardwareIDCollection

IN string ElementName

The ElementName to be assigned to the created collection.

IN string HardwareIDs[]

Array of strings containing representations of references to StorageHardwareID instances that will become members of the new collection.

OUT CIM_SystemSpecificCollection REF Collection

The new instance of SystemSpecificCollection that is created.

19.5.8 AddHardwareIDsToCollection

19.5.8.1 General

Create MemberOfCollection instances between the specified Collection and the StorageHardwareIDs. This method allows the client to make a request of a specific Service instance to create the associations. When these capabilities are standardized in CIM/WBEM, this method can be deprecated and intrinsic methods used.

19.5.8.2 uint32 AddHardwareIDsToCollection

IN string HardwareIDs[]

Array of strings containing representations of references to StorageHardwareID instances that will become members of the collection.

IN CIM_SystemSpecificCollection REF Collection

The Collection which groups the StorageHardwareIDs.

EXPERIMENTAL

19.5.9 DeleteProtocolController

19.5.9.1 General

DeleteProtocolController deletes the ProtocolController and all associations connected directly to this ProtocolController. It shall also delete any AuthorizedPrivilege instances associated to this ProtocolController as otherwise they would be left dangling. Since this subprofile does not have the notion of child ProtocolControllers, the DeleteChildrenProtocolControllers parameter shall be false. If the DeleteLogicalUnits parameter is True, the provider also deletes LogicalDevice instances associated via ProtocolControllerForUnit to this ProtocolController. LogicalDevice instances shall only be deleted when they are not part of any other ProtocolControllerForUnit associations. Whether or not the volumes may be deleted shall be determined by the instrumentation's support for the ReturnToStoragePool method in Block Services.

19.5.9.2 Uint32 DeleteProtocolController

IN CIM_ProtocolController REF ProtocolController

ProtocolController to be deleted.

IN boolean DeleteChildrenProtocolControllers

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.

IN boolean DeleteUnits

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.

EXPERIMENTAL

19.6 Client Considerations and Recipes

19.6.1 Expose and Hide LUNs

```
// DESCRIPTION:
//
// Test the accuracy of the Masking and Mapping
//
// PRE-EXISTING CONDITIONS AND ASSUMPTIONS
//
// 1. A reference to a storage element, a Storage Volume or Logical Disk
//    is defined in the $StorageElement-> variable
//    This storage element must not already be masked to any initiator
// 2. The WWN of two different Initiator Ports to be masked to is defined
//    in the
//    #InitiatorWWN1 and #InitiatorWWN2 variables.
// 3. The value of
//
//
//    CIM_ProtocolControllerMaskingCapabilities.ClientSelectableDeviceNumbers
//    is stored in #ClientSelectableDeviceNumbers
// 4. If #ClientSelectableDeviceNumbers is TRUE, the device number to be
//    used
//    for mapping is defined in #DeviceNumber.
// 5. The value of
//
//    CIM_ProtocolControllerMaskingCapabilities.PortsPerView is
//    stored in #PortsPerView
// 6. If #PortsPerView != 4 (All ports share the same view), the target
//    port WWN
//    is contained in the #TargetPortWWN variable.
// 7. The ControllerConfigurationService has been found and the object
//    path
//    value is stored in $ControllerConfigService->
// 8. The value of
//
//    CIM_ProtocolControllerMaskingCapabilities.OneHardwareIDPerView is
//    stored in #OneHardwareIDPerView
//
// Determine if there is a job created by method
// and wait for the job to complete
// Input:
// #ReturnCode : The return code of the method
```

```

// $ConcreteJob-> :The output parameter that may have a ConcreteJob
// REF.
// This method will return control if the recipe was not exited because
// of error
sub void WaitForJob(#ReturnCode, $ConcreteJob->) {
    if (4096 == #ReturnCode) {
        if ($ConcreteJob-> != null) {
            /*Wait until the completion of the job using $ConcreteJob-> as
            a filter Verify that the OperationalStatus contains 2 ("OK"),
            or 17 ("Completed") */
            $JobInstance = GetInstance($ConcreteJob->,
                false, false, false, null)
            if ($JobInstance.JobState != 7) { // 7 - Completed
                <ERROR! Job failed! >
            }
        } else {
            <ERROR! Missing Job reference>
        }
    }
}

/ Step 1. Subscribe for indications on the Job
// Job success -- Status is '17' ("Completed") and '2' ("OK")
#Filter1 = "SELECT FROM CIM_InstModification
           WHERE SourceInstance ISA CIM_ConcreteJob
           AND ANY SourceInstance.OperationalStatus[*] = 17
           AND ANY SourceInstance.OperationalStatus[*] = 2 "
// Determine if the Indication already exists
// If it doesn't, create it

// Job failure -- Status is '17' ("Completed") and '6' ("Error")
#Filter2 = "SELECT * FROM CIM_InstModification
           WHERE SourceInstance ISA CIM_ConcreteJob
           AND ANY SourceInstance.OperationalStatus[*] = 17
           AND ANY SourceInstance.OperationalStatus[*] = 6 "
// Determine if the Indication already exists
// If it doesn't, create it

/ Step 2. Expose a new LUN to an initiator
$StorageElement = GetInstance($StorageElement->,
    false, false, false, {"Name"})
%InputArguments["LUNames"] = {$StorageElement.Name}
%InputArguments["InitiatorPortIDs"] = {#InitiatorWWN1}

if (#PortsPerView != 4) { // 4 = All ports share the same view
    %InputArguments["TargetPortIDs"] = {#TargetPortWWN}
}

if (#ClientSelectableDeviceNumbers == TRUE) {
    %InputArguments["DeviceNumbers"] = {#DeviceNumber}
    %InputArguments["DeviceAccesses"] = {2} // Read-Write
}

```

```
}
else {
    %InputArguments["DeviceNumbers"] = NULL
    %InputArguments["DeviceAccesses"] = NULL
}

#ReturnCode = InvokeMethod($ControllerConfigService->,
    "ExposePaths",
    %InputArguments,
    %OutputArguments)
// 0 is "Success" and 4096 is "Method Parameters Checked - Job Started"
if (#ReturnCode != 0 || #ReturnCode != 4096) {
    <ERROR! Method failure>
}

$MMJob-> = %OutputArguments["Job"]
if ($MMJob-> == null) {
    $CreatedOrModifiedSPCs->[] = %OutputArguments["ProtocolControllers"]
}
else {
    // Wait until job is finished
    &WaitForJob(#ReturnCode, $MMJob->)

    // Now get the SPCs
    $CreatedOrModifiedSPCs->[] = Associators(
        $MMJob->,
        "CIM_AffectedJobElement",
        "CIM_ProtocolController",
        "AffectingElement",
        "AffectedElement",
        false, false, null)
}

// Verify results
if ($CreatedOrModifiedSPCs->[].length == 0) {
    <ERROR! There must be one or more SPC created or modified>
}

#Found = false
for #i in $CreatedOrModifiedSPCs->[] {
    $CheckSPCForUnits[] = References($CreatedOrModifiedSPCs->[#i],
        "CIM_ProtocolControllerForUnit",
        "Antecedent",
        false, false, null)
    for #u in $CheckSPCForUnits[] {
        if (#ClientSelectableDeviceNumbers == TRUE) {
            if ($CheckSPCForUnits[#u].DeviceNumber != #DeviceNumber ||
                $CheckSPCForUnits[#u].DeviceAccess != 2) {
                // no match found try next one (if any)
                continue
            }
        }
    }
}
```



```

    }

    // Validate Initiator ID
    $CheckAuthTargets->[] =
        AssociatorNames($CheckSPCForUnits[#u].Antecedent,
            "CIM_AuthorizedTarget",
            "CIM_AuthorizedPrivilege",
            null, null)

    for #k in $CheckAuthTargets->[] {
        $StorageHWIDs[] = Associators($CheckAuthTargets->[#k],
            "CIM_AuthorizedSubject",
            "CIM_StorageHardwareID",
            null, null, false, false, null)
        for #j in $StorageHWIDs[] {
            if ($StorageHWIDs[#j].StorageID == #InitiatorWWN1) {
                #Found = true
                break
            }
        }
        if (#Found == true) {
            break
        }
    }
    // Validate StorageElement
    if (#Found == true) { // If we didn't find initiator then don't bother
        $CheckStorageElement =
            GetInstance($CheckSPCForUnits[#u].Dependent,
                false, false, false, null)
        if ($StorageElement.Name != $CheckStorageElement.Name) {
            <ERROR! Masked and Mapped Storage Element not found>
        }
    }
    }
    }
    if (#Found == false) {
        <ERROR! Created mapping and masking was not found>
    }
    // Note: since we created one SPC, there should only be one entry here
    $AllCreatedOrModifiedSPCs->[] = $CreatedOrModifiedSPCs->[]

    // Step 3. Expose a currently exposed LUN to a different initiator
    if (#OneHardwareIDPerView == FALSE) {
        %InputArguments["LUNNames"] = NULL
        %InputArguments["InitiatorPortIDs"] = {#InitiatorWWN2}
        %InputArguments["TargetPortIDs"] = NULL
        %InputArguments["DeviceAccesses"] = NULL
        // Note: ExposePaths on a modify operation takes an array containing
        // one and only one SPC, which is what we have here
    }

```

```
%InputArguments["ProtocolControllers"] = { $CreatedOrModifiedSPCs->[0]}

#ReturnCode = InvokeMethod($ControllerConfigService->,
                           "ExposePaths",
                           %InputArguments,
                           %OutputArguments)
// 0 is "Success" and 4096 is "Method Parameters Checked - Job Started"
if (#ReturnCode != 0 || #ReturnCode != 4096) {
    <ERROR! Method failure>
}

$MMJob-> = %OutputArguments["Job"]
if ($MMJob-> == null) {
    $CreatedOrModifiedSPCs->[] =
        %OutputArguments["ProtocolControllers"]
}
else {
    // Wait until job is finished
    &WaitForJob(#ReturnCode, $MMJob->)

    // Now get the SPCs
    $CreatedOrModifiedSPCs->[] = Associators($MMJob->,
        "CIM_AffectedJobElement",
        "CIM_ProtocolController",
        "AffectingElement",
        "AffectedElement",
        false, false, null)
}

// Verify results
if ($CreatedOrModifiedSPCs->[].length == 0) {
    <ERROR! There must be one or more SPC created or modified>
}

#Found = false
for #i in $CreatedOrModifiedSPCs->[] {
    $CheckSPCForUnits[] = References($CreatedOrModifiedSPCs->[#i],
        "CIM_ProtocolControllerForUnit",
        "Antecedent",
        false, false, null)
    for #u in $CheckSPCForUnits[] {
        // Validate Initiator ID
        $CheckAuthTargets->[] =
            AssociatorNames($CheckSPCForUnits[#u].Antecedent,
                "CIM_AuthorizedTarget",
                "CIM_AuthorizedPrivilege",
                null, null)
        for #k in $CheckAuthTargets->[] {
            $StorageHWIDs[] = Associators($CheckAuthTargets->[#k],
                "CIM_AuthorizedSubject",
```

```

        "CIM_StorageHardwareID",
        null, null, false, false, null)
    for #j in $StorageHWIDs[] {
        if ($StorageHWIDs[#j].StorageID == #InitiatorWWN2) {
            #Found = true
            break
        }
    }
    if (#Found == true) {
        break
    }
}
// Validate StorageElement
if (#Found == true) { // If we didn't find initiator then don't
    bother
    $CheckStorageElement =
        GetInstance($CheckSPCForUnits[#u].Dependent,
            false, false, false, null)
    if ($StorageElement.Name != $CheckStorageElement.Name) {
        <ERROR! Masked and Mapped Storage Element not found>
    }
}
}
}
if (#Found == false) {
    <ERROR! Created mapping and masking was not found>
}

$AllCreatedOrModifiedSPCs->[] = $AllCreatedOrModifiedSPCs->[] +
    $CreatedOrModifiedSPCs->[]
/* Current contents of $AllCreatedOrModifiedSPCs->[] array
plus any new, unique SPC REFS */
} // if #OneHardwareIDPerView == FALSE

/ Step 4. Hide the paths previously exposed
// Since we can only pass in one SPC to HidePaths, we need to loop
// through the SPCs and call HidePaths for each one
$ModifiedSPCs->[] = null

for #spc in $AllCreatedOrModifiedSPCs->[] {
    $StorageElement = GetInstance($StorageElement->,
        false, false, false, {"Name"})
    %InputArguments2["LUNames"] = {$StorageElement.Name}
    if (#OneHardwareIDPerView == FALSE) {
        %InputArguments2["InitiatorPortIDs"] =
            {#InitiatorWWN1, #InitiatorWWN2}
    }
    else {
        %InputArguments2["InitiatorPortIDs"] = {#InitiatorWWN1}
    }
}

```

```
if (#PortsPerView != 4) { // All ports share the same view
    %InputArguments["TargetPortIDs"] = {#TargetPortWWN}
}
%InputArguments2["ProtocolControllers"] = {$AllCreatedOrModifiedSPCs-
    >[#spc]}

#ReturnCode = InvokeMethod($ControllerConfigService->,
    "HidePaths",
    %InputArguments2, %OutputArguments2)
// 0 is "Success" and 4096 is "Method Parameters Checked - Job Started"
if(#ReturnCode != 0 || #ReturnCode != 4096) {
    <ERROR! Method failure>
}

// Save any SPCs returned for later validation
$MMJob-> = %OutputArguments["Job"]
if ($MMJob == null) {
    $ModifiedSPCs->[] = %OutputArguments["ProtocolControllers"]
}
else {
    // Wait until job is finished
    &WaitForJob(#ReturnCode, $MMJob->)

    // Now get the SPCs
    $CreatedOrModifiedSPCs->[] = Associators(
        $MMJob->,
        "CIM_AffectedJobElement",
        "CIM_ProtocolController",
        "AffectingElement",
        "AffectedElement",
        false,
        false,
        null)
    $ModifiedSPCs->[] = $ModifiedSPCs->[] + $CreatedOrModifiedSPCs->[]
    /* Current contents of $ModifiedSPCs->[] array
    plus any new, unique SPC REFs from $CreatedOrModifiedSPCs->[]
    this list may be null */
}
}

// Verify results
#Found = false
// See if the storage element is still associated to one of the SPCs
$CheckSPCs->[] = AssociatorNames($StorageElement->,
    "CIM_ProtocolControllerForUnit",
    "CIM_ProtocolController",
    // Assumes StorageElement LogicalDevice
    null, null)
for #x in $CheckSPCs->[] {
    for #i in $ModifiedSPCs->[] {
        if($CheckSPCs->[#x].DeviceID == $ModifiedSPCs->[#i].DeviceID) {
```

```

        #Found = true
        break
    }
}
if (#Found == true) {
    <ERROR! Element still mapped>
}
}

// See if the Initiator WWNs are still associated to one of the SPCs
for #i in $ModifiedSPCs->[] {
    $CheckAuthPrivilege->[] = AssociatorNames($ModifiedSPCs->[#i],
        "CIM_AuthorizedTarget",
        "CIM_AuthorizedPrivilege",
        null, null)

    for #k in $CheckAuthPrivilege->[] {
        $StorageHWIDs[] = Associators($CheckAuthPrivilege->[#k],
            "CIM_AuthorizedSubject",
            "CIM_StorageHardwareID",
            null, null, false, false, { "StorageID" })

        for #j in $StorageHWIDs[] {
            if($StorageHWIDs[#j].StorageID == #InitiatorWWN1 ||
                $StorageHWIDs[#j].StorageID == #InitiatorWWN2 ) {
                #Found = true
                break
            }
        }

        if(#Found == true) {
            break. // CheckAuthTargets loop
        }
    }
    if(#Found == true) {
        <ERROR! Element still masked>
    }
}
}

```

19.6.2 Set Host Mode for a Port

```

// DESCRIPTION:
//
// Associate a ElementSettingData to a Port
// In this use case, the client wishes to set the FCPort to a specific
// OS-type.
// 1. Find a StorageClientSettingData instance to uses by enumerating
//    all instances of StorageClientSettingData scoped to that
//    ComputerSystem
// 2. Identify the Port to use
// 3. Find the existing ElementSettingData association (if any),

```

```
// 4. Delete it via DeleteInstance
// 5. Create a new one from the Port to the
//   StorageClientSettingData via CreateInstance

// PRE-EXISTING CONDITIONS AND ASSUMPTIONS
//
// 1. The StorageClientSettingData instance to use has been identified
//   and its reference stored in $ClientSettingData->
// 2. A Port has been identified and the reference stored in
//   $Port->

// Step 1. Delete any existing ElementSettingData association

$ExistingAssocs[] = Associators(
    $Port->,
    "CIM_ElementSettingData",
    "CIM_StorageClientSettingData",
    "ManagedElement",
    "SettingData",
    false, false, null)
for #i in $ExistingAssocs[] {
    $ObjectPath-> = $ExistingAssocs[#i].getObjectPath()
    #Result = DeleteInstance($ObjectPath->)
}

// Step 2. Associate the Port to the new setting

$instance = newInstance("CIM_ElementSettingData")
$instance.ManagedElement = $Port->
$instance.SettingData = $ClientSettingData->

$CreatedInst-> = CreateInstance($instance)
```

19.6.3 Set Host Mode for a ProtocolController

```
// DESCRIPTION:
//
// Associate a ElementSettingData to a ProtocolController
// In this use case, the client wishes to set the ProtocolController
// to a specific OS-type.
// 1. Find a StorageClientSettingData instance to uses by enumerating
//   all instances of StorageClientSettingData scoped to that
//   ComputerSystem
// 2. Identify the ProtocolController to use
// 3. Find the existing ElementSettingData association (if any),
// 4. Delete it via DeleteInstance
// 5. Create a new one from the ProtocolController to the
//   StorageClientSettingData via CreateInstance

// PRE-EXISTING CONDITIONS AND ASSUMPTIONS
//
```

```

// 1. The StorageClientSettingData instance to use has been identified
//    and its reference stored in $ClientSettingData->
// 2. A ProtocolController has been identified and the reference stored
//    in
//    $SPC->

// Step 1. Delete any existing ElementSettingData association

$ExistingAssocs[] = Associators(
    $SPC->,
    "CIM_ElementSettingData",
    "CIM_StorageClientSettingData",
    "ManagedElement",
    "SettingData",
    false, false, null)
for #i in $ExistingAssocs[] {
    $ObjectPath-> = $ExistingAssocs[#i].getObjectPath()
    #Result = DeleteInstance($ObjectPath->)
}

// Step 2. Associate the ProtocolController to the new setting

$instance = newInstance("CIM_ElementSettingData")
$instance.ManagedElement = $SPC->
$instance.SettingData = $ClientSettingData->

$CreatedInst-> = CreateInstance($instance)

```

19.7 Registered Name and Version

Masking and Mapping version 1.4.0 (Component Profile).

19.8 CIM Elements

19.8.1 General

Table 343 describes the CIM elements for Masking and Mapping.

Table 343 - CIM Elements for Masking and Mapping

Element Name	Requirement	Description
19.8.2 CIM_AuthorizedPrivilege	Mandatory	
19.8.3 CIM_AuthorizedSubject	Mandatory	
19.8.4 CIM_AuthorizedTarget	Mandatory	
19.8.5 CIM_ConcreteDependency (Associates ControllerConfigurationService and ProtocolController)	Mandatory	
19.8.6 CIM_ConcreteDependency (Associates PrivilegeManagementService and AuthorizedPrivilege)	Mandatory	
19.8.7 CIM_ConcreteDependency (Associates StorageHardwareIDManagementService and StorageHardwareID)	Mandatory	

Table 343 - CIM Elements for Masking and Mapping (Continued)

Element Name	Requirement	Description
19.8.8 CIM_ConcreteDependency (Associates StorageHardwareIDManagementService and SystemSpecificCollection)	Conditional	Conditional requirement: Implementation support for collections of StorageHardwareIDs.
19.8.9 CIM_ControllerConfigurationService	Mandatory	
19.8.10 CIM_ElementCapabilities (EnabledLogicalElementCapabilities to ControllerConfigurationService)	Optional	Associates EnabledLogicalElementCapabilities with ControllerConfigurationService.
19.8.11 CIM_ElementCapabilities (EnabledLogicalElementCapabilities to ProtocolController)	Optional	Expressed the ability for the element to be named or have its state changed.
19.8.12 CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StorageHardwareID)	Optional	Associates EnabledLogicalElementCapabilities to StorageHardwareID.
19.8.13 CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StorageHardwareIDManagementService)	Optional	Associates EnabledLogicalElementCapabilities with StorageHardwareIDManagementService.
19.8.14 CIM_ElementCapabilities (EnabledLogicalElementCapabilities to SystemSpecificCollection)	Conditional	Conditional requirement: Implementation support for collections of StorageHardwareIDs. Associates EnabledLogicalElementCapabilities and SystemSpecificCollection.
19.8.15 CIM_ElementCapabilities (System to ProtocolControllerMaskingCapabilities)	Mandatory	
19.8.16 CIM_ElementSettingData (Associates ComputerSystem and StorageClientSettingData)	Mandatory	
19.8.17 CIM_ElementSettingData (Associates Port and StorageClientSettingData)	Optional	
19.8.18 CIM_ElementSettingData (Associates ProtocolController and StorageClientSettingData)	Optional	
19.8.19 CIM_ElementSettingData (Associates StorageHardwareID and StorageClientSettingData)	Optional	
19.8.20 CIM_EnabledLogicalElementCapabilities	Optional	This class is used to express the naming and possible requested state change possibilities for storage elements.
19.8.21 CIM_HostedCollection	Conditional	Conditional requirement: Implementation support for collections of StorageHardwareIDs.
19.8.22 CIM_HostedService (Associates ComputerSystem and ControllerConfigurationService)	Mandatory	
19.8.23 CIM_HostedService (Associates ComputerSystem and PrivilegeManagementService)	Mandatory	
19.8.24 CIM_HostedService (Associates ComputerSystem and StorageHardwareIDManagementService)	Mandatory	
19.8.25 CIM_MemberOfCollection	Conditional	Conditional requirement: Implementation support for collections of StorageHardwareIDs.
19.8.26 CIM_PrivilegeManagementService	Mandatory	
19.8.27 CIM_ProtocolController	Mandatory	
19.8.28 CIM_ProtocolControllerForUnit	Mandatory	

Table 343 - CIM Elements for Masking and Mapping (Continued)

Element Name	Requirement	Description
19.8.29 CIM_ProtocolControllerMaskingCapabilities	Mandatory	
19.8.30 CIM_SAPAvailableForElement	Mandatory	
19.8.31 CIM_StorageClientSettingData	Mandatory	
19.8.32 CIM_StorageHardwareID	Mandatory	
19.8.33 CIM_StorageHardwareIDManagementService	Mandatory	
19.8.34 CIM_SystemSpecificCollection	Conditional	Conditional requirement: Implementation support for collections of StorageHardwareIDs.
19.8.35 SNIA_ProtocolControllerMaskingCapabilities	Optional	An experimental subclass of CIM_ProtocolControllerMaskingCapabilities.
19.8.36 SNIA_StorageHardwareID	Optional	Experimental SNIA class adding SAS Address IDs.
19.8.37 SNIA_StorageHardwareIDManagementService	Optional	Experimental subclass with support for SAS StorageHardwareIDs.
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_ProtocolController	Mandatory	Creation of a ProtocolController.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_ProtocolController	Mandatory	Deletion of a ProtocolController.
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_ProtocolControllerForUnit	Mandatory	Creation of a ProtocolControllerForUnit association.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_ProtocolControllerForUnit	Mandatory	Deletion of a ProtocolControllerForUnit association.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_ProtocolControllerForUnit	Mandatory	Modification of a ProtocolControllerForUnit association (e.g. changing DeviceNumber).
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_AuthorizedSubject	Mandatory	Creation of an AuthorizedSubject association.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_AuthorizedSubject	Mandatory	Deletion of an AuthorizedSubject association.

19.8.2 CIM_AuthorizedPrivilege

Created By: Extrinsic: CIM_ControllerConfigurationService.ExposePaths,
CIM_ControllerConfigurationService.HidePaths

Modified By: Extrinsic: CIM_ControllerConfigurationService.ExposePaths,
CIM_ControllerConfigurationService.HidePaths

Deleted By: Extrinsic: CIM_ControllerConfigurationService.ExposePaths,
CIM_ControllerConfigurationService.HidePaths

Requirement: Mandatory.

Table 344 describes class CIM_AuthorizedPrivilege.

Table 344 - SMI Referenced Properties/Methods for CIM_AuthorizedPrivilege

Properties	Flags	Requirement	Description and Notes
InstanceID		Mandatory	Opaque and unique identifier.
ElementName		Optional	User friendly name.
PrivilegeGranted		Mandatory	Indicates if the privilege is granted or not.
Activities		Mandatory	For SMI-S, shall be 5,6 ('Read' and 'Write').

19.8.3 CIM_AuthorizedSubject

Created By: Extrinsic: CIM_ControllerConfigurationService.ExposePaths,
CIM_ControllerConfigurationService.HidePaths

Modified By: Extrinsic: CIM_ControllerConfigurationService.ExposePaths,
CIM_ControllerConfigurationService.HidePaths

Deleted By: Extrinsic: CIM_ControllerConfigurationService.ExposePaths,
CIM_ControllerConfigurationService.HidePaths

Requirement: Mandatory.

Table 345 describes class CIM_AuthorizedSubject.

Table 345 - SMI Referenced Properties/Methods for CIM_AuthorizedSubject

Properties	Flags	Requirement	Description and Notes
PrivilegedElement		Mandatory	The Subject for which Privileges are granted or denied.
Privilege		Mandatory	The Privilege either granted or denied to an Identity or group of Identities collected by a Role.

19.8.4 CIM_AuthorizedTarget

Created By: Extrinsic: CIM_ControllerConfigurationService.ExposePaths,
CIM_ControllerConfigurationService.HidePaths

Modified By: Extrinsic: CIM_ControllerConfigurationService.ExposePaths,
CIM_ControllerConfigurationService.HidePaths

Deleted By: Extrinsic: CIM_ControllerConfigurationService.ExposePaths,
CIM_ControllerConfigurationService.HidePaths

Requirement: Mandatory.

Table 346 describes class CIM_AuthorizedTarget.

Table 346 - SMI Referenced Properties/Methods for CIM_AuthorizedTarget

Properties	Flags	Requirement	Description and Notes
TargetElement		Mandatory	The target set of resources to which the Privilege applies.
Privilege		Mandatory	The Privilege affecting the target resource.

19.8.5 CIM_ConcreteDependency (Associates ControllerConfigurationService and ProtocolController)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 347 describes class CIM_ConcreteDependency (Associates ControllerConfigurationService and ProtocolController).

Table 347 - SMI Referenced Properties/Methods for CIM_ConcreteDependency (Associates ControllerConfigurationService and ProtocolController)

Properties	Flags	Requirement	Description and Notes
Dependent		Mandatory	
Antecedent		Mandatory	

19.8.6 CIM_ConcreteDependency (Associates PrivilegeManagementService and AuthorizedPrivilege)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 348 describes class CIM_ConcreteDependency (Associates PrivilegeManagementService and AuthorizedPrivilege).

Table 348 - SMI Referenced Properties/Methods for CIM_ConcreteDependency (Associates PrivilegeManagementService and AuthorizedPrivilege)

Properties	Flags	Requirement	Description and Notes
Antecedent		Mandatory	
Dependent		Mandatory	

19.8.7 CIM_ConcreteDependency (Associates StorageHardwareIDManagementService and StorageHardwareID)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 349 describes class CIM_ConcreteDependency (Associates StorageHardwareIDManagementService and StorageHardwareID).

Table 349 - SMI Referenced Properties/Methods for CIM_ConcreteDependency (Associates StorageHardwareIDManagementService and StorageHardwareID)

Properties	Flags	Requirement	Description and Notes
Antecedent		Mandatory	
Dependent		Mandatory	

19.8.8 CIM_ConcreteDependency (Associates StorageHardwareIDManagementService and SystemSpecificCollection)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Implementation support for collections of StorageHardwareIDs.

Table 350 describes class CIM_ConcreteDependency (Associates StorageHardwareIDManagementService and SystemSpecificCollection).

Table 350 - SMI Referenced Properties/Methods for CIM_ConcreteDependency (Associates StorageHardwareIDManagementService and SystemSpecificCollection)

Properties	Flags	Requirement	Description and Notes
Antecedent		Mandatory	
Dependent		Mandatory	

19.8.9 CIM_ControllerConfigurationService

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 351 describes class CIM_ControllerConfigurationService.

Table 351 - SMI Referenced Properties/Methods for CIM_ControllerConfigurationService

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	The scoping System CreationClassName.
SystemName		Mandatory	The scoping System Name.
CreationClassName		Mandatory	The name of the concrete subclass.
Name		Mandatory	Unique identifier for the Service.
ExposePaths()		Mandatory	
HidePaths()		Mandatory	
ExposeDefaultLUs()		Optional	
HideDefaultLUs()		Optional	
DeleteProtocolController()		Optional	

19.8.10 CIM_ElementCapabilities (EnabledLogicalElementCapabilities to ControllerConfigurationService)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 352 describes class CIM_ElementCapabilities (EnabledLogicalElementCapabilities to ControllerConfigurationService).

**Table 352 - SMI Referenced Properties/Methods for CIM_ElementCapabilities
(EnabledLogicalElementCapabilities to ControllerConfigurationService)**

Properties	Flags	Requirement	Description and Notes
Capabilities		Mandatory	The capabilities object associated with the element.
ManagedElement		Mandatory	The managed element.

19.8.11 CIM_ElementCapabilities (EnabledLogicalElementCapabilities to ProtocolController)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 353 describes class CIM_ElementCapabilities (EnabledLogicalElementCapabilities to ProtocolController).

**Table 353 - SMI Referenced Properties/Methods for CIM_ElementCapabilities
(EnabledLogicalElementCapabilities to ProtocolController)**

Properties	Flags	Requirement	Description and Notes
Capabilities		Mandatory	The capabilities object associated with the element.
ManagedElement		Mandatory	

19.8.12 CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StorageHardwareID)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 354 describes class CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StorageHardwareID).

**Table 354 - SMI Referenced Properties/Methods for CIM_ElementCapabilities
(EnabledLogicalElementCapabilities to StorageHardwareID)**

Properties	Flags	Requirement	Description and Notes
Capabilities		Mandatory	The capabilities object associated with the element.
ManagedElement		Mandatory	

19.8.13 CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StorageHardwareIDManagementService)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 355 describes class CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StorageHardwareIDManagementService).

**Table 355 - SMI Referenced Properties/Methods for CIM_ElementCapabilities
(EnabledLogicalElementCapabilities to StorageHardwareIDManagementService)**

Properties	Flags	Requirement	Description and Notes
Capabilities		Mandatory	The capabilities object associated with the element.
ManagedElement		Mandatory	The managed element.

19.8.14 CIM_ElementCapabilities (EnabledLogicalElementCapabilities to SystemSpecificCollection)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Implementation support for collections of StorageHardwareIDs.

Table 356 describes class CIM_ElementCapabilities (EnabledLogicalElementCapabilities to SystemSpecificCollection).

**Table 356 - SMI Referenced Properties/Methods for CIM_ElementCapabilities
(EnabledLogicalElementCapabilities to SystemSpecificCollection)**

Properties	Flags	Requirement	Description and Notes
Capabilities		Mandatory	The capabilities object associated with the element.
ManagedElement		Mandatory	

19.8.15 CIM_ElementCapabilities (System to ProtocolControllerMaskingCapabilities)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 357 describes class CIM_ElementCapabilities (System to ProtocolControllerMaskingCapabilities).

**Table 357 - SMI Referenced Properties/Methods for CIM_ElementCapabilities
(System to ProtocolControllerMaskingCapabilities)**

Properties	Flags	Requirement	Description and Notes
ManagedElement		Mandatory	
Capabilities		Mandatory	

19.8.16 CIM_ElementSettingData (Associates ComputerSystem and StorageClientSetting-Data)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 358 describes class CIM_ElementSettingData (Associates ComputerSystem and StorageClientSettingData).

**Table 358 - SMI Referenced Properties/Methods for CIM_ElementSettingData
(Associates ComputerSystem and StorageClientSettingData)**

Properties	Flags	Requirement	Description and Notes
ManagedElement		Mandatory	
SettingData		Mandatory	

19.8.17 CIM_ElementSettingData (Associates Port and StorageClientSettingData)

Created By: CreateInstance

Modified By: Static

Deleted By: DeleteInstance

Requirement: Optional.

Table 359 describes class CIM_ElementSettingData (Associates Port and StorageClientSettingData).

**Table 359 - SMI Referenced Properties/Methods for CIM_ElementSettingData
(Associates Port and StorageClientSettingData)**

Properties	Flags	Requirement	Description and Notes
ManagedElement		Mandatory	
SettingData		Mandatory	

19.8.18 CIM_ElementSettingData (Associates ProtocolController and StorageClientSettingData)

Created By: CreateInstance

Modified By: Static

Deleted By: DeleteInstance

Requirement: Optional.

Table 360 describes class CIM_ElementSettingData (Associates ProtocolController and StorageClientSettingData).

**Table 360 - SMI Referenced Properties/Methods for CIM_ElementSettingData
(Associates ProtocolController and StorageClientSettingData)**

Properties	Flags	Requirement	Description and Notes
ManagedElement		Mandatory	
SettingData		Mandatory	

19.8.19 CIM_ElementSettingData (Associates StorageHardwareID and StorageClientSettingData)

Created By: Extrinsic:

CIM_StorageHardwareIDManagementService.CreateStorageHardwareID

Modified By: Static

Deleted By: Extrinsic:
CIM_StorageHardwareIDManagementService.DeleteStorageHardwareID
Requirement: Optional.

Table 361 describes class CIM_ElementSettingData (Associates StorageHardwareID and StorageClientSettingData).

**Table 361 - SMI Referenced Properties/Methods for CIM_ElementSettingData
(Associates StorageHardwareID and StorageClientSettingData)**

Properties	Flags	Requirement	Description and Notes
ManagedElement		Mandatory	
SettingData		Mandatory	

19.8.20 CIM_EnabledLogicalElementCapabilities

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

Table 362 describes class CIM_EnabledLogicalElementCapabilities.

Table 362 - SMI Referenced Properties/Methods for CIM_EnabledLogicalElementCapabilities

Properties	Flags	Requirement	Description and Notes
ElementName		Mandatory	The moniker for the instance.
ElementNameEditSupported		Mandatory	Denotes whether an storage element can be named.
MaxElementNameLen		Mandatory	Specifies the maximum length in glyphs (letters) for the name. See MOF for details.
ElementNameMask		Mandatory	The regular expression that specifies the possible content and format for the element name. See MOF for details.
RequestedStatesSupported		Optional	Expresses the states to which this element may be changed using the RequestStateChange method. If this property, it may be assumed that the state may not be changed.

19.8.21 CIM_HostedCollection

Created By: Extrinsic:
CIM_StorageHardwareIDManagementService.CreateHardwareIDCollection
Modified By: Static
Deleted By: Static
Requirement: Implementation support for collections of StorageHardwareIDs.

Table 363 describes class CIM_HostedCollection.

Table 363 - SMI Referenced Properties/Methods for CIM_HostedCollection

Properties	Flags	Requirement	Description and Notes
Antecedent		Mandatory	
Dependent		Mandatory	

19.8.22 CIM_HostedService (Associates ComputerSystem and ControllerConfigurationService)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 364 describes class CIM_HostedService (Associates ComputerSystem and ControllerConfigurationService).

Table 364 - SMI Referenced Properties/Methods for CIM_HostedService (Associates ComputerSystem and ControllerConfigurationService)

Properties	Flags	Requirement	Description and Notes
Antecedent		Mandatory	
Dependent		Mandatory	

19.8.23 CIM_HostedService (Associates ComputerSystem and PrivilegeManagementService)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 365 describes class CIM_HostedService (Associates ComputerSystem and PrivilegeManagementService).

Table 365 - SMI Referenced Properties/Methods for CIM_HostedService (Associates ComputerSystem and PrivilegeManagementService)

Properties	Flags	Requirement	Description and Notes
Antecedent		Mandatory	
Dependent		Mandatory	

19.8.24 CIM_HostedService (Associates ComputerSystem and StorageHardwareIDManagementService)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 366 describes class CIM_HostedService (Associates ComputerSystem and StorageHardwareIDManagementService).

Table 366 - SMI Referenced Properties/Methods for CIM_HostedService (Associates ComputerSystem and StorageHardwareIDManagementService)

Properties	Flags	Requirement	Description and Notes
Antecedent		Mandatory	
Dependent		Mandatory	

19.8.25 CIM_MemberOfCollection

Created By: Extrinsic:

CIM_StorageHardwareIDManagementService.CreateHardwareIDCollection,
CIM_StorageHardwareIDManagementService.AddHardwareIDsToCollection

Modified By: Static

Deleted By: Static

Requirement: Implementation support for collections of StorageHardwareIDs.

Table 367 describes class CIM_MemberOfCollection.

Table 367 - SMI Referenced Properties/Methods for CIM_MemberOfCollection

Properties	Flags	Requirement	Description and Notes
Collection		Mandatory	
Member		Mandatory	

19.8.26 CIM_PrivilegeManagementService

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 368 describes class CIM_PrivilegeManagementService.

Table 368 - SMI Referenced Properties/Methods for CIM_PrivilegeManagementService

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	The scoping System CreationClassName.
CreationClassName		Mandatory	The name of the concrete subclass.
SystemName		Mandatory	The scoping System Name.
Name		Mandatory	Uniquely identifies the Service.
ElementName		Mandatory	User friendly name.
AssignAccess()		Mandatory	
RemoveAccess()		Mandatory	

19.8.27 CIM_ProtocolController

Created By: Extrinsic: CIM_ControllerConfigurationService.ExposePaths,
CIM_ControllerConfigurationService.HidePaths

Modified By: Extrinsic: CIM_ControllerConfigurationService.ExposePaths,
CIM_ControllerConfigurationService.HidePaths

Deleted By: Extrinsic: CIM_ControllerConfigurationService.ExposePaths,
CIM_ControllerConfigurationService.HidePaths

Requirement: Mandatory.

Table 369 describes class CIM_ProtocolController.

Table 369 - SMI Referenced Properties/Methods for CIM_ProtocolController

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	The scoping System CreationClassName.
CreationClassName		Mandatory	The name of the concrete subclass.
SystemName		Mandatory	The scoping System's Name.
DeviceID		Mandatory	Unique name for the ProtocolController.

19.8.28 CIM_ProtocolControllerForUnit

Created By: Extrinsic: CIM_ControllerConfigurationService.ExposePaths,
CIM_ControllerConfigurationService.HidePaths,
CIM_ControllerConfigurationService.ExposeDefaultLUs,
CIM_ControllerConfigurationService.HideDefaultLUs

Modified By: Extrinsic: CIM_ControllerConfigurationService.ExposePaths,
CIM_ControllerConfigurationService.HidePaths,
CIM_ControllerConfigurationService.ExposeDefaultLUs,
CIM_ControllerConfigurationService.HideDefaultLUs

Deleted By: Extrinsic: CIM_ControllerConfigurationService.ExposePaths,
CIM_ControllerConfigurationService.HidePaths,
CIM_ControllerConfigurationService.ExposeDefaultLUs,
CIM_ControllerConfigurationService.HideDefaultLUs

Requirement: Mandatory.

Table 370 describes class CIM_ProtocolControllerForUnit.

Table 370 - SMI Referenced Properties/Methods for CIM_ProtocolControllerForUnit

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

19.8.29 CIM_ProtocolControllerMaskingCapabilities

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 371 describes class CIM_ProtocolControllerMaskingCapabilities.

Table 371 - SMI Referenced Properties/Methods for CIM_ProtocolControllerMaskingCapabilities

Properties	Flags	Requirement	Description and Notes
InstanceID		Mandatory	Opaque and unique identifier.
ElementName		Mandatory	User-friendly name.
ValidHardwareIDTypes		Mandatory	A list of the valid values for StorageHardwareID.IDType.
PortsPerView		Mandatory	Indicates the way that ports per view (ProtocolController) are handled.
ClientSelectableDeviceNumbers		Mandatory	Indicates whether the client can specify the DeviceNumbers parameter when calling ControllerConfigurationService.ExposePaths().
OneHardwareIDPerView		Mandatory	Set to true if this storage system limits configurations to a single subject hardware ID per view.
PrivilegeDeniedSupported		Mandatory	Set to true if this storage system allows a client to create a Privilege instance with PrivilegeGranted set to FALSE.
UniqueUnitNumbersPerPort		Mandatory	Indicates whether different ProtocolControllers attached to a SCSIProtocolEndpoint can expose the same unit numbers (e.g. multiple LUN 0s) or if the numbers shall be unique.
ProtocolControllerSupportsCollections		Optional	Indicates the storage system supports SystemSpecificCollections of StorageHardwareIDs.
OtherValidHardwareIDTypes		Conditional	Conditional requirement: Properties required when ValidHardwareIDTypes includes 1 (Other). An array of strings describing types for valid StorageHardwareID.IDType. Used when the ValidHardwareIDTypes includes Other.
MaximumMapCount		Mandatory	The maximum number of ProtocolControllerForUnit associations that can be associated with a single LogicalDevice (for example, StorageVolume). Zero indicates there is no limit.
SPCAllowsNoLUs		Mandatory	Set to true if a client can create an SPC with no LogicalDevices.
SPCAllowsNoTargets		Mandatory	Set to true if a client can create an SPC with no target SCSIProtocolEndpoints.
SPCAllowsNoInitiators		Mandatory	Set to true if a client can create an SPC with no StorageHardwareIDs.
SPCSupportsDefaultViews		Mandatory	Set to true if the instrumentation supports default view SPCs that exposes logical units to all initiators.

19.8.30 CIM_SAPAvailableForElement

Created By: Extrinsic: CIM_ControllerConfigurationService.ExposePaths,
CIM_ControllerConfigurationService.HidePaths,
CIM_ControllerConfigurationService.ExposeDefaultLUs,
CIM_ControllerConfigurationService.HideDefaultLUs

Modified By: Extrinsic: CIM_ControllerConfigurationService.ExposePaths,
CIM_ControllerConfigurationService.HidePaths,
CIM_ControllerConfigurationService.ExposeDefaultLUs,
CIM_ControllerConfigurationService.HideDefaultLUs

Deleted By: Extrinsic: CIM_ControllerConfigurationService.ExposePaths,
CIM_ControllerConfigurationService.HidePaths,
CIM_ControllerConfigurationService.ExposeDefaultLUs,
CIM_ControllerConfigurationService.HideDefaultLUs

Requirement: Mandatory.

Table 372 describes class CIM_SAPAvailableForElement.

Table 372 - SMI Referenced Properties/Methods for CIM_SAPAvailableForElement

Properties	Flags	Requirement	Description and Notes
AvailableSAP		Mandatory	
ManagedElement		Mandatory	

19.8.31 CIM_StorageClientSettingData

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 373 describes class CIM_StorageClientSettingData.

Table 373 - SMI Referenced Properties/Methods for CIM_StorageClientSettingData

Properties	Flags	Requirement	Description and Notes
InstanceID		Mandatory	Opaque and unique identifier.
ElementName		Mandatory	A user-friendly name.
ClientTypes		Mandatory	Array of OS names.

19.8.32 CIM_StorageHardwareID

Created By: Extrinsic:

CIM_StorageHardwareIDManagementService.CreateStorageHardwareID,
CIM_ControllerConfigurationService.ExposePaths

Modified By: Static

Deleted By: Extrinsic:

CIM_StorageHardwareIDManagementService.DeleteStorageHardwareID

Requirement: Mandatory.

Table 374 describes class CIM_StorageHardwareID.

Table 374 - SMI Referenced Properties/Methods for CIM_StorageHardwareID

Properties	Flags	Requirement	Description and Notes
InstanceID		Mandatory	Opaque and unique identifier.
StorageID	N	Mandatory	The worldwide unique ID.
IDType		Mandatory	StorageID type. Values may be 1 2 3 4 5 (Other or PortWWN or NodeWWN or Hostname or iSCSI Name).

19.8.33 CIM_StorageHardwareIDManagementService

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 375 describes class CIM_StorageHardwareIDManagementService.

Table 375 - SMI Referenced Properties/Methods for CIM_StorageHardwareIDManagementService

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	The scoping System CreationClassName.
SystemName		Mandatory	The scoping System Name.
CreationClassName		Mandatory	The name of the concrete subclass.
Name		Mandatory	Uniquely identifies the Service.
CreateStorageHardwareID()		Mandatory	
DeleteStorageHardwareID()		Mandatory	
CreateHardwareIDCollection()		Optional	
AddHardwareIDsToCollection()		Optional	

19.8.34 CIM_SystemSpecificCollection

Created By: Extrinsic:

CIM_StorageHardwareIDManagementService.CreateHardwareIDCollection

Modified By: Static

Deleted By: Static

Requirement: Implementation support for collections of StorageHardwareIDs.

Table 376 describes class CIM_SystemSpecificCollection.

Table 376 - SMI Referenced Properties/Methods for CIM_SystemSpecificCollection

Properties	Flags	Requirement	Description and Notes
InstanceID		Mandatory	Opaque and unique identifier.
ElementName		Mandatory	A user-friendly name.

19.8.35 SNIA_ProtocolControllerMaskingCapabilities

An experimental subclass of CIM_ProtocolControllerMaskingCapabilities that adds properties asserting method support and support for SAS StorageHardwareIDs.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 377 describes class SNIA_ProtocolControllerMaskingCapabilities.

Table 377 - SMI Referenced Properties/Methods for SNIA_ProtocolControllerMaskingCapabilities

Properties	Flags	Requirement	Description and Notes
SupportedAsynchronousActions		Mandatory	Indicates which operations will result in a Job being created.
SupportedSynchronousActions		Mandatory	Indicates which operations will execute without a Job being created.

19.8.36 SNIA_StorageHardwareID

Created By: Extrinsic:

CIM_StorageHardwareIDManagementService.CreateStorageHardwareID,
CIM_ControllerConfigurationService.ExposePaths

Modified By: Static

Deleted By: Extrinsic:

CIM_StorageHardwareIDManagementService.DeleteStorageHardwareID

Requirement: Optional

Table 378 describes class SNIA_StorageHardwareID.

Table 378 - SMI Referenced Properties/Methods for SNIA_StorageHardwareID

Properties	Flags	Requirement	Description and Notes
InstanceID		Mandatory	Opaque and unique identifier.
StorageID	N	Mandatory	The worldwide unique ID.
IDType		Mandatory	StorageID type. Values may be 1 2 3 4 5 6 (Other or PortWWN or NodeWWN or Hostname or iSCSI Name or SAS Address).

19.8.37 SNIA_StorageHardwareIDManagementService

Experimental subclass with support for SAS StorageHardwareIDs.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 379 describes class SNIA_StorageHardwareIDManagementService.

Table 379 - SMI Referenced Properties/Methods for SNIA_StorageHardwareIDManagementService

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	The scoping System CreationClassName.
SystemName		Mandatory	The scoping System Name.
CreationClassName		Mandatory	The name of the concrete subclass.
Name		Mandatory	Uniquely identifies the Service.
CreateStorageHardwareID()		Mandatory	Experimental: may use SAS Address IDType.
DeleteStorageHardwareID()		Mandatory	
CreateHardwareIDCollection()		Optional	
AddHardwareIDsToCollection()		Optional	

STABLE

DEPRECATED

20 Pool Manipulation Capabilities, and Settings Subprofile

The functionality of the LUN Creation and Pool Manipulation Capabilities, and Settings Subprofiles has been subsumed by Clause 6 Block Services Package.

The Pool Manipulation Capabilities, and Settings Subprofile is defined in 7.3.3.10 of SMI-S 1.0.2.

DEPRECATED

IECNORM.COM : Click to view the full PDF of ISO/IEC 24775-4:2014

EXPERIMENTAL

21 Storage Server Asymmetry Profile

21.1 Description

21.1.1 Overview

High-availability storage servers using multiple redundant storage processors exhibit a range of interrelated behavior involving load-balancing, ports, and failover. This profile provides for management of these aspects.

Many such systems have the concept of a storage resource (either a RAID group or a storage volume) having an assignment to, or affinity for, one of the storage processors in a redundant set. This affinity may have one or more underlying architectural reasons for existing. Examples are both front-end (target) port connectivity with and between processors, cache processing, virtualization (RAID) processing, or connectivity partitioning of back end resources.

When the storage processor for which the storage resource has affinity fails, the resource is taken over by one of the other processors in the redundancy set.

When both storage processors are healthy, the ports on the storage processor for which the storage resource has affinity provide full bandwidth access to the resource. The ports on the "other" storage processors provide full, limited, or standby access, depending on implementation.

21.1.2 Relationship to Multiple Computer System Subprofile

This profile is a component profile (or subprofile) and extends the functionality of the Multiple Computer System Subprofile, which in turn references this profile as a supported profile. This profile requires the use of the Multiple Computer System Subprofile.

A separate profile was created for two purposes. Firstly, the functionality of Asymmetric Access is largely storage-related and since the MCS is a common profile, the asymmetry functions are specified separately. Secondly, although some asymmetric behavior may be modeled using provisions under the Multiple Computer System Profile regarding aggregating resources to the lowest level ComputerSystem that represents availability, many implementations aggregate all resources to the top-level ComputerSystem, even though these implementations exhibit asymmetric behavior. These resources include CIM_StorageVolumes, CIM_StoragePools, CIM_ProtocolControllers, CIM_ProtocolEndpoints, and the CIM_StorageConfiguration and CIM_ControllerConfiguration services. CIM_LogicalPorts are usually aggregated to the lower level systems that represent the storage processors.

Asymmetric behavior is modeled through constructs in this profile and is independent of SystemDevice and Hosting associations in a Multiple Computer System.

21.1.3 Relationship to Masking and Mapping Subprofile

The Masking and Mapping Subprofile provides the means to expose storage volumes to initiators through front-end ports. In systems with asymmetric behavior, Masking and Mapping alone does not provide for determining whether the action of the ExposePaths method will result in the creation of a path that is primary, secondary, or standby from a performance standpoint.

This profile does not formally extend Masking and Mapping but augments its functionality by providing the model constructs to support this determination by a client. It does this with model relationships directly between groups of front-end ports (which are represented by subclasses of CIM_ProtocolEndpoint) and groups of storage resources, independent of the implementation of Mapping and Masking "View" CIM_ProtocolControllers. This is necessary because some implementations may not generate "primary" and "standby" view/mappings for the ports on each storage processor but instead share common view controllers between storage

processors, making it impossible to use the “view” CIM_ProtocolController to group ports with volumes.

21.1.4 Relationship to T10

This subprofile supports the passive management of the functionality defined in the Target Port Group Access States clause in INCITS 513 (future ISO/IEC 14776-454, see Bibliography).

21.1.5 Behavior, Characteristics, and Capabilities

21.1.5.1 General

The behavioral use cases for redundant systems are used to derive asymmetry characteristics which, in turn, are used to distill capabilities for the profile that allow a client to interpret the asymmetric model objects.

21.1.5.2 Port Failover

21.1.5.2.1 General

The first differentiator to consider when trying to classify asymmetric behavior is target port failover behavior. Front-end ports on storage processors in a redundancy set exhibit either transparent or non-transparent behavior when the supporting storage processor fails

21.1.5.2.2 Transparent

In transparent failover, a storage processor can support multiple virtual ports, that is the ports that it normally has, and the functionality of ports from a failed storage processor in the same redundancy set. Stated another way, when a storage processor fails, its ports don't fail, they fail over to a healthy storage processor. This mode is called transparent because the host sees only a transient loss of access to the port. The port itself is still present after the failover.

21.1.5.2.3 Non-Transparent

In this type of architecture, the ports supported by a storage processor fail when the processor fails. Access to the storage volumes that were exposed through the failed ports is provided through ports on a surviving processor.

21.1.5.3 Port Asymmetry

Healthy storage servers have variant functionality with respect to access to volumes through ports on different storage processors. This may be related to the affinity of such volumes (or the pools to which they belong) to storage processors as described in 21.1.5.4 “Storage Resource Affinity”. In some systems, there is “full” bandwidth access to a volume through both ports on processor A and ports on processor B. This is actually symmetric access. In other cases, access to a volume is full bandwidth access through ports on the storage processor (“this”) for which the volumes have affinity and “reduced” bandwidth access through ports on the “other” processor. The third variation is that there is no access at all, other than inquiry type commands, through ports on the “other” processor, until the processor for which the volumes have affinity fails. This functionality is reflexive in that there is full access to volumes having affinity for the “other” processor through ports on that processor, while there is reduced access or no access to volumes that have affinities to “other” through ports on “this”.

21.1.5.4 Storage Resource Affinity

Storage resource affinity is the behavior that in many redundant servers, storage resources, either individual volumes or RAID groups (also called RAID sets or RAID ranks) and thus the volumes allocated from them, have an affinity for a given storage processor in a redundancy set. This affinity may stem from allocation of non-dual ported drives to a processor or assignment of these resources to a processor for cache or RAID processing architectural considerations. Managing this affinity is necessary on redundant systems as part of a static load balancing strategy. This is true even when the front-end ports exhibit symmetric access behavior, because assigning all resources to one storage processor may degrade the overall system throughput.

21.1.6 Model

21.1.6.1 Classes

21.1.6.1.1 General

This profile introduces five new classes. These include one capabilities class, two collections, and two associations, shown in Figure 86.

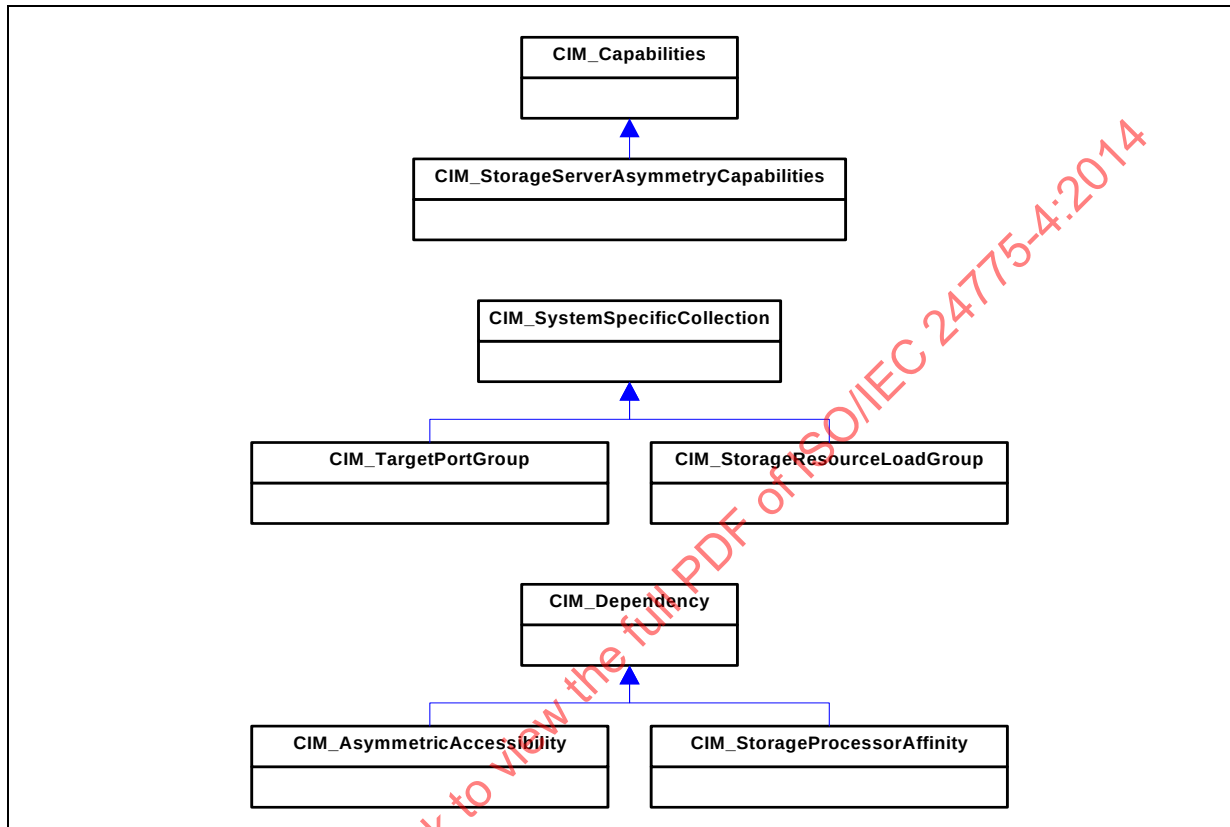


Figure 86 - Storage Asymmetry Class Hierarchy

21.1.6.1.2 Asymmetry Capabilities

This class contains properties that enable a client to determine the combination of asymmetry characteristics implemented by the subject storage system. More specifically, they guide the client algorithms in interpretation of the instances of the asymmetry classes and associations. The capabilities are detailed in 21.8 "CIM Elements".

21.1.6.1.3 TargetPortGroup

21.1.6.1.3.1 General

This subclass of CIM_SystemSpecificCollection aggregates the instances of CIM_ProtocolEndpoint or its subclasses that represent the ports on a storage processor (represented by CIM_ComputerSystem). The ports are aggregated because their relationship to the storage processors for failover and to the storage resources for accessibility are the same.

Whether ProtocolEndpoint is used directly or one of its subclasses is used depends on which Target Port component profile is implemented by the storage server.

Because CIM_TargetPortGroup is a CIM_SystemSpecificCollection there shall be an instance of CIM_HostedCollection from each instance of CIM_TargetPortGroup to the instance of CIM_ComputerSystem in the referencing Multiple Computer System Profile that represents the Top-Level System.

21.1.6.1.3.2 Multiple Hierarchical TargetPortGroups

Some Target Port profiles, such as the ISCS Target Port Profile, may have a hierarchy of ProtocolEndpoints. Each layer of ProtocolEndpoints in the hierarchy that can have affinity for a storage processor may be aggregated by a separate TargetPortGroup. This enables a client to determine which lower-level ProtocolEndpoints in the hierarchy may be used to create upper-level ProtocolEndpoints with the desired affinity. An example is the need to select TCPProtocolEndpoints with the same affinity for a storage processor when attempting to create an iSCSIProtocolEndpoint for that same processor.

21.1.6.1.4 StorageResourceLoadGroup

21.1.6.1.4.1 General

This subclass of CIM_SystemSpecificCollection aggregates either the storage volumes or storage pools that have the same affinity for a storage processor. What type of storage resource is aggregated depends on whether the pools have affinity or are common between processors and just the individual volumes have affinity. There is a capabilities property to specify this. There is one static instance of StorageResourceLoadGroup for each storage processor, with a single exception described in 21.1.6.1.4.2.

Because CIM_StorageResourceLoadGroup is a CIM_SystemSpecificCollection there shall be an instance of CIM_HostedCollection from each instance of CIM_StorageResourceLoadGroup to the instance of CIM_ComputerSystem in the referencing Multiple Computer System Profile that represents the Top-Level System.

21.1.6.1.4.2 Single Volume Accessibility Override

Some implementations allow for the normal "healthy" accessibility to a Storage Volume on the "other" storage processor through ports on "this" storage processor to be overridden. Normally in an asymmetric system this accessibility is "Standby" or "Active-NonOptimized". This override gives Active-Optimized, or full bandwidth access to this single volume.

This is modeled by an additional instance of StorageResourceLoadGroup that collects the subject volume together with an instance of AsymmetricAccessibility that associates that special StorageResourceLoadGroup with the TargetPortGroup. The properties on AsymmetricAccessibility reflect the override. This profile does not support the action that creates or removes the override. Methods of this profile that relate to assignment of affinity operate on the default static instance of StorageResourceLoadGroup only.

21.1.6.1.5 StorageProcessorAffinity

This subclass of CIM_Dependency associates instances of StorageResourceLoadGroup in a Redundancy Set to each instance of CIM_ComputerSystem representing a storage processor. Primary and Active properties are used to surface what the affinity is in both healthy and failed situations, and which storage processor owns the resource group which is where the Load Group will fail back to.

21.1.6.1.6 Asymmetric Accessibility

This subclass of CIM_Dependency associates instances of StorageResourceLoadGroup in a Redundancy Set to each instance of CIM_CIM_TargetPortGroup in the same RedundancySet. The AccessibilityState surfaces both the current and normal (healthy) accessibility of volumes in the LoadGroup from ports in the Port Group.

21.1.6.2 Instance Diagrams

21.1.6.2.1 General

The following instance diagrams provide show various asymmetry use cases. They are extensions of the MCS model, but for readability do not show Hosting and SystemDevice relationships. All instances are scoped to the top-level system.

Figure 87 shows the Asymmetry instances in context of the Multiple Computer System Profile for a dual redundant storage server.

Figure 87, Figure 88, Figure 89, and Figure 90 do not show the RedundancySet-related classes.

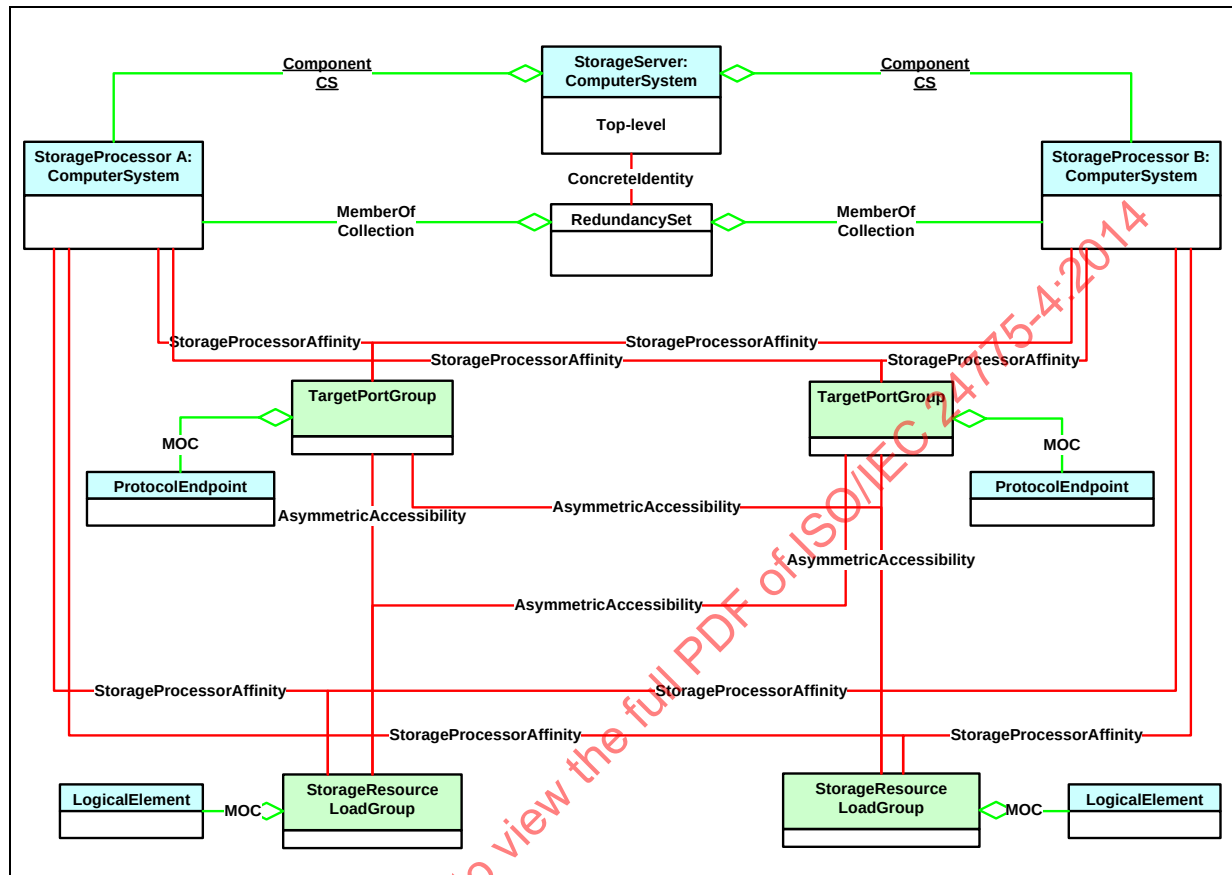


Figure 87 - Asymmetry with MCS

21.1.6.2.2 Multiple Tiers of Systems

Not shown is a system that has three tiers (see Clause 31 Multiple Computer System Subprofile in *Information Technology – Storage Management, Part 3 Common Profiles*). This type of system may aggregate storage processors into more than one redundant-failover subsystem. These subsystems are then clustered in a non-failover, but load-balancing relationship to form the top-level Storage server. In this type of system, StorageProcessorAffinity associations would be contained within failover subsystems, but AsymmetricAccessibility associations may span subsystem boundaries to reflect mid-level load-balancing paths.

21.1.6.2.3 Non-Transparent Asymmetry Cases

Figure 88: "Ports Do Not Failover, Healthy" and Figure 89: "Ports Do Not Failover, Failed Controller" are instance diagrams that show the model for healthy and failed situations in a non-transparent port implementation. Because the ports and thus the Target Port Group do not

The diagram illustrates the architecture of a Storage Server, showing the relationships between various components and their properties.

Top Level Components:

- StorageServer: ComputerSystem** (Top-level)
- StorageProcessor A: ComputerSystem**
- StorageProcessor B: ComputerSystem**

Intermediate Components and Connections:

- StorageProcessor A/B** are connected to **TargetPortGroup** via **StorageProcessorAffinity** (IsPrimary=true, IsActive=true).
- TargetPortGroup** is connected to **ProtocolEndpoint** via **MOC**.
- TargetPortGroup** is connected to **AsymmetricAccessibility** (State=ActiveOptimized | ActiveNonOptimized | Standby).
- ProtocolEndpoint** is connected to **AsymmetricAccessibility** (State=ActiveOptimized | ActiveNonOptimized | Standby).
- AsymmetricAccessibility** is connected to **StorageResource LoadGroup** via **StorageProcessorAffinity** (IsPrimary=false, IsActive=false).
- StorageResource LoadGroup** is connected to **LogicalElement** via **MOC**.

Properties and States:

- StorageProcessorAffinity:** IsPrimary=true, IsActive=true.
- StorageProcessorAffinity:** IsPrimary=false, IsActive=false.
- AsymmetricAccessibility:** State=ActiveOptimized | ActiveNonOptimized | Standby.
- StorageResource LoadGroup:** LogicalElement.

Figure 88 - Ports Do Not Failover, Healthy

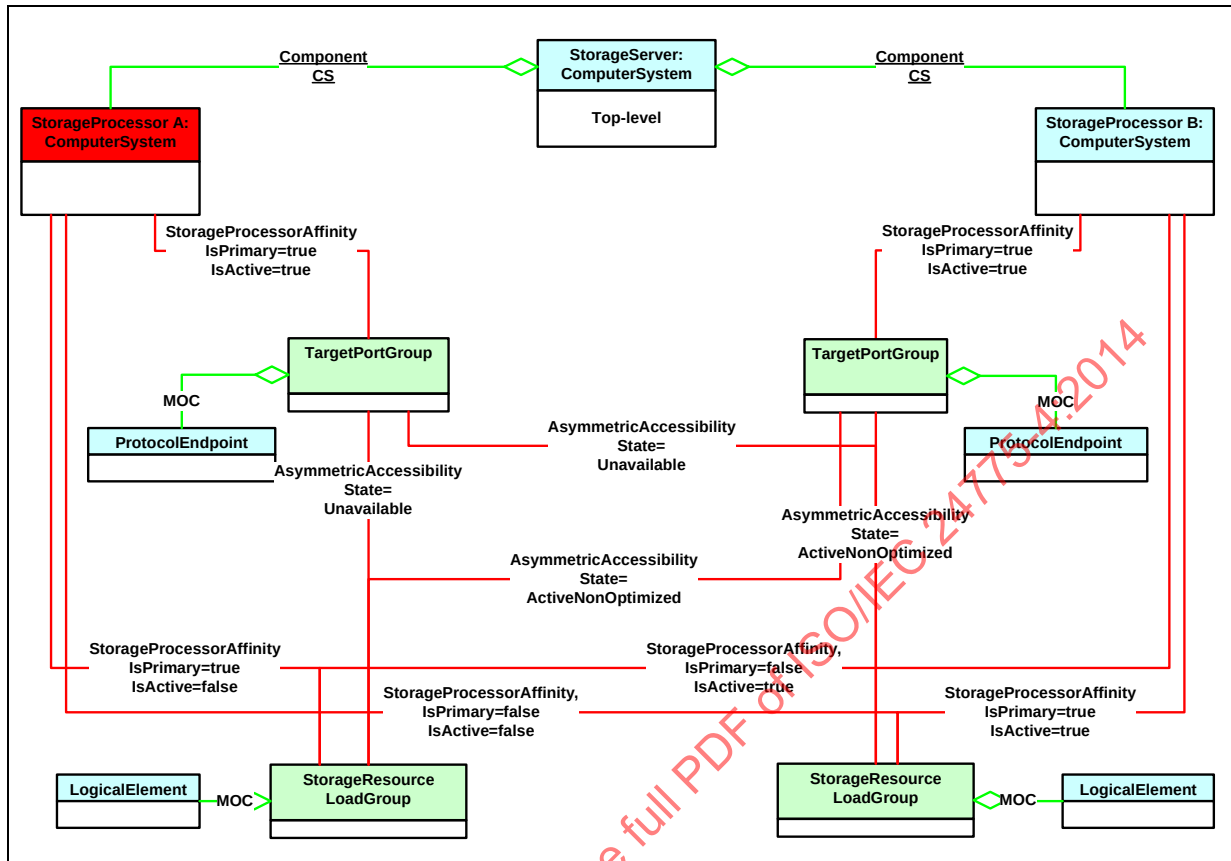


Figure 89 - Ports Do Not Failover, Failed Controller

21.1.6.2.4 Transparent Asymmetry Cases

Figure 90: "Ports Failover, Healthy" and Figure 91: "Ports Failover, Failed Controller" are instance diagrams that show the model for healthy and failed situations in a transparent failover port implementation.

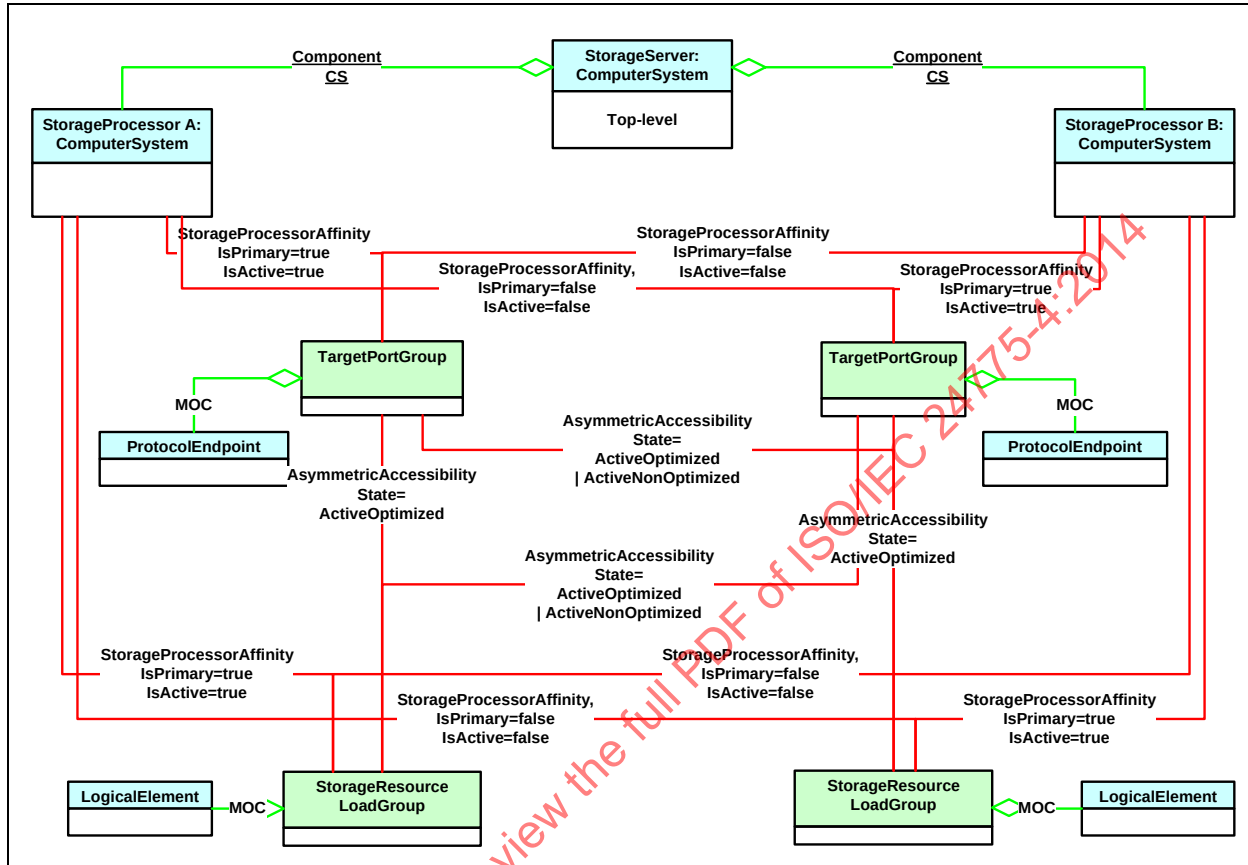


Figure 90 - Ports Failover, Healthy

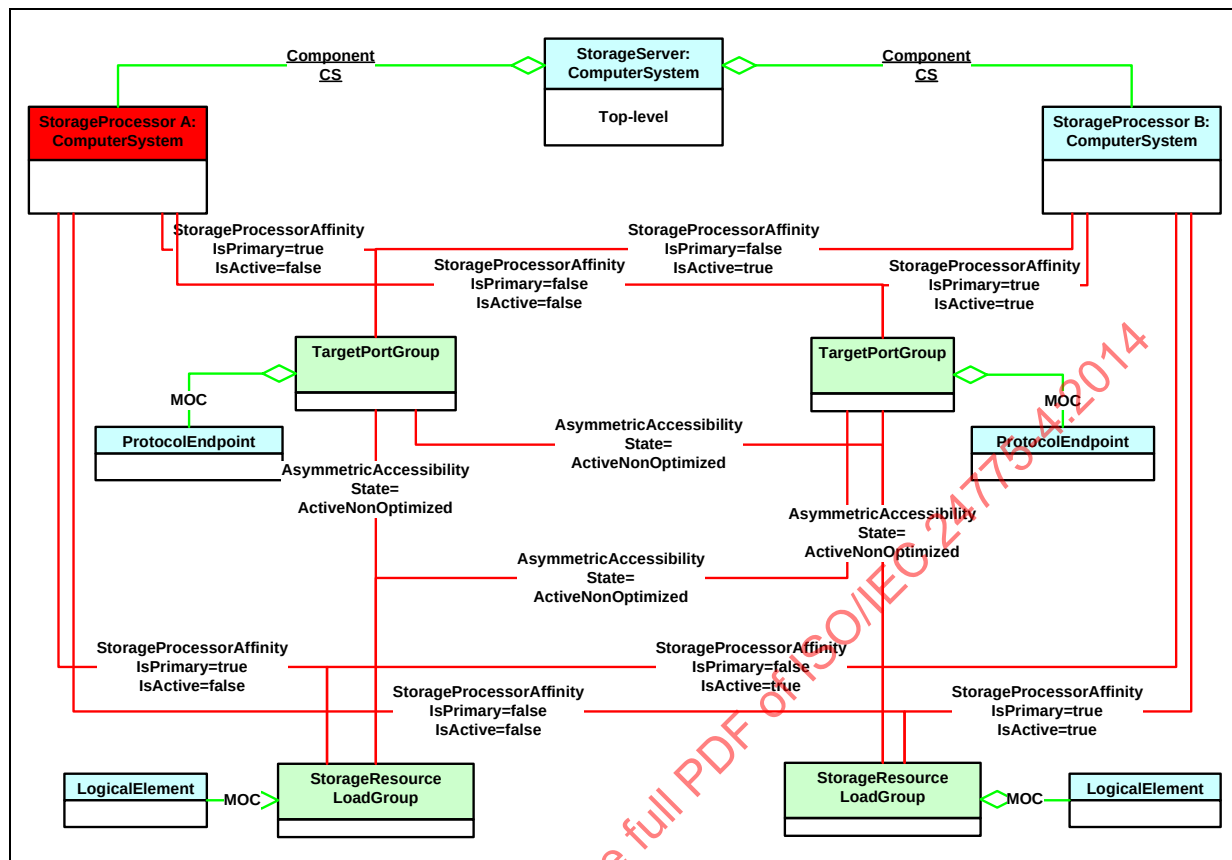


Figure 91 - Ports Failover, Failed Controller

21.2 Health and Fault Management Consideration

None.

21.3 Cascading Considerations

None.

21.4 Supported Profiles, Subprofiles, and Packages

None.

21.5 Methods of the Profile

21.5.1 Assign Storage Resource Affinity

21.5.1.1 General

This profile specific method of CIM_StorageConfigurationService starts a job to assign affinity of a StoragePool(s) or StorageVolume(s) to a storage processor. At the conclusion of the operation, the resource will be associated by CIM_MemberOfCollection to the StorageResourceLoadGroup with the primary affinity for the specified storage processor. The existing instance of CIM_MemberOfCollection to the existing StorageResourceLoadGroup is deleted.

Support for this method is indicated by the presence of an instance of StorageServerAsymmetryCapabilities in which the property StorageResourceAffinityAssignable is 'true'. If 0 is returned, the function completed successfully and no ConcreteJob instance was required. If 4096/0x1000 is returned, a job will be started to assign the element. The Job's reference will be returned in the output parameter Job.

AssignStorageResourceAffinity

IN, string **ResourceType**

This specifies whether the resource is a StorageVolume (= 2) or StoragePool (= 3).

OUT, CIM_ConcreteJob REF **JOB**

Reference to a job which may be created (may be null if job completed).

IN, CIM_ComputerSystem REF **StorageProcessor**

Reference to the storage processor to which to assign the resource.

IN, CIM_LogicalElement REF **StorageResources[]**

Array of references to storage resource instances to be assigned.

21.5.1.2 Return Codes

Completed with No Error - 0

Not Supported - 1

Unknown - 2

Timeout - 3

Failed - 4

Invalid Parameter - 5

In Use - 6

Method Parameters Checked - Job Started - 4096

Size Not Supported - 4097

21.6 Client Considerations and Recipes

21.6.1 Determine which ports provide full bandwidth access to a storage element

```
//  
// DESCRIPTION  
// Determine which ports on a storage server provide full  
// bandwidth access to a storage volume.  
//  
// PRE-EXISTING CONDITIONS AND ASSUMPTIONS  
// 1. The Top-Level ComputerSystem representing the target system of  
//    interest  
//    has been previously identified and defined in the $StorageServer->  
//    variable.  
//  
// 2. The CIM_StorageVolume of interest has been previously identified  
//    and defined in the $StorageVolume-> variable.  
//  
// MAIN  
// Step 1. Locate the instance of CIM_StorageServerAsymmetryCapabilities  
//    associated to the  
//    target ComputerSystem to insure the profile is supported.  
//
```

```

$StorageServerAsymmetryCapabilities[] = Associators($StorageServer->,
    "CIM_ElementCapabilities",
    "CIM_StorageServerAsymmetryCapabilities",
    "ManagedElement",
    "Capabilities",
    false,
    false,
    {"StorageResourceSymmetryCapability"})

if ($StorageServerAsymmetryCapabilities[] == null ||
    $StorageServerAsymmetryCapabilities[].length
    != 1) {
    <ERROR! The profile capabilities could not be found>
}
// Step 2. Check to see if this server has symmetric behavior.
// If so, exit here as an optimization.
//
if (
    $StorageServerAsymmetryCapabilities[0].Storage
    ResourceSymmetryCapability == 2 ) //
    Symmetric
    { <EXIT! Symmetric. All ports on the server provide full bandwidth
      access.> }

// Step 3. Find the Storage Resource Load Group to which this volume
    belongs.
//
$StorageResourceLoadGroup->[] = AssociatorNames($StorageVolume->,
    "CIM_MemberOfCollection",
    "CIM_StorageResourceLoadGroup",
    "ManagedElement",
    "Collection")
if ($StorageResourceLoadGroup[] == null ||
    $StorageResourceLoadGroup[].length != 1)
    { <ERROR! Volume must be a member of one and only one Load Group > }

// Step 4. Find the Target Port groups whose member ports provide full
// bandwidth access to the subject volume, and collect the port
    references for
// each such port group.
//
$AsymmetricAccessibility[] = References($StorageResourceLoadGroup->[0],
    "CIM_AsymmetricAccessibility",
    "Dependent",
    false,
    false,
    {"Antecedent", "NormalAccessState"})

#index = 0
for #i in $AsymmetricAccessibility[] {
    if ( $AsymmetricAccessibility[#i].NormalAccessState == 5 ) { // Active
        Optimized
    }
}

```

```

$Ports->[] =
    AssociatorNames($AsymmetricAccessibility[#i].
    Antecedent,
    "CIM_MemberOfCollection",
    "CIM_ProtocolEndpoint",
    "Collection",
    "ManagedElement")

if ($Ports->[] != null ) {
    for #j in $Ports->[] {
        $FullAccessPorts->[#index] = $Ports->[#j]
        #index++
    }
}
}

<EXIT: $Ports will contain the references to the ProtocolEndpoints
representing the ports which
will give full bandwidth access to the volume.>

```

21.7 Registered Name and Version

Storage Server Asymmetry version 1.4.0 (Component Profile).

21.8 CIM Elements

21.8.1 General

Table 380 describes the CIM elements for Storage Server Asymmetry.

Table 380 - CIM Elements for Storage Server Asymmetry

Element Name	Requirement	Description
21.8.2 CIM_AsymmetricAccessibility	Mandatory	This association indicates the accessibility of StorageVolumes in the StorageResourceLoadGroup through ports in the associated TargetPortGroup.
21.8.3 CIM_ElementCapabilities (To Top-level ComputerSystem)	Mandatory	
21.8.4 CIM_HostedCollection (Top-Level System to Load Group)	Mandatory	Associates the instances of StorageResourceLoadGroup to the Top-Level ComputerSystem. Enables a Client to find these groups without first traversing to each Storage Processor ComputerSystem.
21.8.5 CIM_HostedCollection (Top-Level System to Port Group)	Mandatory	Associates the instances of TargetPortGroup to the Top-Level ComputerSystem. Enables a Client to find these groups without first traversing to each Storage Processor ComputerSystem.
21.8.6 CIM_MemberOfCollection (SATA Target Port Group)	Conditional	Conditional requirement: Requires TargetPortGroup to aggregate CIM_ProtocolEndpoint. Used to aggregate SATA Target Ports in a Target Port Group.
21.8.7 CIM_MemberOfCollection (SB Target Port Group)	Conditional	Conditional requirement: Requires TargetPortGroup to aggregate SNIA_SBProtocolEndpoint. Used to aggregate SB Target Ports in a Target Port Group.

Table 380 - CIM Elements for Storage Server Asymmetry (Continued)

Element Name	Requirement	Description
21.8.8 CIM_MemberOfCollection (SCSI Target Port Group)	Conditional	Conditional requirement: Requires TargetPortGroup to aggregate CIM_SCSIProtocolEndpoint or Requires TargetPortGroup to aggregate CIM_SCSIProtocolEndpoint or Requires TargetPortGroup to aggregate CIM_SCSIProtocolEndpoint or Requires TargetPortGroup to aggregate CIM_SCSIProtocolEndpoint. Used to aggregate DA, FC, SPI, or SAS Target Ports in a Target Port Group.
21.8.9 CIM_MemberOfCollection (Storage Resource Load Group aggregating Storage Pools)	Conditional	Conditional requirement: Requires StorageResourceLoadGroup to aggregate CIM_StoragePool. Aggregates Storage Pools in a Storage Resource Load Group.
21.8.10 CIM_MemberOfCollection (Storage Resource Load Group aggregating Storage Volumes)	Conditional	Conditional requirement: Requires StorageResourceLoadGroup to aggregate CIM_StorageVolume. Aggregates Storage Volumes in a Storage Resource Load Group.
21.8.11 CIM_MemberOfCollection (iSCSI Target Port Group)	Conditional	Conditional requirement: Requires TargetPortGroup to aggregate CIM_iSCSIProtocolEndpoint. Used to aggregate iSCSI Target Ports in a Target Port Group.
21.8.12 CIM_StorageConfigurationService	Optional	
21.8.13 CIM_StorageProcessorAffinity (StorageResourceLoadGroup)	Mandatory	Indicates a processing affinity and state between a TargetPortGroup and a ComputerSystem representing a storage processor in a redundant storage server. The processor can host the group in either a healthy or failover state. Instances of this association are static, one for each combination of StorageResourceLoadGroup and ComputerSystem in the RedundancySet.
21.8.14 CIM_StorageProcessorAffinity (Target Port Group)	Mandatory	Indicates a processing affinity and state between a TargetPortGroup and a ComputerSystem representing a storage processor in a redundant storage server. The processor can host the group in either a healthy or failover state. Instances of this association are static, one for each combination of StorageResourceLoadGroup and ComputerSystem in the RedundancySet.
21.8.15 CIM_StorageResourceLoadGroup (Load Groups)	Mandatory	StorageResourceLoadGroup aggregates either the StoragePools or the individual StorageVolumes that have the same affinity for a storage processor. The affinity of this group may change during failover or fallback/rebind from one storage processor to another in a storage server. StorageResourceLoadGroup has a instance of the StorageProcessorAffinity association to each instance of CIM_ComputerSystem representing a storage processor that may host the StorageResourceLoadGroup in either a healthy or failover state. Each instance of StorageResourceLoadGroup in a storage server is also associated to each instance of TargetPortGroup in the server by the AsymmetricAccessibility class.

Table 380 - CIM Elements for Storage Server Asymmetry (Continued)

Element Name	Requirement	Description
21.8.16 CIM_StorageServerAsymmetryCapabilities	Mandatory	This class defines the asymmetric characteristics and capabilities of a redundant storage server. The properties in this class guide client algorithms in the interpretation of the instances of StorageResourceLoadGroup, TargetPortGroup, StorageProcessorAffinity, and AsymmetricAccessibility, and also determining support for methods that affect assignment of storage resources to storage processors.
21.8.17 CIM_TargetPortGroup (Port Groups)	Mandatory	TargetPortGroup aggregates the ProtocolEndpoints representing a group of target ports in a storage server. The ProtocolEndpoints may be a subclass of CIM_ProtocolEndpoint as appropriate for the type of target port implemented by the storage server. The target ports are aggregated because they have the same affinity for an associated storage processor for failover and the same accessibility state to storage resources in a given StorageResourceLoadGroup. The TargetPortGroup may have either a fixed affinity for a storage processor within the server or an affinity that changes during failover from one storage processors to another. TargetPortGroup has a instance of the StorageProcessorAffinity association to each instance of CIM_ComputerSystem representing a storage processor that may host the TargetPortGroup in either a healthy or failover state. Each instance of TargetPortGroup in a storage server is also associated to each instance of StorageResourceLoadGroup in the server by the AsymmetricAccessibility class.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_StorageProcessorAffinity AND SourceInstance.CIM_StorageProcessorAffinity::IsActive <> PreviousInstance.CIM_StorageProcessorAffinity::IsActive	Mandatory	CQL - Change in Affinity of a StorageResourceLoadGroup.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_StorageProcessorAffinity AND SourceInstance.IsActive <> PreviousInstance.IsActive	Mandatory	Deprecated WQL - Change in Affinity of a StorageResourceLoadGroup.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_AsymmetricAccessibility AND SourceInstance.CIM_AsymmetricAccessibility::CurrentAccessState <> PreviousInstance.CIM_AsymmetricAccessibility::CurrentAccessState	Mandatory	CQL - Modification of accessibility to a storage element.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_AsymmetricAccessibility AND SourceInstance.CurrentAccessState <> PreviousInstance.CurrentAccessState	Mandatory	Deprecated WQL - Modification of accessibility to a storage element.

21.8.2 CIM_AsymmetricAccessibility

Created By: Static

Modified By: External

Deleted By: Static

Requirement: Mandatory.

Table 381 describes class CIM_AsymmetricAccessibility.

Table 381 - SMI Referenced Properties/Methods for CIM_AsymmetricAccessibility

Properties	Flags	Requirement	Description and Notes
CurrentAccessState		Mandatory	This property indicates the current accessibility state of volumes in the StorageResourceLoadGroup through ports in the TargetPortGroup. With the exception of 'Unavailable', the states are those defined in INCITS 513 (future ISO/IEC 14776-454, see Bibliography), Target Port Group Access States clause. 2 (Unavailable): The volumes are not accessible in any way. 3 (Standby): No data access to the volume is possible. Status and other non-data access commands are available. 4 (Active Non-Optimized): Data access to the volume is available at less than full bandwidth. 5 (Active Optimized): Data access to the volume is available at full bandwidth.
NormalAccessState		Mandatory	This property indicates the accessibility state of volumes in the StorageResourceLoadGroup through ports in the TargetPortGroup when the primary storage processor hosting the groups is healthy. With the exception of 'Unavailable', the states are those defined in INCITS 513 (future ISO/IEC 14776-454, see Bibliography), Target Port Group Access States clause. 2 (Unavailable): The volumes are not accessible in any way. 3 (Standby): No data access to the volume is possible. Status and other non-data access commands are available. 4 (Active Non-Optimized): Data access to the volume is available at less than full bandwidth. 5 (Active Optimized): Data access to the volume is available at full bandwidth.
Antecedent		Mandatory	The Port Group.
Dependent		Mandatory	The Storage Resource Load Group.

21.8.3 CIM_ElementCapabilities (To Top-level ComputerSystem)

Created By: Static

Modified By: External

Deleted By: Static

Requirement: Mandatory.

Table 382 describes class CIM_ElementCapabilities (To Top-level ComputerSystem).

Table 382 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (To Top-level ComputerSystem)

Properties	Flags	Requirement	Description and Notes
ManagedElement		Mandatory	The Top-level Storage Server ComputerSystem.
Capabilities		Mandatory	StorageServerAsymmetryCapabilities.

21.8.4 CIM_HostedCollection (Top-Level System to Load Group)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 383 describes class CIM_HostedCollection (Top-Level System to Load Group).

**Table 383 - SMI Referenced Properties/Methods for CIM_HostedCollection
(Top-Level System to Load Group)**

Properties	Flags	Requirement	Description and Notes
Antecedent		Mandatory	
Dependent		Mandatory	

21.8.5 CIM_HostedCollection (Top-Level System to Port Group)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 384 describes class CIM_HostedCollection (Top-Level System to Port Group).

**Table 384 - SMI Referenced Properties/Methods for CIM_HostedCollection
(Top-Level System to Port Group)**

Properties	Flags	Requirement	Description and Notes
Antecedent		Mandatory	
Dependent		Mandatory	

21.8.6 CIM_MemberOfCollection (SATA Target Port Group)

Created By: Static

Modified By: External

Deleted By: Static

Requirement: Requires TargetPortGroup to aggregate CIM_ProtocolEndpoint.

Table 385 describes class CIM_MemberOfCollection (SATA Target Port Group).

**Table 385 - SMI Referenced Properties/Methods for CIM_MemberOfCollection
(SATA Target Port Group)**

Properties	Flags	Requirement	Description and Notes
Collection		Mandatory	The Target Port Group.
Member		Mandatory	The SATA Target Ports.

21.8.7 CIM_MemberOfCollection (SB Target Port Group)

Created By: Static

Modified By: External

Deleted By: Static

Requirement: Requires TargetPortGroup to aggregate SNIA_SBProtocolEndpoint.

Table 386 describes class CIM_MemberOfCollection (SB Target Port Group).

**Table 386 - SMI Referenced Properties/Methods for CIM_MemberOfCollection
(SB Target Port Group)**

Properties	Flags	Requirement	Description and Notes
Collection		Mandatory	The Target Port Group.
Member		Mandatory	The SB Target Ports.

21.8.8 CIM_MemberOfCollection (SCSI Target Port Group)

Created By: Static

Modified By: External

Deleted By: Static

Requirement: Requires TargetPortGroup to aggregate CIM_SCSIProtocolEndpoint or Requires TargetPortGroup to aggregate CIM_SCSIProtocolEndpoint or Requires TargetPortGroup to aggregate CIM_SCSIProtocolEndpoint or Requires TargetPortGroup to aggregate CIM_SCSIProtocolEndpoint.

Table 387 describes class CIM_MemberOfCollection (SCSI Target Port Group).

**Table 387 - SMI Referenced Properties/Methods for CIM_MemberOfCollection
(SCSI Target Port Group)**

Properties	Flags	Requirement	Description and Notes
Collection		Mandatory	The Target Port Group.
Member		Mandatory	The DA, FC, SPI, or SAS Target Ports.

21.8.9 CIM_MemberOfCollection (Storage Resource Load Group aggregating Storage Pools)

Created By: Static

Modified By: External

Deleted By: Static

Requirement: Requires StorageResourceLoadGroup to aggregate CIM_StoragePool.

Table 388 describes class CIM_MemberOfCollection (Storage Resource Load Group aggregating Storage Pools).

**Table 388 - SMI Referenced Properties/Methods for CIM_MemberOfCollection
(Storage Resource Load Group aggregating Storage Pools)**

Properties	Flags	Requirement	Description and Notes
Collection		Mandatory	The Storage Resource Load Group.
Member		Mandatory	The StoragePools.

21.8.10 CIM_MemberOfCollection (Storage Resource Load Group aggregating Storage Volumes)

Created By: Static

Modified By: External

Deleted By: Static

Requirement: Requires StorageResourceLoadGroup to aggregate CIM_StorageVolume.

Table 389 describes class CIM_MemberOfCollection (Storage Resource Load Group aggregating Storage Volumes).

Table 389 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Storage Resource Load Group aggregating Storage Volumes)

Properties	Flags	Requirement	Description and Notes
Collection		Mandatory	The Storage Resource Load Group.
Member		Mandatory	The Storage Volumes.

21.8.11 CIM_MemberOfCollection (iSCSI Target Port Group)

Created By: Static

Modified By: External

Deleted By: Static

Requirement: Requires TargetPortGroup to aggregate CIM_iSCSIProtocolEndpoint.

Table 390 describes class CIM_MemberOfCollection (iSCSI Target Port Group).

Table 390 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (iSCSI Target Port Group)

Properties	Flags	Requirement	Description and Notes
Collection		Mandatory	The Target Port Group.
Member		Mandatory	The iSCSI Target Ports.

21.8.12 CIM_StorageConfigurationService

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 391 describes class CIM_StorageConfigurationService.

Table 391 - SMI Referenced Properties/Methods for CIM_StorageConfigurationService

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	
CreationClassName		Mandatory	

Table 391 - SMI Referenced Properties/Methods for CIM_StorageConfigurationService

Properties	Flags	Requirement	Description and Notes
SystemName		Mandatory	
Name		Mandatory	
AssignStorageResourceAffinity()		Optional	Start a job to assign affinity of a StoragePool(s) or StorageVolume(s) to a storage processor. At the conclusion of the operation, the resource will be a member of the StorageResourceLoadGroup with the primary affinity for the specified storage processor. Support for this method is indicated by the presence of an instance of StorageServerAsymmetryCapabilities in which the property StorageResourceAffinityAssignable is 'true'. If 0 is returned, the function completed successfully and no ConcreteJob instance was required. If 4096/0x1000 is returned, a job will be started to assign the element. The Job's reference will be returned in the output parameter Job.

21.8.13 CIM_StorageProcessorAffinity (StorageResourceLoadGroup)

Created By: Static

Modified By: External

Deleted By: Static

Requirement: Mandatory.

Table 392 describes class CIM_StorageProcessorAffinity (StorageResourceLoadGroup).

Table 392 - SMI Referenced Properties/Methods for CIM_StorageProcessorAffinity (StorageResourceLoadGroup)

Properties	Flags	Requirement	Description and Notes
IsPrimary		Mandatory	This property is set to true if the TargetPortGroup is hosted by the storage processor when the processor is healthy. It is set to false if the group can be hosted by the processor when the primary storage processor for the group has failed. For each StorageResourceLoadGroup, one instance of StorageProcessorAffinity will have IsPrimary=true, the rest will have IsPrimary=false.
IsActive		Mandatory	This property is set to true if the StorageResourceLoadGroup is currently being hosted by the storage processor.
Antecedent		Mandatory	The storage processor for which the Storage Resource Load Group has affinity.
Dependent		Mandatory	The Storage Resource Load Group.

21.8.14 CIM_StorageProcessorAffinity (Target Port Group)

Created By: Static

Modified By: External

Deleted By: Static

Requirement: Mandatory.

Table 393 describes class CIM_StorageProcessorAffinity (Target Port Group).

Table 393 - SMI Referenced Properties/Methods for CIM_StorageProcessorAffinity (Target Port Group)

Properties	Flags	Requirement	Description and Notes
IsPrimary		Mandatory	This property is set to true if the TargetPortGroup is hosted by the storage processor when the processor is healthy. It is set to false if the group can be hosted by the processor when the primary storage processor for the group has failed. For each StorageResourceLoadGroup, one instance of StorageProcessorAffinity will have IsPrimary=true, the rest will have IsPrimary=false.
IsActive		Mandatory	This property is set to true if the TargetPortGroup is currently being hosted by the storage processor.
Antecedent		Mandatory	The storage processor for which the Port Group has affinity.
Dependent		Mandatory	The Target Port Group.

21.8.15 CIM_StorageResourceLoadGroup (Load Groups)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

21.8.16 CIM_StorageServerAsymmetryCapabilities

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 394 describes class CIM_StorageServerAsymmetryCapabilities.

Table 394 - SMI Referenced Properties/Methods for CIM_StorageServerAsymmetryCapabilities

Properties	Flags	Requirement	Description and Notes
StorageResourceSymmetryCapability		Mandatory	If this property is set to Symmetric it indicates that the StoragePools or StorageVolumes are processed in a distributed load-balanced manner between storage processors. If this property is set to Asymmetric it indicates that the StoragePools or StorageVolumes have a primary affinity for one storage processor.
StorageResourceType		Mandatory	If this property is set to StorageVolume it indicates that the StoragePools have symmetric behavior (or no affinity) and that the Volumes have affinity for one storage processor or the other. If this property is set to StoragePool it indicates that a StoragePool as well as the Volumes allocated from it have affinity for one storage processor or the other.
StorageResourceAffinityAssignable		Mandatory	Set to true if this storage system allows the client to specify which storage processor a storage resource is assigned to, either using one of the CreateOrModify methods or the AssignStorageResourceAffinity method on StorageConfigurationService.
PortGroupFailoverBehavior		Mandatory	This property specifies whether a storage server supports transparent or non-transparent failover of TargetPortGroups. If this value is 2 (Port Group Fails), a TargetPortGroup will have a single StorageProcessorAffinity association to the storage processor it belongs to and will fail with it. If this property has a value of 3, the TargetPortGroup will have a StorageProcessorAffinity association to each storage processor that can host its function, and the properties on the association will indicate both which processor is primary and which is currently hosting the ports in the group.
TargetPortSymmetryCapability		Mandatory	This property indicates the normal (healthy) state accessibility to volumes both in the StorageResourceLoadGroup on the same storage processor as a TargetPortGroup, and to volumes in StorageResourceLoadGroups on 'other' storage processors in the redundant server. If this value is 2 (Symmetric): There is equal bandwidth access to volumes on all storage processors through target ports on this storage processor. If this value is 3 (Asymmetric Non-Optimized): There is full bandwidth access to volumes in the StorageResourceLoadGroup on the same storage processor as the TargetPortGroup and degraded bandwidth access to volumes in the StorageResourceLoadGroups on the 'other' storage processors. If this value is 4 (Asymmetric No Access): There is full bandwidth access to volumes in the StorageResourceLoadGroup on the same storage processor as the TargetPortGroup and no access to volumes on 'other' storage processors.

21.8.17 CIM_TargetPortGroup (Port Groups)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

EXPERIMENTAL

IECNORM.COM : Click to view the full PDF of ISO/IEC 24775-4:2014

DEPRECATED

22 Block Services Resource Ownership Subprofile

22.1 Description

22.1.1 General

The Block Services Resource Ownership common subprofile models control over the rights of a client to grant or deny access to block storage resources, as shown in Figure 92. By asserting exclusive control over these rights, one client can control which other clients may access those resources. This subprofile is intended for environments in which multiple CIM clients may not be completely aware of each other's activities, making it important that use of the resource not be disrupted by a client that is unaware of shared resource use. Specific examples include use of a volume by in-band virtualizers and NAS gateways, where attempts to manage the volume by clients not associated with this use could be seriously disruptive. An intended configuration is that a CIM client exists in the cascading device that has exclusive use of the volume. The Resource Ownership Subprofile is optional.

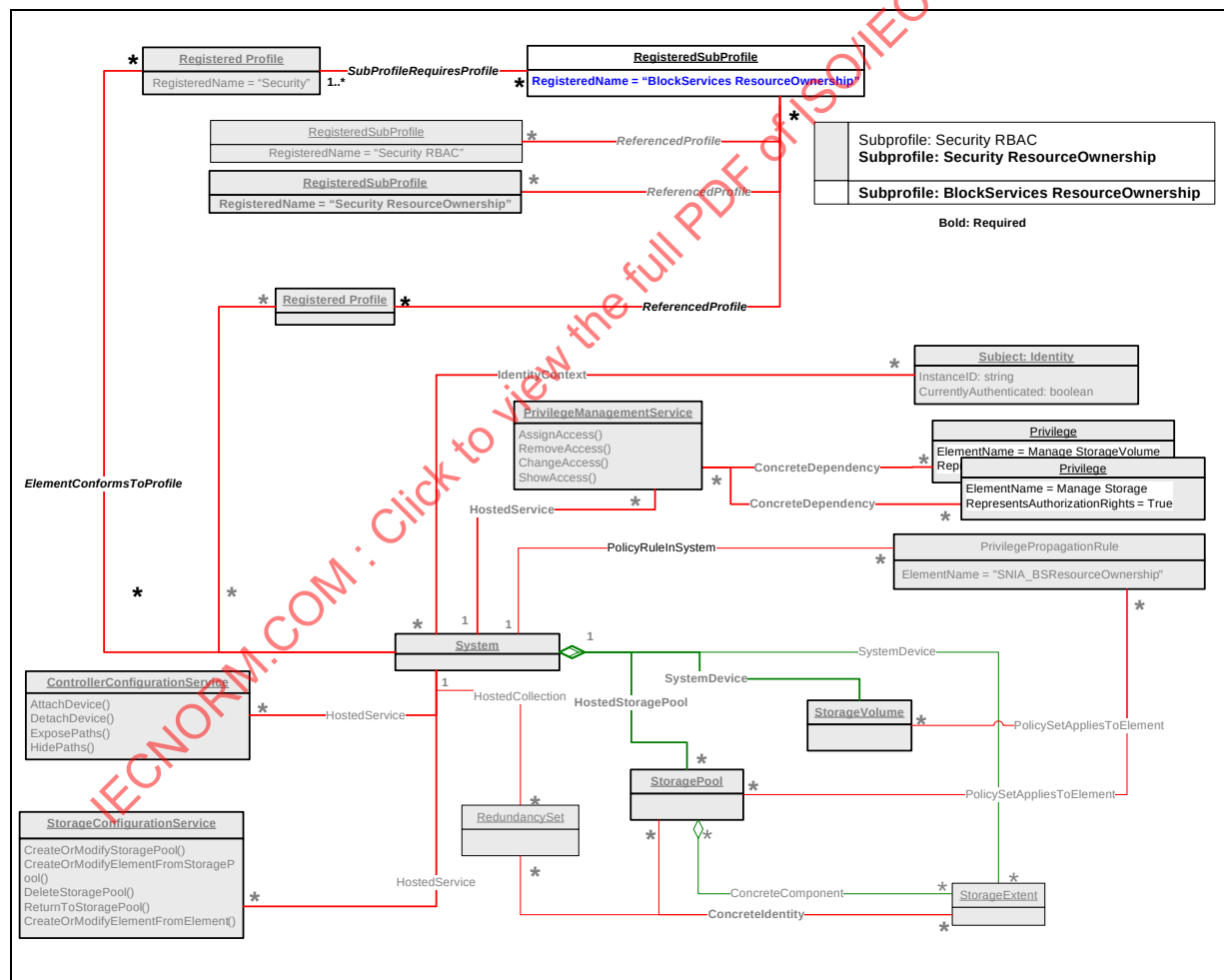


Figure 92 - Resource Ownership for Block Services

This profile concerns itself with the existence and use of two sets of rights which may be realized as two Privilege instances that are associated via ConcreteDependency to a

PrivilegeManagementService. There is one Privilege to "Manage StorageVolume" and a superset of that to "Manage Storage". Each is described in Table 395.

Table 395 - Block Service Management Rights

ElementName	Property	Index	Value
Manage StorageVolume	Activities	0	Execute
	QualifiersFormats	0	<class>.method
	ActivityQualifiers	0	StorageConfigurationService. ReturnToStoragePool StorageConfigurationService. CreateorModifyElementfromElements StorageConfigurationService.AttachDevice, StorageConfigurationService.DetachDevice, StorageConfigurationService.ExposePaths, StorageConfigurationService.HidePaths, PrivilegeManagementService.AssignAccess, PrivilegeManagementService.ChangeAccess, ModifyInstance, SetProperty
Manage Storage	Activities	0	Execute
	QualifiersFormats	0	<class>.method
	ActivityQualifiers	0	StorageConfigurationService. CreateOrModifyStoragePool, StorageConfigurationService. CreateOrModifyElementFromStoragePool, StorageConfigurationService. DeleteStoragePool, StorageConfigurationService. ReturnToStoragePool, StorageConfigurationService. CreateorModifyElementfromElements, ControllerConfigurationService.AttachDevice, ControllerConfigurationService.DetachDevice, ControllerConfigurationService.ExposePaths, ControllerConfigurationService.HidePaths, PrivilegeManagementService.AssignAccess, PrivilegeManagementService.ChangeAccess, ModifyInstance, SetProperty

This profile assumes that the intrinsic CreateInstance and DeleteInstance methods are not supported for either StorageVolumes or StoragePools.

With RepresentsAuthorizationRights set to True, the ChangeAccess call may be used to assign "Manage StorageVolume" rights to a StorageVolume for a particular set of subjects, each represented by an Identity. Once this assignment is made, only members of that set of subjects are permitted to assign "Manage StorageVolume" rights to other subjects, regardless of the setting of RepresentsAuthorizationRights. The ShowAccess call may be used to list the rights granted to a particular subject Identity and target StorageVolume or StoragePool.

To establish an "Owner" in the sense meant by this profile, only one subject is assigned the "Manage StorageVolume" privilege with RepresentsAuthorizationRights set both to True and False.

The same strategy is used to assign "Manage Storage" rights to a StoragePool.

Even though the SMI-S 1.1 ExposePaths and HidePaths extrinsics act on StorageVolumes by the LUID string parameter rather than a reference, nevertheless they are governed by authorization rights.

This profile requires that every StorageVolume allocated from a StoragePool that is governed by "Manage Storage" rights be assigned the corresponding "Manage StorageVolume" rights to the same subject. This is an implicit PrivilegePropagationRule, which need not be made explicit to be in effect. Whenever ChangeAccess, or other means, is used to modify the "Manage StorageVolume" rights of a particular subject to a StoragePool, those rights are propagated for that subject to all StorageVolumes that have an AllocatedFromStoragePool association to that StoragePool.

If an explicit PrivilegePropagationRule is used, it shall have ElementName set to "SNIA_BSResourceOwnership".

Optionally, a QueryCondition, (not shown), may be associated to that PrivilegePropagationRule via PolicyConditionInPolicyRule, (not shown). If specified the QueryCondition instance shall have its QueryLanguage property set to "2", meaning "CQL", its QueryResultName set to "SNIA_BSResourceOwnershipCondition" and its Query property set to

```
"SELECT (M.SourceInstanceHost || '/' || M.SourceInstanceModelPath) AS PMSPath,
        M.MethodParameters.Subject,
        M.MethodParameters.Target,
FROM CIM_InstMethodCall M,
WHERE M.MethodName = 'ChangeAccess'
AND    M.ReturnValue = 0
AND    M.PreCall = FALSE
AND    M.MethodParameters.Target ISA CIM_StoragePool
AND    ANY P IN M.MethodParameters.Privileges[*]
        SATISFIES (P.ElementName = 'ManageStorage')
```

Additionally, if this optional QueryCondition is associated then a corresponding MethodAction instance, (not shown), shall also be associated to the same PrivilegePropagationRule via PolicyActionInPolicyRule, (not shown). The MethodAction instance shall have its QueryLanguage property set to "2", meaning "CQL", its InstMethodCallName set to "SNIA_BSResourceOwnershipAction" and its Query property set to

```
"SELECT (BS.PMSPath || '/' || 'ChangeAccess') AS MethodName,
        BS.Subject AS Subject,
        ObjectPath(SV) AS Target,
        NULL AS PropagationPolicies,
        BS.Privileges AS Privileges
FROM SNIA_BSResourceOwnershipCondition BS,
        CIM_AllocatedFromStoragePool AFSP,
        CIM_StorageVolume SV,
        CIM_Privilege P
WHERE ObjectPath(SV) = AFSP.Dependent
AND    BS.Target = AFSP.Antecedent
AND    P.ElementName = 'Manage StorageVolume'
```

If AuthorizedSubject/AuthorizedTarget associations are implemented, then these need to be created as appropriate to reflect the assigned rights. In any case, a client may use ShowAccess to determine what privileges are in force for particular subject Identity, StorageVolume or StoragePool.

If the ChangeAccess request to establish ownership is not permitted, then the return status shall be CIM_ERR_ACCESS_DENIED. This result may be because the requestor is not permitted to make the call, or the requestor does not have sufficient rights to modify ownership of the target.

Some vendors may define additional vendor-specific extrinsic operations that need to be restricted in order to realize the functionality of Resource Ownership. Execution of each such vendor-specific extrinsics shall be added to the above list of restricted activities. Clients may

check for the presence of at least the above list of restricted activities, but shall not check for an exact match to the above list, as such a check may fail if there are vendor-specific extrinsics that are also restricted.

22.1.2 Design considerations

This list realizes a number of design decisions:

- For simplicity, the "Manage Storage" Privilege is a superset of the "Manage StorageVolume" Privilege. The "Manage Storage" Privilege may be used against both StorageVolumes and StoragePools. When applied to a StorageVolume, methods called out in that Privilege that do not affect StorageVolumes are simply ignored.
- The capability to own StoragePools is signaled by a PrivilegeManagementService with a ConcreteDependency.Dependent "Manage Storage" Privilege with RepresentsAuthorizationRights set to True.
- The "Manage StorageVolume" Privilege does not provide the ability to manage StoragePools.
- DeleteProtocolController is not restricted. The design goal is to control resource management in a fashion that keeps reasonably well-behaved clients from causing unintended problems. Control of the StoragePool and StorageVolume instances is sufficient, as a reasonably well-behaved client should at least call DetachDevice or HidePaths on the associated StorageVolumes before calling DeleteProtocolController (both DeleteDevice and HidePaths are controlled), or at the very least understand what the attached volumes are being used for before deleting the protocol controller. The ProtocolControllerforPort and the associated port (e.g., FCPort) are also not restricted for similar reasons.
- RemoveAccess and ChangeAccess are not restricted to avoid complexity. These could be restricted by creating a second type of resource ownership Privilege to control them, and the corresponding access Privileges to enforce the restrictions, but for 1.1, it seems reasonable to trust clients that don't know what they're doing to avoid invocations of RemoveAccess and ChangeAccess.
- ServiceAffectsElement associations are assumed between Services and affected elements. (See Figure 93: "ServiceAffectsElement Associations for ResourceOwnership".) This subprofile does not REQUIRE an implementation to present these associations unless there is more than one of a particular type Service in the profiled Namespace.

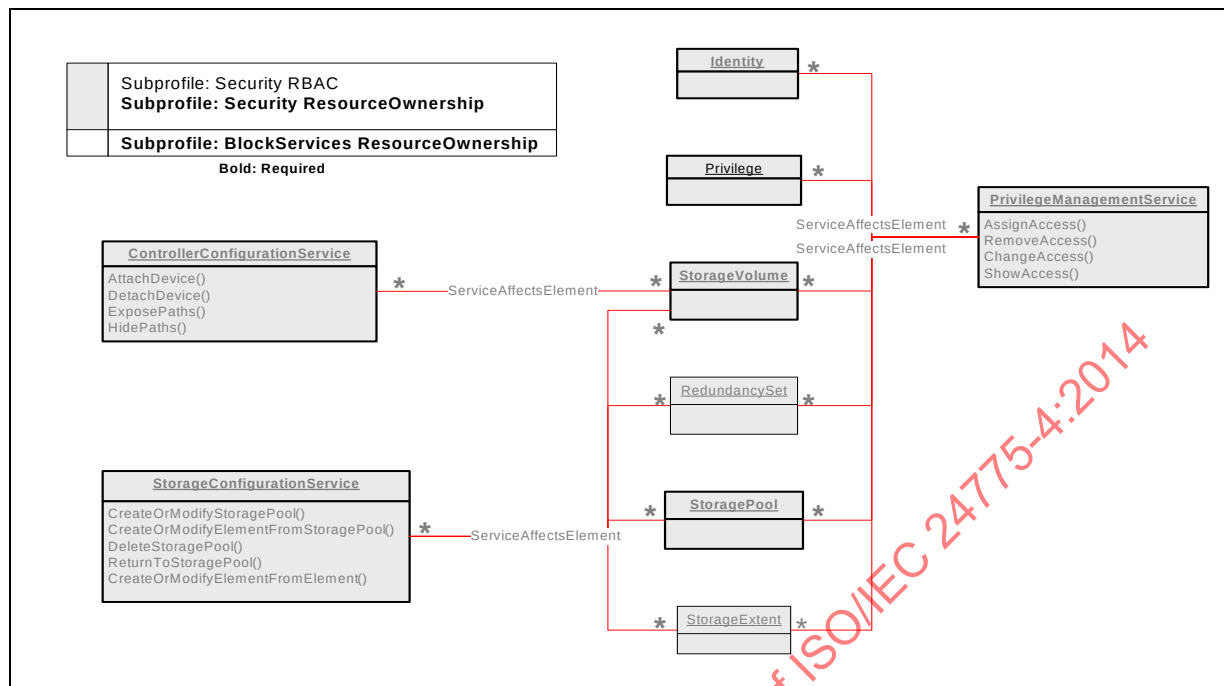


Figure 93 - ServiceAffectsElement Associations for ResourceOwnership

- AuthorizedPrivilege instances are assumed when a Privilege is granted to a subject or assigned to a target. (See Figure 94: "AuthorizedPrivilege Associations for ResourceOwnership".) AuthorizedTarget and AuthorizedSubject associations are assumed between the AuthorizedPrivilege and the target and subject entities respectively. This subprofile does not REQUIRE the implementation to make these instances explicit. Instead this profile relies on the ChangeAccess method to grant or deny rights and on the ShowAccess method to display rights.

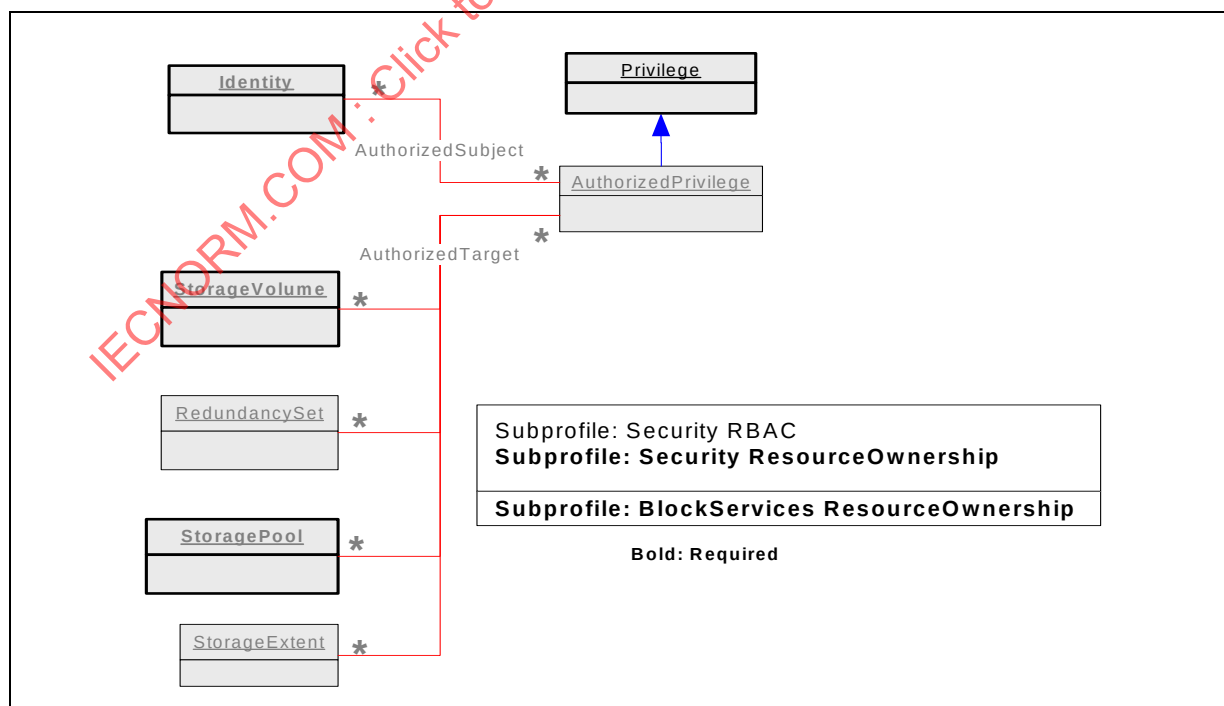


Figure 94 - AuthorizedPrivilege Associations for ResourceOwnership

- PrivilegePropagationRule instances are assumed with appropriate PolicySetAppliesToElement associations to StoragePool and StorageVolume instances and a PolicyRuleInSystem association to a System instances. This subprofile does not REQUIRE either the PrivilegePropagationRule instances nor the related association instances.

22.1.3 Privilege Propagation

Propagation is a means of restricting the number of AuthorizedTarget associations for a Privilege. Propagation has two elements:

- 1) Privilege restrictions on a StoragePool propagate to ConcreteComponent.PartComponent StorageExtents.
- 2) Privilege restrictions on a StoragePool propagate across ConcreteIdentity to a StorageExtent aspect. (For instance when a Raid 5 extent is used as a StoragePool.)
- 3) Privilege restrictions on a StorageExtent propagate across ConcreteIdentity to a RedundancySet aspect. (For instance when spares are available for a Raid 5 extent.)

To place these rules in force, a PrivilegePropagationRule instance is associated via PolicySetAppliesToElement to affected StoragePools or StorageVolumes. This rule shall have its ElementName set to "BlockServices ResourceOwnership", and it shall not have any PolicyCondition or PolicyAction instances associated with it.

ShowAccess may be used to determine the resulting behavior.

22.2 Client Considerations and Recipes

Resource Ownership Privileges can be distinguished from LUN Mapping/Masking privileges as the latter contain Execute (instance of Activities[]) cdb=* (ActivityQualifiers[]) SCSI Command (QualifierFormats[]).

A cascading provider determines whether or not Resource Ownership is supported by an array by looking for Block Services Resource Ownership as a RegisteredSubprofile of the Array Profile.

While this subprofile is intended to support cascading, it can be used with any CIM Client that can authenticate to the CIM Server and thereby obtain an authenticated Identity.

A client can determine whether resource ownership restrictions are enforced on a StorageVolume or StoragePool by using the ShowAccess method (preferred) or by association traversal via AuthorizedTarget to resource ownership Privileges.

When CIM Servers are cascaded, it is necessary to be able to associate the embedded CIM client (e.g., in a virtualizer or NAS head) with the Identity in the array that is the AuthorizedSubject of the privileges. Assuming shared secrets, this can be modeled and realized as follows:

- In the virtualizer or NAS Head, a CIM Service instance is associated (ServiceSAPDependency) with a RemoteServiceAccessPoint that has associated via CredentialContext to a SharedSecret credential that contains information necessary for authentication.
 - RemoteID: String by which the principal is known. This maps to Account.UserID.
 - Secret: Password or other secret. This is set, but is not typically readable.
- In the array, the Identity instance is created that is authenticated by the Credential in the previous step. This Identity may be associated via ConcreteIdentity to an Account. Or, it may be associated via IdentityContext to a RemoteServiceAccessPoint that provides access to a 3rd Party Authentication service. If the latter, then the Security 3rdPartyAuthentication Subprofile shall be present on the Array.
- When the CIM client uses HTTP authentication with that username and password, the authenticated Identity is assigned to the CIM client's session.

There is no requirement that the Identity and Account instances in the array be creatable or manipulable via CIM. The contents of these instances have significant security implications and hence the ability to create and change them need to be carefully controlled. This example uses HTTP authentication, but is not meant to exclude other forms of authentication.

DEPRECATED

IECNORM.COM : Click to view the full PDF of ISO/IEC 24775-4:2014

IMPLEMENTED

23 Storage Virtualizer Profile

23.1 Description

23.1.1 General

Storage virtualizers act like RAID arrays but can use storage provided by systems external to the storage virtualizer and local disks. A storage virtualizer system combines both remote and local storage to create a seamless pool. The virtualization system allocates volumes from the pool for host systems to use.

The basic virtualizer system profile provides a read-only view of the system. The various subprofiles indicated in Figure 95: "Storage Virtualizer Package Diagram" extend this description and also enable configuration. Refer to 23.4 for more information on these optional extensions. This profile also includes the mandatory Clause 32 Physical Package Package (in *Information Technology – Storage Management, Part 3 Common Profiles*) that describes the physical layout of the system and includes product identification information. The modeling in this standard is split into various subclauses that describe how to model particular elements of a storage virtualizer system.

Figure 95: "Storage Virtualizer Package Diagram" illustrates the relationship between the packages related to the Storage Virtualizer Profile.

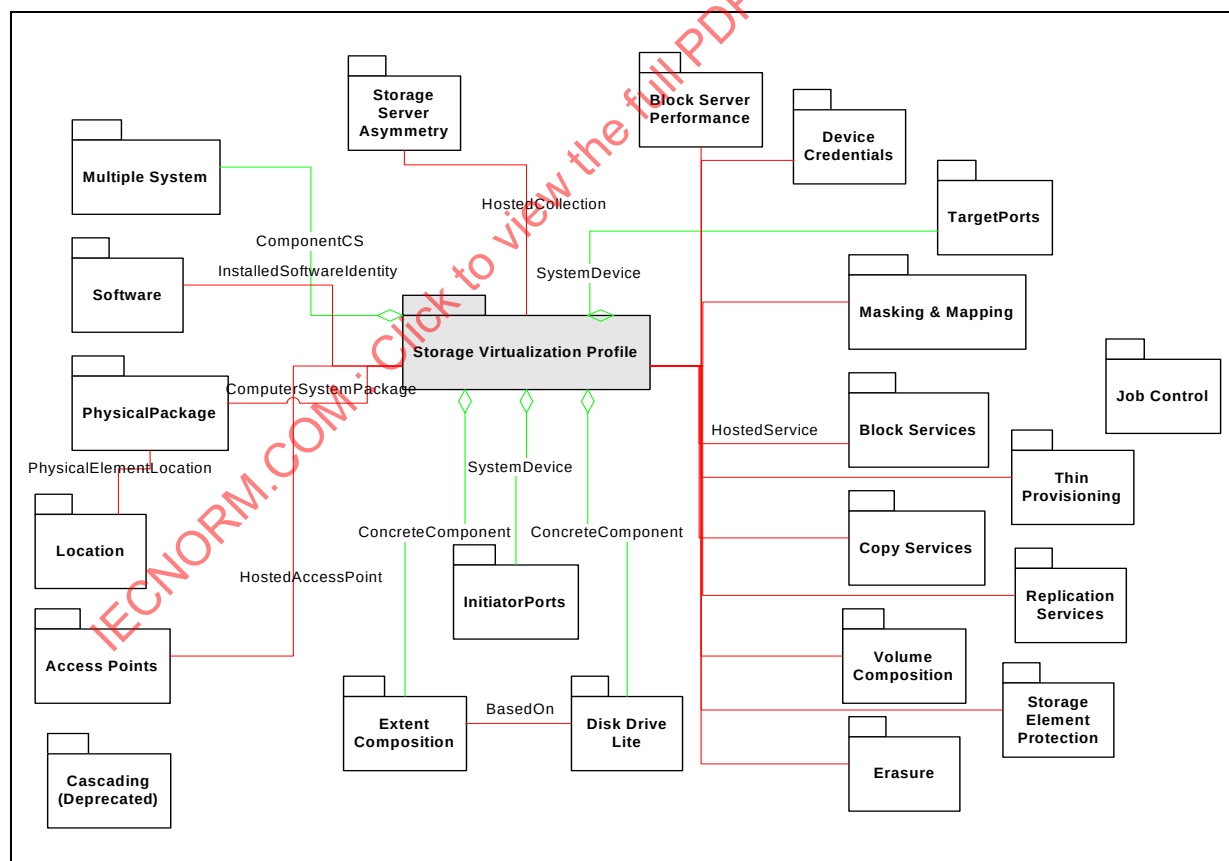


Figure 95 - Storage Virtualizer Package Diagram

23.1.2 Instance Diagram

The diagrams used in this standard 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. Refer to the DMTF MOF files for information on class inheritance information and full information on the properties and methods used.

Figure 96: "Storage Virtualizer System Instance" is an instance diagram of a simple Storage Virtualization system.

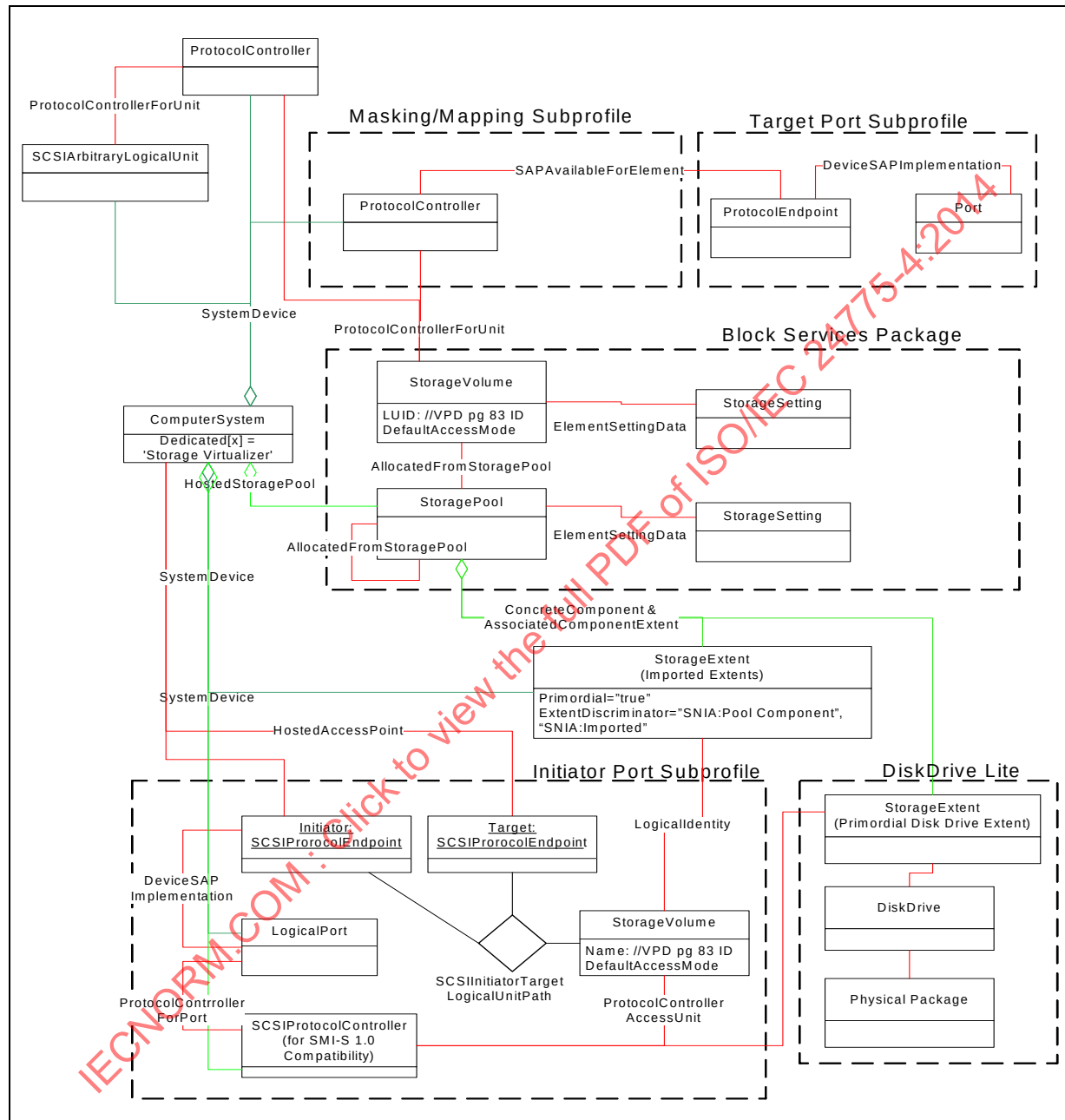


Figure 96 - Storage Virtualizer System Instance

23.1.3 Storage Virtualization System

The Virtualization system is modeled using the **ComputerSystem** class with the "Dedicated" properties set to 'BlockServer' and "StorageVirtualizer". 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 Multiple Computer System 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 hosted 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. The SCSIProtocolController with the ProtocolControllerAccessesUnit association to the StorageVolume are provided for clients convenience (and compatibility with SMI-S 1.0).

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 Port indicate ports the volume is mapped to. The StorageVolumes are described by the StorageSetting class connected by the ElementSettingData association.

23.1.4 Disk Drive Lite

The Disk Drive Lite Subprofile is optional. It should be used to model storage local to the storage virtualizer system. The Disk Drive Lite model includes a StorageExtent instance that represents the storage of the disk drive. If the Disk Drive Lite Profile is implemented, the StorageExtent shall be associated to a primordial pool. It may share a primordial pool with external storage or it can have its own primordial pool.

23.1.5 Controller Software

Information on the installed controller software is represented by the optional Software Subprofile. This is linked to the controller using an InstalledSoftwareIdentity association.

23.1.6 Device Management Access

Most devices now have a web GUI to allow device specific configuration. This is modeled using the common subprofile "Access Point".

23.1.7 Physical Modeling

The physical aspects of the storage virtualizer ComputerSystem are represented by the *Information Technology – Storage Management, Part 3 Common Profiles* Package Clause 32 "Physical Package Package" and the optional *Information Technology – Storage Management, Part 3 Common Profiles* Clause 28 "Location Subprofile", which provide more details.

23.1.8 Services

The system hosts services used to control the configuration of the system's resources. These services are optional and modeled by Clause 6 "Block Services Package", Clause 10 "Copy Services Subprofile", and ISO/IEC 24775-3:2014, clause 27 "Job Control Subprofile".

23.1.9 Ports

An implementation of the storage virtualizer shall implement at least one Target Ports Subprofile and may implement one or more of the Initiator Ports Subprofiles. However, this International Standard does not specify any particular port type to be supported. In either target or initiator cases, the ports could be FC or iSCSI. All port subprofiles are documented in *Information Technology – Storage Management, Part 3 Common Profiles*.

The storage virtualizer ConcreteComponent StorageExtent instances shown in the Initiator Ports Profile are the optional remote LogicalDevice instances from Initiator Ports. However, these StorageExtents are mandatory in the Storage Virtualizer Profile.

EXPERIMENTAL

23.1.10 Model Element Summary

This Profile defines the following CIM Classes (and their uses):

- ComputerSystem (Top Level System) - This is the top level ComputerSystem of the Storage Virtualizer, distinguished by the Dedicated Property of '15' and '21'.
- ComputerSystem (Shadow) - This is the ComputerSystem(s) to which the Storage Virtualizer cascades.

- `SCSIArbitraryLogicalUnit` - To represent a LUN address for receiving SCSI commands.
- `SCSIProtocolController` - To represent wide-open mapping of volumes (in the absence of the Masking and Mapping Profile).
- `StorageExtent (Imported Extents)` - Used to represent the volumes that have been imported from external devices.
- `StorageVolume (Shadow)` - Used to represent the volumes that are imported to the Storage Virtualizer.

EXPERIMENTAL

23.2 Health and Fault Management

Defined in the included subprofiles.

EXPERIMENTAL

23.3 Storage Virtualizer Support for Cascading

The Cascading Profile (see Clause 25 Cascading Subprofile in the *Information Technology – Storage Management, Part 3 Common Profiles*) has been deprecated in favor of embedding the cascading related classes in the Storage Virtualizer Profile. The classes identified in this subclause identify the elements of Storage Virtualizer support for the cascading function.

Figure 97: "Virtualizer, Cascading and Initiator Ports" shows the relationship between the Storage Virtualizer and the elements that support cascading of elements to other block server profiles. For example, cascading is required when the virtualizer imports logical units from arrays.

Each imported array is modeled in the virtualizer with a shadow `ComputerSystem`. The arrays' logical units are modeled using shadow `StorageVolume` instances. These are depicted in Figure 97: "Virtualizer, Cascading and Initiator Ports" in the box labeled "Cascading Support".

Each shadow `ComputerSystem` (representing an array) is associated to the Storage Virtualizer `ComputerSystem` using a Dependency association. `StorageVolume` models an Array logical unit and is associated to storage virtualizer `ConcreteComponent StorageExtent` via the `LogicalIdentity` association. The `StorageExtent` represents the virtualizer's view of logical units imported from arrays. The `StorageExtents` are local resources. The shadow `ComputerSystem` and `StorageVolumes` contain the correlatable IDs needed to map virtualizer resources to equivalent objects in an Array Profile.

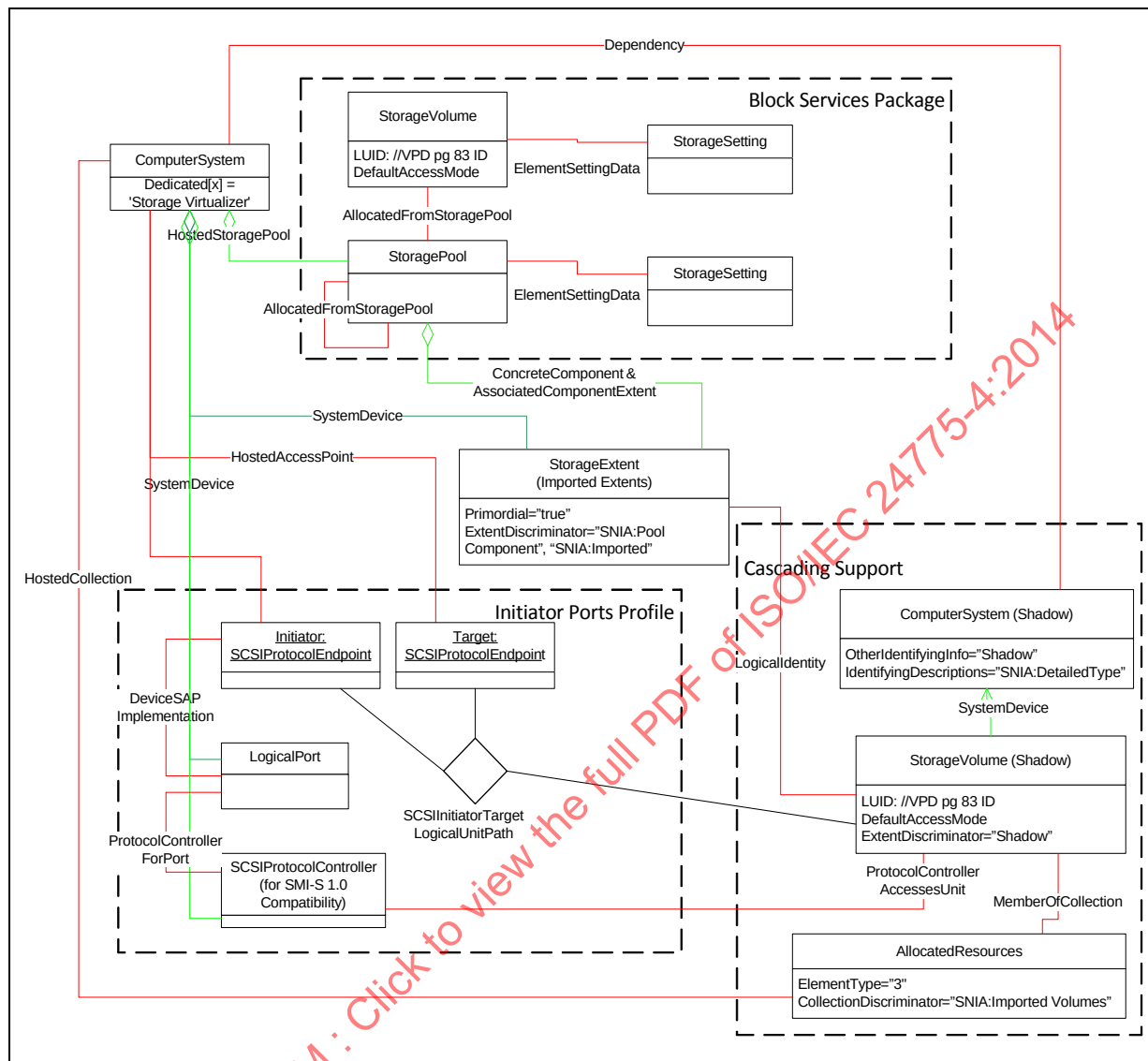


Figure 97 - Virtualizer, Cascading and Initiator Ports

The AllocatedResources collection identifies the shadow StorageVolumes that are actually allocated to the StorageVirtualizer for its use. Optionally, the implementation may also have a RemoteResources collection that identifies all the storage volumes it can see on the SAN.

EXPERIMENTAL

23.4 Supported Subprofiles and Packages

Table 396 describes the supported profiles for Storage Virtualizer.

Table 396 - Supported Profiles for Storage Virtualizer

Profile Name	Organization	Version	Requirement	Description
Access Points	SNIA	1.3.0	Optional	
Block Server Performance	SNIA	1.5.0	Optional	
Block Storage Views	SNIA	1.5.0	Optional	
CKD Block Services	SNIA	1.5.0	Optional	

Table 396 - Supported Profiles for Storage Virtualizer (Continued)

Profile Name	Organization	Version	Requirement	Description
Cluster	SNIA	1.0.2	Optional	
Extra Capacity Set	SNIA	1.0.2	Optional	
Disk Drive	SNIA	1.0.2	Optional	
Disk Drive Lite	SNIA	1.5.0	Optional	
Extent Mapping	SNIA	1.0.2	Optional	
Erasure	SNIA	1.2.0	Optional	
Storage Server Asymmetry	SNIA	1.4.0	Optional	
Volume Composition	SNIA	1.5.0	Optional	
Storage Element Protection	SNIA	1.4.0	Optional	
Copy Services	SNIA	1.5.0	Optional	
Pool Manipulation Capabilities and Settings	SNIA	1.0.2	Optional	
LUN Creation	SNIA	1.0.2	Optional	
Device Credentials	SNIA	1.3.0	Optional	
LUN Mapping and Masking	SNIA	1.0.2	Optional	
Job Control	SNIA	1.5.0	Optional	
Location	SNIA	1.4.0	Optional	
Masking and Mapping	SNIA	1.4.0	Optional	
Software	SNIA	1.4.0	Optional	
Multiple Computer System	SNIA	1.2.0	Optional	
Backend Ports	SNIA	1.0.2	Optional	
Disk Sparing	SNIA	1.5.0	Optional	
FC Initiator Ports	SNIA	1.4.0	Optional	
iSCSI Initiator Ports	SNIA	1.2.0	Optional	
SAS Initiator Ports	SNIA	1.4.0	Optional	
ATA Initiator Ports	SNIA	1.4.0	Optional	
Extent Composition	SNIA	1.5.0	Optional	
Cascading	SNIA	1.3.0	Mandatory	Deprecated. This is deprecated in favor of embedding cascading elements in the Storage Virtualizer profile.
Indication	SNIA	1.5.0	Mandatory	
Experimental Indication	SNIA	1.5.0	Optional	
Block Services	SNIA	1.5.0	Mandatory	
Physical Package	SNIA	1.5.0	Mandatory	
Health	SNIA	1.2.0	Mandatory	
Thin Provisioning	SNIA	1.5.0	Optional	
Replication Services	SNIA	1.5.0	Optional	

Table 396 - Supported Profiles for Storage Virtualizer (Continued)

Profile Name	Organization	Version	Requirement	Description
Operational Power	SNIA	1.5.0	Optional	Experimental.
Launch In Context	DMTF	1.0.0	Optional	Experimental. See DSP1102, version 1.0.0
iSCSI Target Ports	SNIA	1.2.0	Support for at least one is mandatory.	
FC Target Ports	SNIA	1.4.0		
SAS Target Ports	SNIA	1.4.0		
SB Target Ports	SNIA	1.2.0		
SB Initiator Ports	SNIA	1.4.0		

23.5 Methods of the Profile

None.

23.6 Client Considerations and Recipes

None.

23.7 Registered Name and Version

Storage Virtualizer version 1.5.0 (Autonomous Profile).

23.8 CIM Elements

23.8.1 General

Table 397 describes the CIM elements for Storage Virtualizer.

Table 397 - CIM Elements for Storage Virtualizer

Element Name	Requirement	Description
23.8.2 CIM_AssociatedComponentExtent (Pool Component to Primordial Pool)	Conditional	Conditional requirement: Implementation of the Extent Composition profile.
Experimental.	Mandatory	Experimental. 'Top level' system that represents a block storage device (e.g., an Array).
23.8.4 CIM_ComputerSystem (Top Level System)	Mandatory	'Top-level' system that represents the whole virtualizer. Associated to RegisteredProfile.
23.8.5 CIM_ConcreteComponent (Imported Extents to Primordial Pool)	Mandatory	Used to associate StorageExtents that are playing the Pool Component role to a Primordial StoragePool.
23.8.6 CIM_Dependency (Systems)	Mandatory	Experimental. This associates the block storage (e.g., Array) System to the Storage Virtualizer System.
23.8.7 CIM_FilterCollection (Storage Virtualizer Predefined FilterCollection)	Conditional	Experimental. Conditional requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='5' (Predefined Filter Collections). This is a collection of predefined IndicationFilters to which a client may subscribe.
23.8.8 CIM_HostedCollection (Allocated Resources)	Mandatory	Experimental. This would associate the AllocatedResources collection to the top level system for the Storage Virtualizer.

Table 397 - CIM Elements for Storage Virtualizer (Continued)

Element Name	Requirement	Description
23.8.9 CIM_HostedCollection (Remote Resources)	Conditional	Experimental. Conditional requirement: This is required if SNIA_RemoteResources is modeled. This would associate the RemoteResources collection to the top level system for the Storage Virtualizer.
23.8.10 CIM_HostedCollection (Storage Virtualizer to predefined FilterCollection)	Conditional	Experimental. Conditional requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='5' (Predefined Filter Collections).
23.8.11 CIM_IndicationFilter (Storage Virtualizer LogicalPort OperationalStatus)	Conditional	Deprecated. Conditional requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='3' (Predefined Filters). This is the 'pre-defined' CIM_IndicationFilter instance for changes in the OperationalStatus of LogicalPorts.
23.8.12 CIM_IndicationFilter (Storage Virtualizer Storage Volume OperationalStatus)	Conditional	Experimental. Conditional requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='3' (Predefined Filters). This is the 'pre-defined' CIM_IndicationFilter instance for changes in the OperationalStatus of StorageVolumes.
23.8.13 CIM_IndicationFilter (Storage Virtualizer System Creation)	Conditional	Experimental. Conditional requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='3' (Predefined Filters). This is the 'pre-defined' CIM_IndicationFilter instance for the addition of a new Storage Virtualizer system instance.
23.8.14 CIM_IndicationFilter (Storage Virtualizer System Deletion)	Conditional	Experimental. Conditional requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='3' (Predefined Filters). This is the 'pre-defined' CIM_IndicationFilter instance for the removal of a new Storage Virtualizer system instance.
23.8.15 CIM_IndicationFilter (Storage Virtualizer System OperationalStatus)	Conditional	Experimental. Conditional requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='3' (Predefined Filters). This is the 'pre-defined' CIM_IndicationFilter instance for changes in the OperationalStatus of a System.
23.8.16 CIM_IndicationFilter (WQL Storage Virtualizer FCPort OperationalStatus)	Optional	Deprecated. This is the 'pre-defined' WQL version of the CIM_IndicationFilter instance for changes in the OperationalStatus of FCPorts.
23.8.17 CIM_IndicationFilter (WQL Storage Virtualizer Storage Volume OperationalStatus)	Optional	Deprecated. This is the 'pre-defined' WQL version of the CIM_IndicationFilter instance for changes in the OperationalStatus of StorageVolumes.

Table 397 - CIM Elements for Storage Virtualizer (Continued)

Element Name	Requirement	Description
23.8.18 CIM_IndicationFilter (WQL Storage Virtualizer System OperationalStatus)	Optional	Deprecated. This is the 'pre-defined' WQL version of the CIM_IndicationFilter instance for changes in the OperationalStatus of a System.
23.8.19 CIM_LogicalIdentity (Shadow Storage Volume)	Mandatory	Experimental. Associates a Storage Virtualizer StorageExtent to a shadow instance of an (imported) StorageVolume.
23.8.20 CIM_MemberOfCollection (Allocated Resources)	Mandatory	Experimental. This supports collecting StorageVolumes. This is required to support the AllocatedResources collection.
23.8.21 CIM_MemberOfCollection (Predefined Filter Collection to Storage Virtualizer Filters)	Conditional	Experimental. Conditional requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='5' (Predefined Filter Collections). This associates the Storage Virtualizer predefined FilterCollection to the predefined Filters supported by the Storage Virtualizer.
23.8.22 CIM_MemberOfCollection (Remote Resources)	Optional	Experimental. This supports collecting all Shadow instances of StorageVolume that the Storage Virtualizer has available to use. This is optional when used to support the RemoteResources collection (the RemoteResources collection is optional).
23.8.23 CIM_ProtocolControllerForUnit (Arbitrary LU for All LUNs View)	Conditional	Conditional requirement: Elements that are mandatory if Masking and Mapping is not implemented.
23.8.24 CIM_ProtocolControllerForUnit (Storage volumes for All LUNs View)	Conditional	Conditional requirement: Elements that are mandatory if Masking and Mapping is not implemented.
23.8.25 CIM_RemoteServiceAccessPoint (Shadow)	Optional	Experimental. CIM_RemoteServiceAccessPoint represents the management interface to a Shadow system.
23.8.26 CIM_SAPAvailableForElement	Conditional	Experimental. Conditional requirement: This is required if CIM_RemoteServiceAccessPoint is modeled. Represents the association between a RemoteServiceAccessPoint and the Shadow (e.g., Array) System to which it provides access.
23.8.27 CIM_SCSIArbitraryLogicalUnit (Arbitrary LU)	Optional	A SCSI Logical Unit that exists only for management of the virtualizer.
23.8.28 CIM_SCSIProtocolController (All LUNs View)	Conditional	Conditional requirement: Elements that are mandatory if Masking and Mapping is not implemented.
23.8.29 CIM_StorageExtent (Imported Extents)	Mandatory	Used to represent the storage imported from external arrays and used as ConcreteComponents of Primordial StoragePools.
23.8.30 CIM_StorageVolume (Shadow)	Mandatory	Experimental. A shadow copy of a remote StorageVolume that is imported to the Storage Virtualizer.

Table 397 - CIM Elements for Storage Virtualizer (Continued)

Element Name	Requirement	Description
23.8.31 CIM_SystemDevice (Shadow StorageVolumes)	Mandatory	Experimental. This association links shadow StorageVolumes to the scoping (Shadow) system (of the array). This is used to associate the shadow StorageVolumes with the System that manages them.
23.8.32 CIM_SystemDevice (System to SCSIArbitraryLogicalUnit)	Conditional	Conditional requirement: Elements that are mandatory if SCSIArbitraryLogicalUnit is instantiated. This association links SCSIArbitraryLogicalUnit to the scoping system.
23.8.33 CIM_SystemDevice (System to SCSIProtocolController)	Conditional	Conditional requirement: Elements that are mandatory if Masking and Mapping is not implemented. This association links SCSIProtocolController to the scoping system.
23.8.34 CIM_SystemDevice (System to StorageExtent)	Mandatory	This association links the primordial imported StorageExtent to the scoping system.
23.8.35 SNIA_AllocatedResources	Mandatory	Experimental. This is a SystemSpecificCollection for collecting StorageVolumes that are being used by the Storage Virtualizer (e.g., StorageVolumes that the Virtualizer is using as Imported Primordial Extents).
23.8.36 SNIA_RemoteResources	Optional	Experimental. This is a SystemSpecificCollection for collecting StorageVolumes that may be allocated by the Storage Virtualizer profile (e.g., StorageVolumes that may be allocated to support a Storage Virtualizer primordial storage pool).
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_ComputerSystem	Mandatory	Creation of a ComputerSystem instance. See 23.8.13 CIM_IndicationFilter (Storage Virtualizer System Creation).
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_ComputerSystem	Mandatory	Deletion of a ComputerSystem instance. See 23.8.14 CIM_IndicationFilter (Storage Virtualizer System Deletion).
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_StorageVolume AND SourceInstance.OperationalStatus <> PreviousInstance.OperationalStatus	Optional	Deprecated. Deprecated WQL - Modification of OperationalStatus of a Storage Volume instance, provided for backward compatibility with In-band Virtualization. See 23.8.17 CIM_IndicationFilter (WQL Storage Virtualizer Storage Volume OperationalStatus).
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_StorageVolume AND SourceInstance.CIM_StorageVolume::OperationalStatus <> PreviousInstance.CIM_StorageVolume::OperationalStatus	Mandatory	CQL - Modification of OperationalStatus of a Storage Volume instance. See 23.8.12 CIM_IndicationFilter (Storage Virtualizer Storage Volume OperationalStatus).
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_FCPort AND SourceInstance.OperationalStatus <> PreviousInstance.OperationalStatus	Optional	Deprecated. Deprecated WQL - Modification of OperationalStatus of a FC port instance, provided for backward compatibility with In-band Virtualization. See 23.8.16 CIM_IndicationFilter (WQL Storage Virtualizer FCPort OperationalStatus).

Table 397 - CIM Elements for Storage Virtualizer (Continued)

Element Name	Requirement	Description
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_LogicalPort AND SourceInstance.CIM_LogicalPort::OperationalStatus <> PreviousInstance.CIM_LogicalPort::OperationalStatus	Mandatory	Deprecated. CQL - Modification of OperationalStatus of a Logical (FC or Ethernet) port instance. See 23.8.11 CIM_IndicationFilter (Storage Virtualizer LogicalPort OperationalStatus).
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_ComputerSystem AND SourceInstance.OperationalStatus <> PreviousInstance.OperationalStatus	Optional	Deprecated. Deprecated WQL - Modification of OperationalStatus of a ComputerSystem instance, provided for backward compatibility with In-band Virtualization. See 23.8.18 CIM_IndicationFilter (WQL Storage Virtualizer System OperationalStatus).
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_ComputerSystem AND SourceInstance.CIM_ComputerSystem::OperationalStatus <> PreviousInstance.CIM_ComputerSystem::OperationalStatus	Mandatory	CQL - Modification of OperationalStatus of a ComputerSystem instance. See 23.8.15 CIM_IndicationFilter (Storage Virtualizer System OperationalStatus).

23.8.2 CIM_AssociatedComponentExtent (Pool Component to Primordial Pool)

The referenced primordial imported StorageExtent indicates capacity has not been allocated, is allocated in part, or is allocated in its entirety.

Requirement: Implementation of the Extent Composition profile.

Table 398 describes class CIM_AssociatedComponentExtent (Pool Component to Primordial Pool).

Table 398 - SMI Referenced Properties/Methods for CIM_AssociatedComponentExtent (Pool Component to Primordial Pool)

Properties	Flags	Requirement	Description and Notes
GroupComponent		Mandatory	The Primordial StoragePool.
PartComponent		Mandatory	The imported storage extent that is a component of the primordial storage pool.

23.8.3 CIM_ComputerSystem (Shadow)

Experimental

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 399 describes class CIM_ComputerSystem (Shadow).

Table 399 - SMI Referenced Properties/Methods for CIM_ComputerSystem (Shadow)

Properties	Flags	Requirement	Description and Notes
CreationClassName		Mandatory	
Name		Mandatory	Unique identifier for the shadow system. E.g., IP address.
ElementName		Mandatory	User friendly name.

Table 399 - SMI Referenced Properties/Methods for CIM_ComputerSystem (Shadow) (Contin-

Properties	Flags	Requirement	Description and Notes
OtherIdentifyingInfo	C	Mandatory	At least one of the indices of this array should contain any of the valid system name formats. Another index should contain the string 'Shadow'.
IdentifyingDescriptions	C	Mandatory	For system names this array property should contain the NameFormat of the system name (e.g., 'Ipv4 Address' (Internet Protocol version 4) if the OtherIdentifyingInfo is an IPv4 address). In the index for the OtherIdentifyingInfo string 'Shadow' the IdentifyingDescriptions entry should be 'SNIA:DetailedType'.
OperationalStatus		Mandatory	Overall status of the shadow system, as seen by the Storage Virtualizer.
NameFormat		Mandatory	Format for Name property.
Dedicated		Mandatory	Indicates that this computer system is dedicated to operation as a block storage system (e.g., an Array).
PrimaryOwnerContact	M	Optional	Contact details for owner.
PrimaryOwnerName	M	Optional	Owner of the shadow system.

23.8.4 CIM_ComputerSystem (Top Level System)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Shall be associated to RegisteredProfile using ElementConformsToProfile association. The RegisteredProfile instance shall have RegisteredName set to 'Storage Virtualizer', RegisteredOrganization set to 'SNIA', and RegisteredVersion set to '1.5.0'.

Table 400 describes class CIM_ComputerSystem (Top Level System).

Table 400 - SMI Referenced Properties/Methods for CIM_ComputerSystem (Top Level System)

Properties	Flags	Requirement	Description and Notes
CreationClassName		Mandatory	
Name	C	Mandatory	Unique identifier for the storage virtualizer. E.g. IP address or a FC WWN.
ElementName		Mandatory	User friendly name.
OtherIdentifyingInfo	C	Mandatory	
IdentifyingDescriptions	C	Mandatory	
OperationalStatus		Mandatory	Overall status of the storage virtualizer.
NameFormat		Mandatory	Format for Name property.
Dedicated		Mandatory	The values 15 and 21 indicate that this computer system is dedicated to operation as a storage virtualizer.
PrimaryOwnerContact	M	Optional	Contact details for owner.
PrimaryOwnerName	M	Optional	Owner of the storage virtualizer.

23.8.5 CIM_ConcreteComponent (Imported Extents to Primordial Pool)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 401 describes class CIM_ConcreteComponent (Imported Extents to Primordial Pool).

Table 401 - SMI Referenced Properties/Methods for CIM_ConcreteComponent (Imported Extents to Primordial Pool)

Properties	Flags	Requirement	Description and Notes
GroupComponent		Mandatory	A Primordial StoragePool.
PartComponent		Mandatory	The imported StorageExtent.

23.8.6 CIM_Dependency (Systems)

Experimental. CIM_Dependency is an association between a shadow System (e.g., Array) and the Storage Virtualizer top level System (ComputerSystem). The specific nature of the dependency is determined by associations between resources (imported StorageExtents) of the Storage Virtualizer system and resources (StorageVolumes) of the shadow system.

CIM_Dependency is not subclassed from anything.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 402 describes class CIM_Dependency (Systems).

Table 402 - SMI Referenced Properties/Methods for CIM_Dependency (Systems)

Properties	Flags	Requirement	Description and Notes
Antecedent		Mandatory	The Storage Virtualizer top level System.
Dependent		Mandatory	The shadow System (e.g., system of the Array device).

23.8.7 CIM_FilterCollection (Storage Virtualizer Predefined FilterCollection)

Experimental. This is a collection of predefined IndicationFilters to which a client may subscribe. A Storage Virtualizer implementation shall indicate support for predefined FilterCollections by the SNIA_IndicationConfigurationCapabilities.FeaturesSupported = '5' (Predefined Filter Collections).

Requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='5' (Predefined Filter Collections).

Table 403 describes class CIM_FilterCollection (Storage Virtualizer Predefined FilterCollection).

**Table 403 - SMI Referenced Properties/Methods for CIM_FilterCollection
(Storage Virtualizer Predefined FilterCollection)**

Properties	Flags	Requirement	Description and Notes
InstanceID		Mandatory	Shall specify the unique identifier for an instance of this class within the Implementation namespace.
CollectionName		Mandatory	The value of CollectionName shall be 'SNIA:Storage Virtualizer'.

23.8.8 CIM_HostedCollection (Allocated Resources)

Experimental. CIM_HostedCollection defines a SystemSpecificCollection in the context of a scoping 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. In the Storage Virtualizer profile, it is used to associate the Allocated Resources to the top level Computer System of the Storage Virtualizer.

cim_hostedcollection is subclassed from cim_hosteddependency.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 404 describes class CIM_HostedCollection (Allocated Resources).

**Table 404 - SMI Referenced Properties/Methods for CIM_HostedCollection
(Allocated Resources)**

Properties	Flags	Requirement	Description and Notes
Antecedent		Mandatory	The Top Level System of the Storage Virtualizer.
Dependent		Mandatory	The AllocatedResources collection of shadow storage volumes.

23.8.9 CIM_HostedCollection (Remote Resources)

Experimental. CIM_HostedCollection defines a SystemSpecificCollection in the context of a scoping 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. In the Storage Virtualizer Profile, it is used to associate the Remote Resources to the top level Computer System of the Storage Virtualizer.

CIM_HostedCollection is subclassed from CIM_HostedDependency.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: This is required if SNIA_RemoteResources is modeled.

Table 405 describes class CIM_HostedCollection (Remote Resources).

Table 405 - SMI Referenced Properties/Methods for CIM_HostedCollection (Remote Resources)

Properties	Flags	Requirement	Description and Notes
Antecedent		Mandatory	The Top Level System of the Storage Virtualizer.
Dependent		Mandatory	The RemoteResources collection of shadow storage volumes.

23.8.10 CIM_HostedCollection (Storage Virtualizer to predefined FilterCollection)

Experimental.

Requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='5' (Predefined Filter Collections).

Table 406 describes class CIM_HostedCollection (Storage Virtualizer to predefined FilterCollection).

Table 406 - SMI Referenced Properties/Methods for CIM_HostedCollection (Storage Virtualizer to predefined FilterCollection)

Properties	Flags	Requirement	Description and Notes
Dependent		Mandatory	Reference to the predefined FilterCollection for the Storage Virtualizer.
Antecedent		Mandatory	Reference to the 'Top level' Storage Virtualizer System.

23.8.11 CIM_IndicationFilter (Storage Virtualizer LogicalPort OperationalStatus)

Deprecated. This is the 'pre-defined' CIM_IndicationFilter instance for changes in the OperationalStatus of LogicalPorts.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='3' (Predefined Filters).

Table 407 describes class CIM_IndicationFilter (Storage Virtualizer LogicalPort OperationalStatus).

Table 407 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Storage Virtualizer LogicalPort OperationalStatus)

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
CreationClassName		Mandatory	See the CreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
SystemName		Mandatory	See the SystemName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).

**Table 407 - SMI Referenced Properties/Methods for CIM_IndicationFilter
(Storage Virtualizer LogicalPort OperationalStatus) (Continued)**

Properties	Flags	Requirement	Description and Notes
Name		Mandatory	This shall be 'SNIA:Storage Virtualizer:LogicalPortOperationalStatus'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
Query		Mandatory	SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_LogicalPort AND SourceInstance.CIM_LogicalPort::OperationalStatus <> PreviousInstance.CIM_LogicalPort::OperationalStatus .
QueryLanguage		Mandatory	This shall be 'DMTF:CQL'.
ElementName	N	Optional	See the ElementName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).

23.8.12 CIM_IndicationFilter (Storage Virtualizer Storage Volume OperationalStatus)

Experimental. This is the 'pre-defined' CIM_IndicationFilter instance for changes in the OperationalStatus of StorageVolumes.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='3' (Predefined Filters).

Table 408 describes class CIM_IndicationFilter (Storage Virtualizer Storage Volume OperationalStatus).

**Table 408 - SMI Referenced Properties/Methods for CIM_IndicationFilter
(Storage Virtualizer Storage Volume OperationalStatus)**

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
CreationClassName		Mandatory	See the CreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
SystemName		Mandatory	See the SystemName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
Name		Mandatory	This shall be 'SNIA:Storage Virtualizer:StorageVolumeOperationalStatus'.

**Table 408 - SMI Referenced Properties/Methods for CIM_IndicationFilter
(Storage Virtualizer Storage Volume OperationalStatus) (Continued)**

Properties	Flags	Requirement	Description and Notes
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
Query		Mandatory	SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_StorageVolume AND SourceInstance.CIM_StorageVolume::OperationalStatus <> PreviousInstance.CIM_StorageVolume::OperationalStatus.
QueryLanguage		Mandatory	This shall be 'DMTF:COL'.
ElementName	N	Optional	See the ElementName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).

23.8.13 CIM_IndicationFilter (Storage Virtualizer System Creation)

Experimental. This is the 'pre-defined' CIM_IndicationFilter instance for the addition of a new Storage Virtualizer system instance. This would represent the addition of a controller computer system to the Storage Virtualizer.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='3' (Predefined Filters).

Table 409 describes class CIM_IndicationFilter (Storage Virtualizer System Creation).

**Table 409 - SMI Referenced Properties/Methods for CIM_IndicationFilter
(Storage Virtualizer System Creation)**

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
CreationClassName		Mandatory	See the CreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
SystemName		Mandatory	See the SystemName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
Name		Mandatory	This shall be 'SNIA:Storage Virtualizer:SystemCreation'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).

**Table 409 - SMI Referenced Properties/Methods for CIM_IndicationFilter
(Storage Virtualizer System Creation) (Continued)**

Properties	Flags	Requirement	Description and Notes
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
Query		Mandatory	SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_ComputerSystem.
QueryLanguage		Mandatory	This should be 'DMTF:CQL' for CQL queries, but may be 'WQL' or 'SMI-S V1.0'. WQL and SMI-S V1.0 are deprecated in favor of 'DMTF:CQL'.
ElementName	N	Optional	See the ElementName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).

23.8.14 CIM_IndicationFilter (Storage Virtualizer System Deletion)

Experimental. This is the 'pre-defined' CIM_IndicationFilter instance for the removal of a new Storage Virtualizer system instance. This would represent the removal of a controller computer system from the Storage Virtualizer.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='3' (Predefined Filters).

Table 410 describes class CIM_IndicationFilter (Storage Virtualizer System Deletion).

**Table 410 - SMI Referenced Properties/Methods for CIM_IndicationFilter
(Storage Virtualizer System Deletion)**

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
CreationClassName		Mandatory	See the CreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
SystemName		Mandatory	See the SystemName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
Name		Mandatory	This shall be 'SNIA:Storage Virtualizer:SystemDeletion'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).

Table 410 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Storage Virtualizer System Deletion) (Continued)

Properties	Flags	Requirement	Description and Notes
Query		Mandatory	SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_ComputerSystem.
QueryLanguage		Mandatory	This should be 'DMTF:CQL' for CQL queries, but may be 'WQL' or 'SMI-S V1.0'. WQL and SMI-S V1.0 are deprecated in favor of 'DMTF:CQL'.
ElementName	N	Optional	See the ElementName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).

23.8.15 CIM_IndicationFilter (Storage Virtualizer System OperationalStatus)

Experimental. This is the 'pre-defined' CIM_IndicationFilter instance for changes in the OperationalStatus of a System.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='3' (Predefined Filters).

Table 411 describes class CIM_IndicationFilter (Storage Virtualizer System OperationalStatus).

Table 411 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Storage Virtualizer System OperationalStatus)

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
CreationClassName		Mandatory	See the CreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
SystemName		Mandatory	See the SystemName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
Name		Mandatory	This shall be 'SNIA:Storage Virtualizer:SystemOperationalStatus'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).

**Table 411 - SMI Referenced Properties/Methods for CIM_IndicationFilter
(Storage Virtualizer System OperationalStatus) (Continued)**

Properties	Flags	Requirement	Description and Notes
Query		Mandatory	SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_ComputerSystem AND SourceInstance.CIM_ComputerSystem::OperationalStatus <> PreviousInstance.CIM_ComputerSystem::OperationalStatus.
QueryLanguage		Mandatory	This shall be 'DMTF:CQL'.
ElementName	N	Optional	See the ElementName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).

23.8.16 CIM_IndicationFilter (WQL Storage Virtualizer FCPort OperationalStatus)

Deprecated. This is the 'pre-defined' WQL version of the CIM_IndicationFilter instance for changes in the OperationalStatus of FCPorts.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 412 describes class CIM_IndicationFilter (WQL Storage Virtualizer FCPort OperationalStatus).

**Table 412 - SMI Referenced Properties/Methods for CIM_IndicationFilter
(WQL Storage Virtualizer FCPort OperationalStatus)**

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
CreationClassName		Mandatory	See the CreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
SystemName		Mandatory	See the SystemName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
Name		Mandatory	This shall be 'SNIA:Storage Virtualizer:FCPortOperationalStatusWQL'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).

**Table 412 - SMI Referenced Properties/Methods for CIM_IndicationFilter
(WQL Storage Virtualizer FCPort OperationalStatus) (Continued)**

Properties	Flags	Requirement	Description and Notes
Query		Mandatory	SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_FCPort AND SourceInstance.OperationalStatus <> PreviousInstance.OperationalStatus.
QueryLanguage		Mandatory	This shall be 'WQL' or 'SMI-S V1.0'.
ElementName	N	Optional	See the ElementName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).

23.8.17 CIM_IndicationFilter (WQL Storage Virtualizer Storage Volume OperationalStatus)

Deprecated. This is the 'pre-defined' WQL version of the CIM_IndicationFilter instance for changes in the OperationalStatus of StorageVolumes.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 413 describes class CIM_IndicationFilter (WQL Storage Virtualizer Storage Volume OperationalStatus).

**Table 413 - SMI Referenced Properties/Methods for CIM_IndicationFilter
(WQL Storage Virtualizer Storage Volume OperationalStatus)**

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
CreationClassName		Mandatory	See the CreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
SystemName		Mandatory	See the SystemName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
Name		Mandatory	This shall be 'SNIA:Storage Virtualizer:StorageVolumeOperationalStatusWQL'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in <i>Information Technology – Storage Management, Part 3 Common Profiles 43.8.4 CIM_IndicationFilter</i> (pre-defined).

Table 413 - SMI Referenced Properties/Methods for CIM_IndicationFilter (WQL Storage Virtualizer Storage Volume OperationalStatus) (Continued)

Properties	Flags	Requirement	Description and Notes
Query		Mandatory	SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_StorageVolume AND SourceInstance.OperationalStatus <> PreviousInstance.OperationalStatus.
QueryLanguage		Mandatory	This shall be 'WQL' or 'SMI-S V1.0'.
ElementName	N	Optional	See the ElementName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).

23.8.18 CIM_IndicationFilter (WQL Storage Virtualizer System OperationalStatus)

Deprecated. This is the 'pre-defined' WQL version of the CIM_IndicationFilter instance for changes in the OperationalStatus of a System.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 414 describes class CIM_IndicationFilter (WQL Storage Virtualizer System OperationalStatus).

Table 414 - SMI Referenced Properties/Methods for CIM_IndicationFilter (WQL Storage Virtualizer System OperationalStatus)

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
CreationClassName		Mandatory	See the CreationClassName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
SystemName		Mandatory	See the SystemName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
Name		Mandatory	This shall be 'SNIA:Storage Virtualizer:SystemOperationalStatusWQL'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).

Table 414 - SMI Referenced Properties/Methods for CIM_IndicationFilter (WQL Storage Virtualizer System OperationalStatus) (Continued)

Properties	Flags	Requirement	Description and Notes
Query		Mandatory	SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_ComputerSystem AND SourceInstance.OperationalStatus <> PreviousInstance.OperationalStatus.
QueryLanguage		Mandatory	This shall be 'WQL' or 'SMI-S V1.0'.
ElementName	N	Optional	See the ElementName definition in <i>Information Technology – Storage Management, Part 3 Common Profiles</i> 43.8.4 CIM_IndicationFilter (pre-defined).

23.8.19 CIM_LogicalIdentity (Shadow Storage Volume)

Experimental. Associates local StorageExtent to a shadow instance of an (imported) StorageVolume.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 415 describes class CIM_LogicalIdentity (Shadow Storage Volume).

Table 415 - SMI Referenced Properties/Methods for CIM_LogicalIdentity (Shadow Storage Volume)

Properties	Flags	Requirement	Description and Notes
SystemElement		Mandatory	This is a reference to the shadow (imported) StorageVolume.
SameElement		Mandatory	This is a reference to the Storage Virtualizer StorageExtent that maps to the shadow (imported) StorageVolume.

23.8.20 CIM_MemberOfCollection (Allocated Resources)

Experimental. This use of MemberOfCollection is to collect all allocated shadow StorageVolume instances (in the AllocatedResources collection).

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 416 describes class CIM_MemberOfCollection (Allocated Resources).

Table 416 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Allocated Resources)

Properties	Flags	Requirement	Description and Notes
Member		Mandatory	A shadow storage volume (one with ExtentDiscriminator='SNIA:Shadow').
Collection		Mandatory	The AllocatedResources collection of shadow storage volumes.

23.8.21 CIM_MemberOfCollection (Predefined Filter Collection to Storage Virtualizer Filters)

Experimental. This associates the Storage Virtualizer predefined FilterCollection to the predefined Filters supported by the Storage Virtualizer.

Requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='5' (Predefined Filter Collections).

Table 417 describes class CIM_MemberOfCollection (Predefined Filter Collection to Storage Virtualizer Filters).

Table 417 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Predefined Filter Collection to Storage Virtualizer Filters)

Properties	Flags	Requirement	Description and Notes
Collection		Mandatory	Reference to the Storage Virtualizer predefined FilterCollection.
Member		Mandatory	Reference to the predefined IndicationFilters of the Storage Virtualizer.

23.8.22 CIM_MemberOfCollection (Remote Resources)

Experimental. This use of MemberOfCollection is to collect all shadow StorageVolume instances (in the RemoteResources collection). Each association (and the RemoteResources collection, itself) is created through external means.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 418 describes class CIM_MemberOfCollection (Remote Resources).

Table 418 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Remote Resources)

Properties	Flags	Requirement	Description and Notes
Member		Mandatory	A shadow storage volume (one with ExtentDiscriminator='SNIA:Shadow').
Collection		Mandatory	The RemoteResources collection of shadow storage volumes.

23.8.23 CIM_ProtocolControllerForUnit (Arbitrary LU for All LUNs View)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Elements that are mandatory if Masking and Mapping is not implemented.

Table 419 describes class CIM_ProtocolControllerForUnit (Arbitrary LU for All LUNs View).

Table 419 - SMI Referenced Properties/Methods for CIM_ProtocolControllerForUnit (Arbitrary LU for All LUNs View)

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

23.8.24 CIM_ProtocolControllerForUnit (Storage volumes for All LUNs View)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Elements that are mandatory if Masking and Mapping is not implemented.

Table 420 describes class CIM_ProtocolControllerForUnit (Storage volumes for All LUNs View).

Table 420 - SMI Referenced Properties/Methods for CIM_ProtocolControllerForUnit (Storage volumes for All LUNs View)

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

23.8.25 CIM_RemoteServiceAccessPoint (Shadow)

Experimental. CIM_RemoteServiceAccessPoint is an instance that provides access information for accessing the actual Shadow (e.g., Array) system via a management interface.

CIM_RemoteServiceAccessPoint is not subclassed from CIM_ServiceAccessPoint.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 421 describes class CIM_RemoteServiceAccessPoint (Shadow).

Table 421 - SMI Referenced Properties/Methods for CIM_RemoteServiceAccessPoint (Shadow)

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	The CIM Class name of the Computer System hosting the management interface.
SystemName		Mandatory	The name of the Computer System hosting the management interface.
CreationClassName		Mandatory	The CIM Class name of the management interface.
Name		Mandatory	The unique name of the management interface.

23.8.26 CIM_SAPAvailableForElement

Experimental.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: This is required if CIM_RemoteServiceAccessPoint is modeled.

Table 422 describes class CIM_SAPAvailableForElement.

Table 422 - SMI Referenced Properties/Methods for CIM_SAPAvailableForElement

Properties	Flags	Requirement	Description and Notes
ManagedElement		Mandatory	Shadow System.
AvailableSAP		Mandatory	The service access point of the shadow system.

23.8.27 CIM_SCSIArbitraryLogicalUnit (Arbitrary LU)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 423 describes class CIM_SCSIArbitraryLogicalUnit (Arbitrary LU).

Table 423 - SMI Referenced Properties/Methods for CIM_SCSIArbitraryLogicalUnit (Arbitrary LU)

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	Opaque identifier.
ElementName		Mandatory	User-friendly name.
Name		Mandatory	
OperationalStatus		Mandatory	

23.8.28 CIM_SCSIProtocolController (All LUNs View)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Elements that are mandatory if Masking and Mapping is not implemented.

Table 424 describes class CIM_SCSIProtocolController (All LUNs View).

Table 424 - SMI Referenced Properties/Methods for CIM_SCSIProtocolController (All LUNs View)

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

23.8.29 CIM_StorageExtent (Imported Extents)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 425 describes class CIM_StorageExtent (Imported Extents).

Table 425 - SMI Referenced Properties/Methods for CIM_StorageExtent (Imported Extents)

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	
BlockSize		Mandatory	
NumberOfBlocks		Mandatory	
ExtentStatus		Mandatory	
OperationalStatus		Mandatory	
Primordial		Mandatory	This shall be true for extents instantiated in the Storage Virtualizer.
ExtentDiscriminator		Mandatory	Experimental. This is an array of values that shall contain 'SNIA:Pool Component' and 'SNIA:Imported'.

23.8.30 CIM_StorageVolume (Shadow)

Experimental. A shadow copy of a remote StorageVolume that is imported to the Storage Virtualizer. If the Storage Virtualizer has access to the leaf profile, the data in this class should reflect what the Storage Virtualizer obtains from that profile. If the referencing profile does not have access to the leaf profile, then this should be filled out as best can be done.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 426 describes class CIM_StorageVolume (Shadow).

Table 426 - SMI Referenced Properties/Methods for CIM_StorageVolume (Shadow)

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	Opaque identifier.
ElementName		Optional	User-friendly name.
Name	CD	Mandatory	The identifier for this volume. If the Storage Virtualizer has access to the CIM Server for the device that exports the storage volume, then this should be the Name property as reported by the CIM Server. If the Storage Virtualizer does not have access to the CIM Server for the device, then it should be one of the names supported for storage volumes.
OtherIdentifyingInfo	CD	Optional	Additional correlatable names. Specific values should be values that may be correlated with the names reported by the device that exports the storage volume.
IdentifyingDescriptions		Optional	
NameFormat		Mandatory	The type of identifier in the Name property.
NameNamespace		Mandatory	The namespace that defines uniqueness for the NameFormat.
ExtentStatus		Mandatory	
OperationalStatus		Mandatory	Value shall be 2 3 6 8 15 (OK or Degraded or Error or Starting or Dormant).
BlockSize		Mandatory	
NumberOfBlocks		Mandatory	The number of blocks of capacity consumed from the parent StoragePool.
ConsumableBlocks		Mandatory	The number of blocks usable by consumers.
IsBasedOnUnderlyingRedundancy		Mandatory	
NoSinglePointOfFailure		Mandatory	
DataRedundancy		Mandatory	
PackageRedundancy		Mandatory	
DeltaReservation		Mandatory	
Usage		Optional	The specialized usage intended for this element.
OtherUsageDescription		Optional	Set when Usage value is "Other".
ClientSettableUsage		Optional	Lists Usage values that can be set by a client for this element.
ExtentDiscriminator		Mandatory	This shall be 'SNIA:Shadow'.

Table 426 - SMI Referenced Properties/Methods for CIM_StorageVolume (Shadow) (Continued)

Properties	Flags	Requirement	Description and Notes
Caption	N	Optional	Not Specified in this version of the Profile.
Description	N	Optional	Not Specified in this version of the Profile.
InstallDate	N	Optional	Not Specified in this version of the Profile.
StatusDescriptions	N	Optional	Not Specified in this version of the Profile.
HealthState	N	Optional	Not Specified in this version of the Profile.
EnabledState	N	Optional	Not Specified in this version of the Profile.
OtherEnabledState	N	Optional	Not Specified in this version of the Profile.
RequestedState	N	Optional	Not Specified in this version of the Profile.
EnabledDefault	N	Optional	Not Specified in this version of the Profile.
TimeOfLastStateChange	N	Optional	Not Specified in this version of the Profile.

23.8.31 CIM_SystemDevice (Shadow StorageVolumes)

Experimental.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 427 describes class CIM_SystemDevice (Shadow StorageVolumes).

Table 427 - SMI Referenced Properties/Methods for CIM_SystemDevice (Shadow StorageVolumes)

Properties	Flags	Requirement	Description and Notes
GroupComponent		Mandatory	The Shadow Computer System that contains this StorageVolume.
PartComponent		Mandatory	The storage volume that is managed by a computer system.

23.8.32 CIM_SystemDevice (System to SCSIArbitraryLogicalUnit)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Elements that are mandatory if SCSIArbitraryLogicalUnit is instantiated.

Table 428 describes class CIM_SystemDevice (System to SCSIArbitraryLogicalUnit).

Table 428 - SMI Referenced Properties/Methods for CIM_SystemDevice (System to SCSIArbitraryLogicalUnit)

Properties	Flags	Requirement	Description and Notes
PartComponent		Mandatory	
GroupComponent		Mandatory	

23.8.33 CIM_SystemDevice (System to SCSIProtocolController)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Elements that are mandatory if Masking and Mapping is not implemented.

Table 429 describes class CIM_SystemDevice (System to SCSIProtocolController).

**Table 429 - SMI Referenced Properties/Methods for CIM_SystemDevice
(System to SCSIProtocolController)**

Properties	Flags	Requirement	Description and Notes
PartComponent		Mandatory	
GroupComponent		Mandatory	

23.8.34 CIM_SystemDevice (System to StorageExtent)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 430 describes class CIM_SystemDevice (System to StorageExtent).

**Table 430 - SMI Referenced Properties/Methods for CIM_SystemDevice
(System to StorageExtent)**

Properties	Flags	Requirement	Description and Notes
PartComponent		Mandatory	The imported StorageExtent.
GroupComponent		Mandatory	The scoping ComputerSystem.

23.8.35 SNIA_AllocatedResources

Experimental. An instance of a default SNIA_AllocatedResources defines the set of StorageVolumes that are allocated and in use by the Storage Virtualizer.

SNIA_AllocatedResources is subclassed from CIM_SystemSpecificCollection.

At least one instance of the SNIA_AllocatedResources shall exist for a Storage Virtualizer Profile and shall be hosted by one of its ComputerSystems (typically the top level ComputerSystem).

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 431 describes class SNIA_AllocatedResources.

Table 431 - SMI Referenced Properties/Methods for SNIA_AllocatedResources

Properties	Flags	Requirement	Description and Notes
InstanceID		Mandatory	
ElementName		Mandatory	A user-friendly name for the AllocatedResources collection (e.g., Allocated StorageVolumes).
ElementType		Mandatory	The type of remote resources collected by the AllocatedResources collection. For this version of SMI-S, the only value supported is '3' (StorageVolume).
CollectionDiscriminator		Mandatory	An array of strings indicating the purposes of the collection of elements. This shall contain 'SNIA:Imported Volumes'.

23.8.36 SNIA_RemoteResources

Experimental. An instance of a default SNIA_RemoteResources defines the set of shadow StorageVolumes that are available to be used by the Storage Virtualizer.

SNIA_RemoteResources is subclassed from CIM_SystemSpecificCollection.

One instance of the SNIA_RemoteResources would exist and shall be hosted by the top level ComputerSystems of the Storage Virtualizer Profile.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional.

Table 432 describes class SNIA_RemoteResources.

Table 432 - SMI Referenced Properties/Methods for SNIA_RemoteResources

Properties	Flags	Requirement	Description and Notes
InstanceID		Mandatory	
ElementName		Mandatory	A user-friendly name for the RemoteResources collection (e.g., Remote Storage Volumes).
ElementType		Mandatory	The type of remote resources collected by the RemoteResources collection. This shall be '3' (StorageVolume).
CollectionDiscriminator		Mandatory	An array of strings indicating the purposes of the collection of elements. This shall contain 'SNIA:Imported Volumes'.

IMPLEMENTED

EXPERIMENTAL

24 Volume Composition Profile

24.1 Description

24.1.1 Overview

Some Arrays and Storage Virtualizers as well as Volume Managers have the ability to combine together existing storage volumes to make them appear to be one, bigger, volume. These are called composite volumes in the International Standard. This is different from the approach shown in the Block Services Package which shows how to create StorageExtents and StoragePools. This subprofile shows how to create StorageVolumes from volumes that are already allocated from the Storage Pool and exposed. These volumes may not necessarily be mapped to a port or masked to a host. These volumes can come from the same or different storage pools. Often the rules to determine which volumes can be combined with other volumes are quite complex and can vary even across a vendor's own product line. Once these elements are combined together, only one storage element is visible and the rest of the storage elements are hidden and cannot be exposed. When the composite storage element is dissolved, the hidden StorageVolumes reappear.

The Volume Composition Subprofile describes how instrumentation would combine exposable storage elements into other exposable storage elements. Storage Elements in this context are Storage Volumes or Logical Disks, although for this International Standard, only StorageVolumes are supported.

This subprofile introduces a number of new methods and capabilities. The existing methods in the StoragePool and StorageConfigurationService classes (CreateOrModifyElementFromStoragePool, CreateOrModifyElementFromElements, ReturnToStoragePool) were found to be inadequate or attached to the wrong class (i.e., StoragePool) to support the desired functionality. For this reason new methods with a composition-specific focus are introduced, instead of extending or overloading the usage of existing methods.

24.1.2 Relationship to Block Services Package

This profile makes use of the Block Services Package model and the applicable methods. Block Services shows how StorageExtents and StoragePools may be constructed from StoragePools and ultimately how StorageExtents may be exposed as a storage element (StorageVolume or LogicalDisk). This subprofile uses the StorageVolume, StorageExtent, and StoragePool classes in essentially the same ways as Block Services. This subprofile does not discuss how to create or delete StoragePools. It does maintain the concept that a StorageVolume is allocated from a StoragePool as shown by the AllocatedFromStoragePool association, although it does extend by allowing a StorageVolume to be allocated from multiple StoragePools. It also maintains the concept that a StorageVolume has a BasedOn association to an underlying StorageExtent. Because of this, the capacity calculations as defined in the Block Services Package shall continue to produce the correct results.

24.1.3 Relationship to Extent Composition

This profile is a component profile (or subprofile) and extends the functionality of the Extent Composition subprofile, which, in turn, references this profile as a supported profile. This profile requires the use of the Extent Composition Subprofile.

Extent Composition shows the hierarchical relationships between StorageVolumes and StorageExtents. This subprofile shows how to model composite storage elements (composite StorageVolumes). Extent Composition does not define any methods. This profile defines methods to perform composition and decomposition of composite StorageVolumes.

24.1.4 Model

To model these composite volumes, this subprofile shall define the use of CompositeExtent to represent the “composition” characteristics of the volume. A composite StorageVolume shall have a BasedOn association to the Antecedent CompositeExtent. That CompositeExtent shall have CompositeExtentBasedOn or BasedOn relationships to the underlying extents (from potentially multiple pools) that comprise the StorageVolume. These underlying extents could, in turn, be CompositeExtents.

If the volume is a composite from multiple pools, there shall be one AllocatedFromStoragePool association to each pool. SpaceConsumed shall show applicable space consumed from each pool. The general class model looks like Figure 98: “Volume Composition Class Mode”.

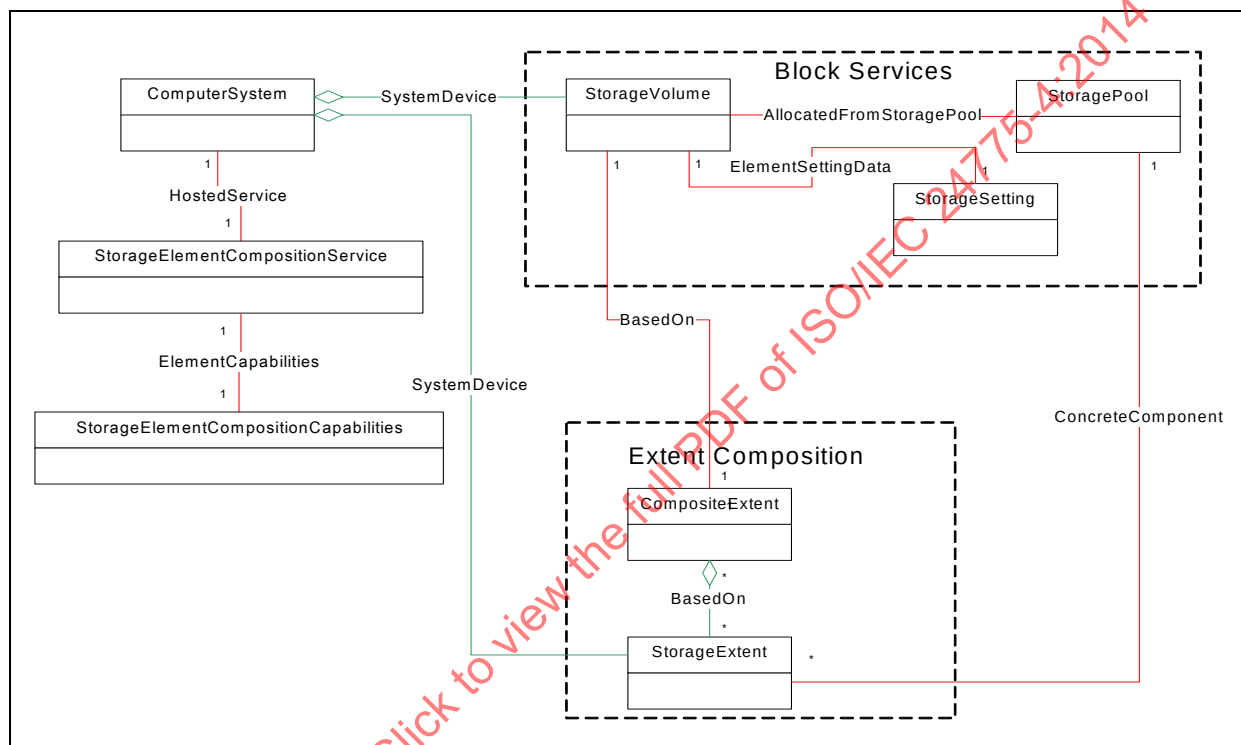


Figure 98 - Volume Composition Class Mode

One important thing to note about the class model is that the CompositeExtent is not associated via ConcreteComponent to the StoragePool.

The client can use the StorageElementCompositionCapabilities to determine which features of this profile are supported. The first property is to check is SupportsComposites, which will be set to true if the instrumentation supports creating and modifying composites. The client should also check MaxCompositeSize and MaxCompositeElements to determine the bounds for composite creation. Since there are a number of differences in the way vendors have implemented creation and modification, the client should check the CompositionCharacteristics array to understand which creation and modification options the instrumentation supports. The SupportedAsynchronousActions and SupportedSynchronousActions indicate which methods are supported and whether or not a job is started when the method is invoked. An entry in both arrays indicates a job may be started in some cases but not in others. SupportedStorageElements indicates the types of storage elements that may be used. For this International Standard, only StorageVolumes are supported. The CompositionMethodsSupported indicates which of the different ways of creating a composite (simple concatenation, striping across elements, concatenate and stripe, etc.) are supported by the instrumentation. Lastly, CompositeSourcesSupported is used to indicate the source of storage elements when they are not explicitly specified in the call to CreateOrModifyCompositeElement. The client can examine the CompositionCharacteristics

property to determine which options are permitted. See Table 433 for a summary of those possible values.

Table 433 - CompositionCharacteristics Property

Value	Description
CompositionIsDestructive	Any data that exists on the elements will be destroyed when the composite is created.
CanCompositeComposites	It is possible to use an existing composite as an element to a new composite.
CanModifyComposite	An existing composite can be modified by adding or removing one or more elements.
CompositeElementsMustBeSameSize	All elements used to create/modify a composite shall be the same size.
CompositeElementsMustBeSameRAID/ QoS	All elements used to create/modify a composite shall have the same RAID or QoS level.
DecompositionDeletesElements	When the composite is dissolved, the component elements (e.g. StorageVolumes) are deleted.
CanAddToComposite	Elements can be added to a composite in any position.
CanAppendToComposite	Elements can only be added at the end of a composite.
CanRemoveFromComposite	Elements can be removed from a composite.
CompositeAdditionIsDestructive	Adding elements to a composite results in loss of data.
CompositeRemovalIsDestructive	Removing elements from a composite results in loss of data.

24.1.5 Quality of Service (QoS) Considerations

It is a requirement of Block Services that each StorageVolume have an associated StorageSetting. This StorageSetting defines a requested 'service level' in terms of data and package redundancy. The currently achieved value is found in the StorageVolume itself.

When a composite is created, it shall have an associated StorageSetting as regular StorageVolumes do. It shall also track the current 'service level' achieved in the StorageVolume properties as specified by Block Services. However, the resulting 'service level' needs to be determined. Determining what this resulting 'service level' will depend upon the parameters passed in to CreateOrModifyCompositeElement. If only InElements is passed in, the 'service level' of the StorageVolume shall be determined by the instrumentation. If Goal or RepresentativeElement is passed in, the instrumentation shall attempt to meet the 'service level' specified by the Goal or RepresentativeElement instead of InElements (if InElements is non-NULL).

24.1.6 Composite Stripe Length and Depth

This profile supports the creation of composites where the elements are either concatenated together, striped, or concatenated and striped. To provide this information, this profile utilizes a StorageSetting that contains additional information about any striping done on the composite. StorageSetting.ExtentStripeLength describes the number of underlying storage elements in a composite volume that data is striped across. For any volumes not participating in the stripe, data is linearly written to the remaining volumes. This property only applies to composites that have a CompositeType of "Stripe elements" or "Concatenate and stripe elements". In the case of "Stripe elements", this value shall be equal to the number of elements in the composite. In the case of "Concatenate and stripe elements", ExtentStripeLength shall be equal to the number of striped elements and not the number of concatenated elements. In other words, for "Concatenate and stripe elements", ExtentStripeLength would be equivalent to the total number of volumes in the composite minus the number of concatenated elements.

The StorageSetting class also defines the UserDataStripeDepth property. This property defines the number bytes written to an individual striped volume in a composite volume before data is

written to the next volume in the stripe. This property only applies to Composite Volumes that have a CompositeType of "Stripe elements" or "Concatenate and stripe elements". Furthermore, for a composite volume there is no relationship between StorageSetting.ExtentStripeLength and StorageSetting.UserDataStripeDepth, which collectively with StorageSetting.ParityLayout describe the RAID level of storage elements. As an example, consider the case where you have a 4-volume composition with 3 striped and 1 concatenated volumes. In this example, UserDataStripeDepth bytes of data are written alternatively to the first 3 volumes until they fill up. Then all the writes go to the last volume.

The CompositeExtent properties are also affected by the stripe length. The CompositeExtent.ExtentStripeLength shall be set to 1 when the CompositeType is "Concatenate elements", n for "Stripe elements", and (n minus number of concatenated volumes) for "Concatenated and stripe elements"; where n is the number of members of a composite volume. CompositeExtent.IsConcatenated shall be set to true for CompositeType "Concatenate elements" and "Concatenated and stripe elements", false otherwise. PackageRedundancy shall be set to zero as there is no package redundancy in the CompositeExtent. IsBasedOnUnderlyingRedundancy shall be set to true if all of the composite volumes' IsBasedOnUnderlyingRedundancy property is set to true, false otherwise. NoSinglePointOfFailure shall be set to false as the CompositeExtent represents a single point of failure for the composite volume.

24.1.7 Examples

24.1.7.1 Example 1

Figure 99 shows how a composite volume may be created. For simplification, the value of the StorageExtent.BlockSize property is 1 and the associations to the underlying primordial StoragePool have been omitted, along with the StorageSettings associated to the volumes. In some implementations, there may be intermediate extents between the volume and the ConcreteComponent StorageExtent.

In this example, we have four StorageExtents of 40 blocks each that are combined into a concrete storage pool of 160 blocks and four storage volumes allocated from the pool, each consuming 40 blocks. The remaining space in the pool is 0 blocks.

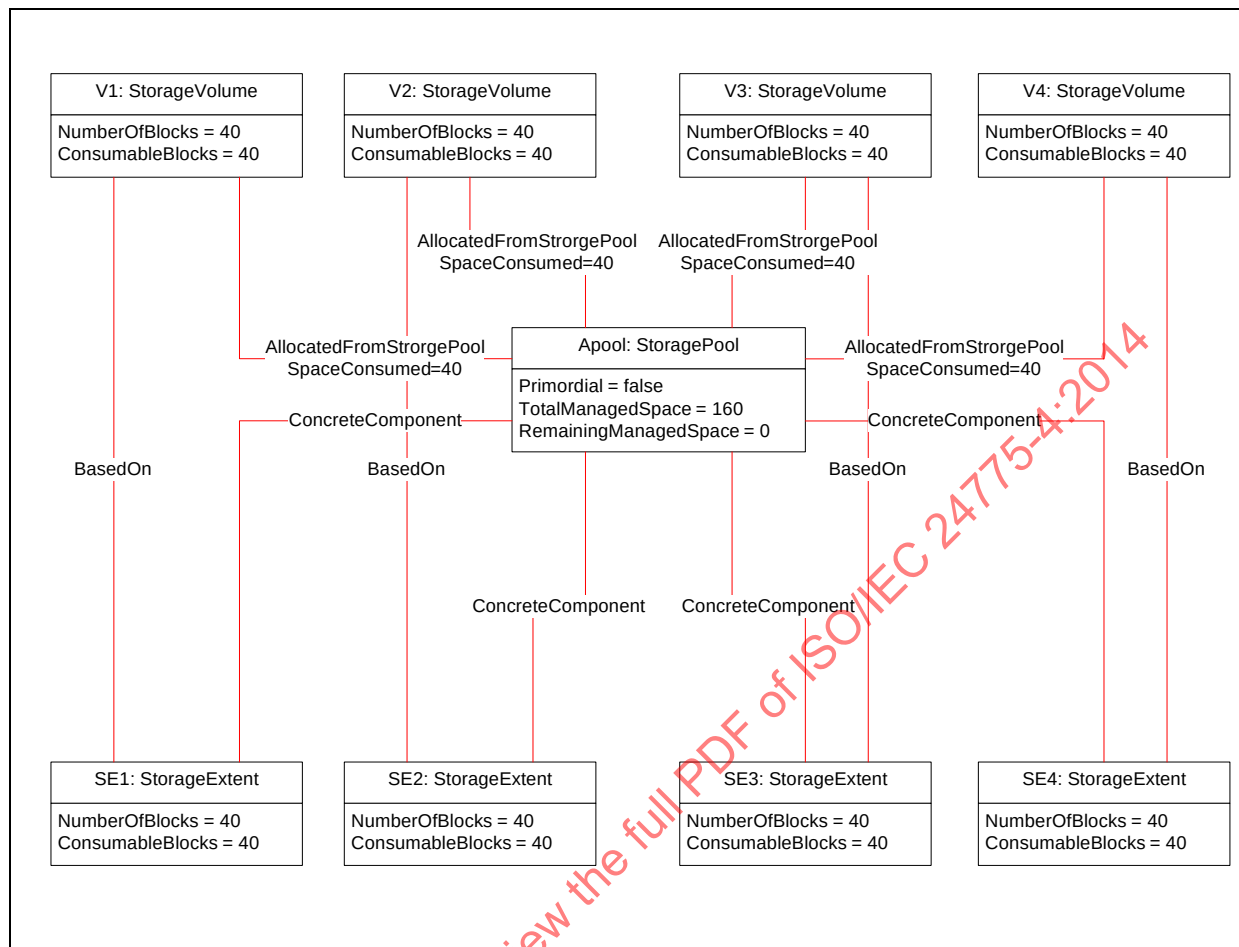


Figure 99 - Example 1 Step 1

Next, a composite volume is created by calling `CreateOrModifyCompositeElement` using three of the volumes (V1, V2, and V3). The result, shown in Figure 100, is the creation of a composite volume with the name V1 whose size is now 120 blocks and volumes V2 and V3 are now inaccessible. The volume V4 is unchanged. A `CompositeExtent` is added and is the Antecedent of a `BasedOn` association to the `StorageVolume`. In turn, the `BasedOn` associations that were going from volumes V1, V2, and V3 from extents SE1, SE2, and SE3 are now associated from the extents to the `CompositeExtent`.

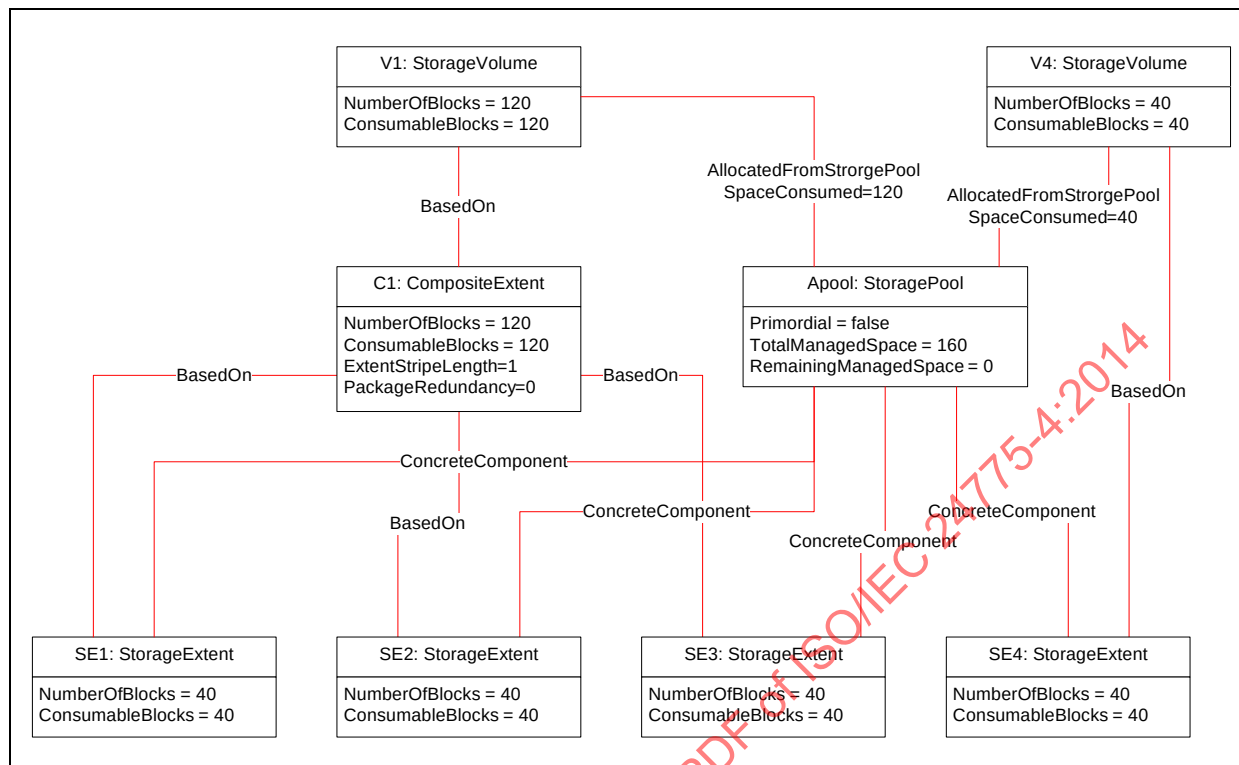


Figure 100 - Example 1 Step 2

24.1.7.2 First Alternative to Example 1

Figure 101 shows how the StorageSetting would be set when two volumes are turned into a composite. In this example, the volumes have a BasedOn relationship to a CompositeExtent. These volumes partially consume the underlying extent. Not shown in the diagram are the other StorageVolumes that consume the rest of the extent. In this example, the first volume, V1, has a DataRedundancy of 2 and a PackageRedundancy of 1. The second volume, V2, has a DataRedundancy of 1 and a PackageRedundancy of 0.

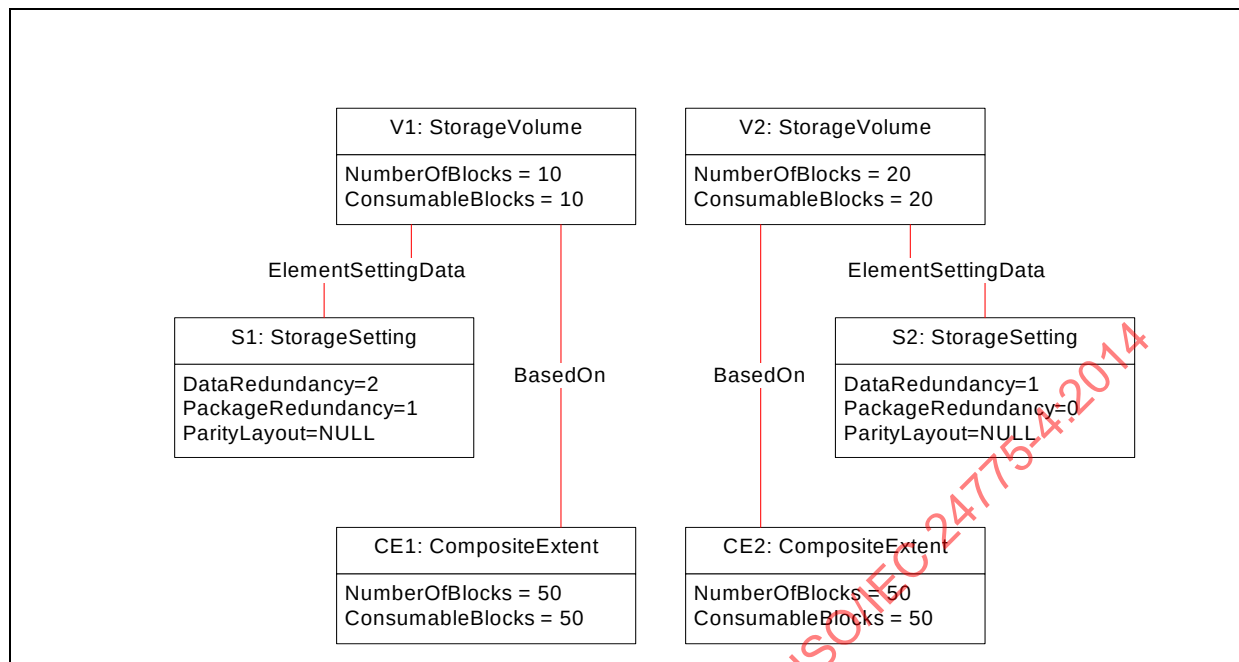


Figure 101 - First Alternative Example - Before Composition

As shown in Figure 102, after composition, the two volumes are combined into a single volume, V1, with a size equal to the sum of the prior two volumes. The StorageSetting of composite volume has been set to the lowest StorageSetting of the “before” volumes, which in this case is the StorageSetting from volume V2, for a DataRedundancy of 1 and a PackageRedundancy of 0. Also note that (partial) StorageExtents have been added between the CompositeExtent representing the composite volume (CE1-2) and the underlying CompositeExtents from before (CE1 and CE2). This is to preserve the consumption information of the original volumes.

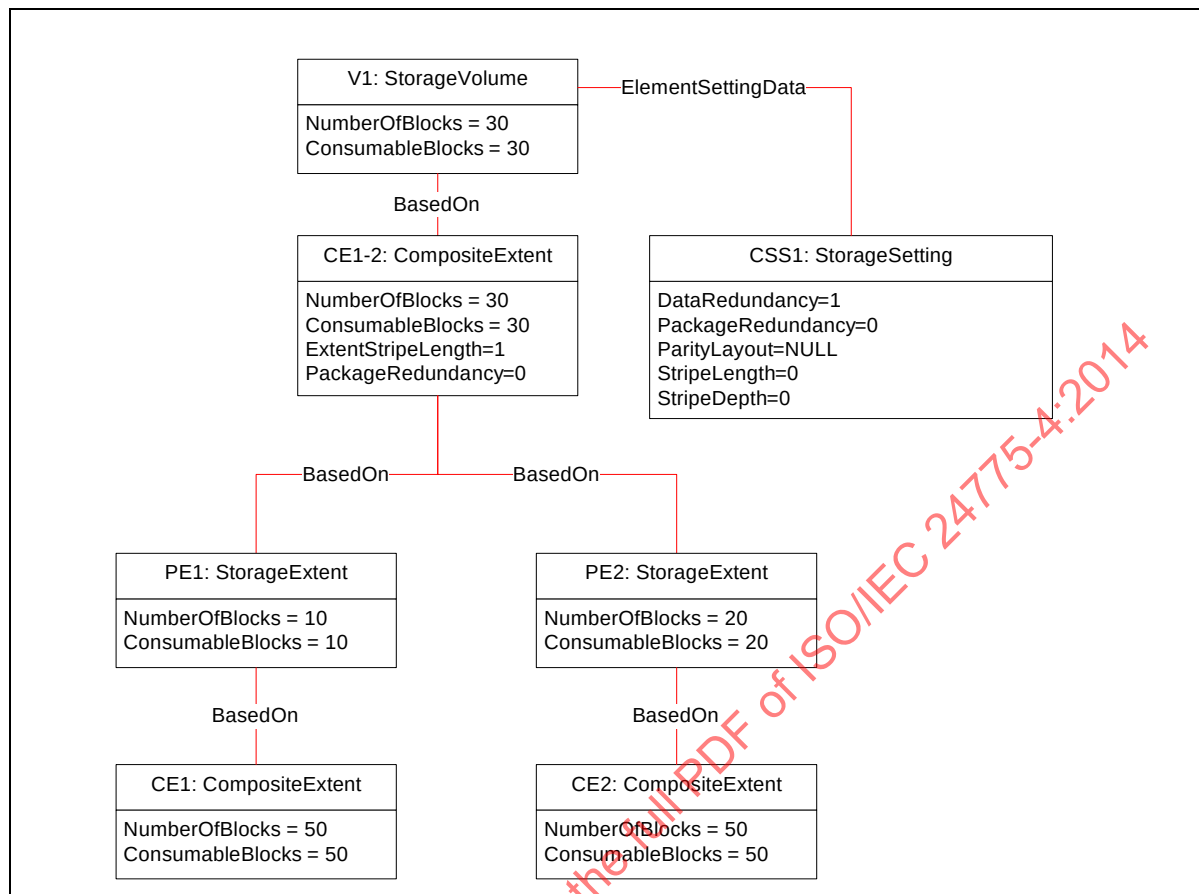


Figure 102 - First Alternative Example - After Composition

24.1.7.3 Second Alternative to Example 1

Figure 103 also shows an alternative extent model. In this example, the volumes have a BasedOn relationship to a CompositeExtent that in turn is based on an underlying StorageExtent (e.g. a ConcreteComponent of a concrete StoragePool). These volumes wholly consume the underlying extent. In this example, both volumes have a DataRedundancy of 2 and a PackageRedundancy of 1.

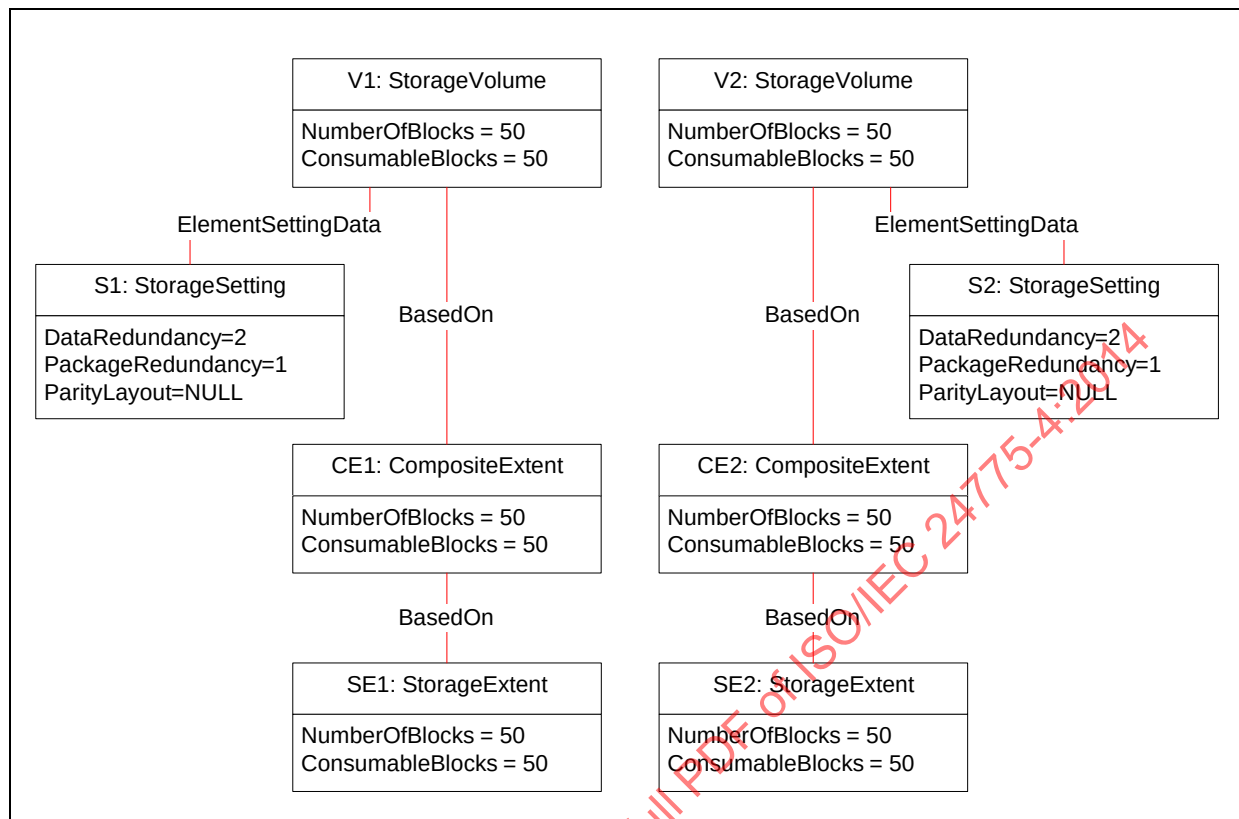


Figure 103 - Second Alternative Example - Before Composition

After composition, as shown in Figure 104, the two volumes are combined into a single volume, V1, with a size equal to the sum of the prior two volumes. The StorageSetting of the composite volume has been set to the StorageSetting of the “before” volumes, which in this case is a DataRedundancy of 2 and a PackageRedundancy of 1. Also note that the volume is now based on a single CompositeExtent (CE2 has been removed), which is now based on the previous two underlying StorageExtents.

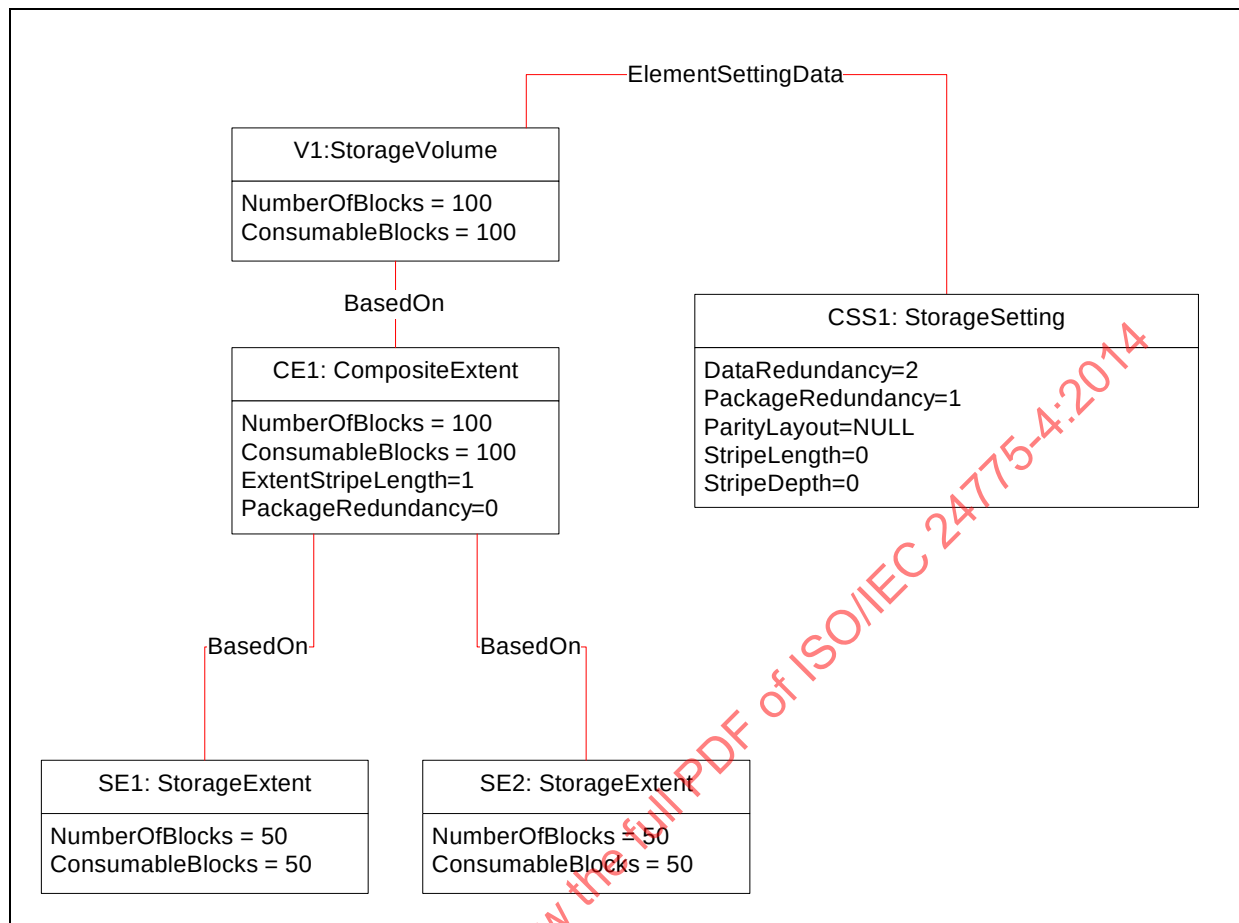


Figure 104 - Second Alternative Example - After Composition

24.1.7.4 Example 2

In this example, shown in Figure 105, a composite volume is built from volumes from two concrete storage pools. The configuration is the same as in the first example, except now there are two concrete StoragePools. Volumes V1 and V2 and extents SE1 and SE2 are associated to StoragePool A, and volumes V3 and V4 and extents SE3 and SE4 are associated to StoragePool B.

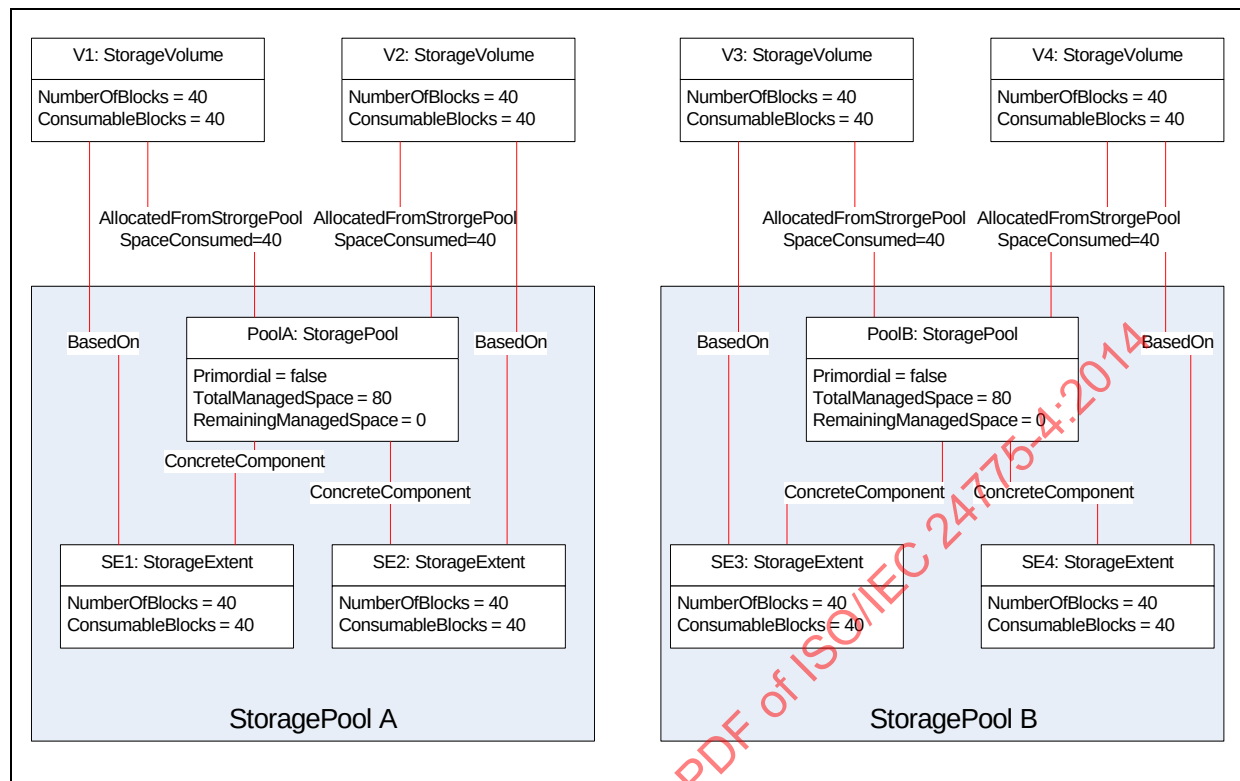


Figure 105 - Example 2 - Before Composition

Like the example shown in Figure 104, three volumes are combined into a composite volume, leaving one original volume. In this case, the composite volume has an `AllocatedFromStoragePool` association to each of the pools from which it was created. The `SpaceConsumed` property in the association is set to the space used from that particular pool. In this case, since two extents were consumed from StoragePool A and one from StoragePool B, the `AllocatedFromStoragePool.SpaceConsumed` for StoragePool A is 80 blocks and the `AllocatedFromStoragePool.SpaceConsumed` for StoragePool B is 40 blocks. The `CompositeExtent` has `BasedOn` associations to the underlying `StorageExtents` in each pool. Figure 106: "Example 2 - After Composition" shows the resulting model.

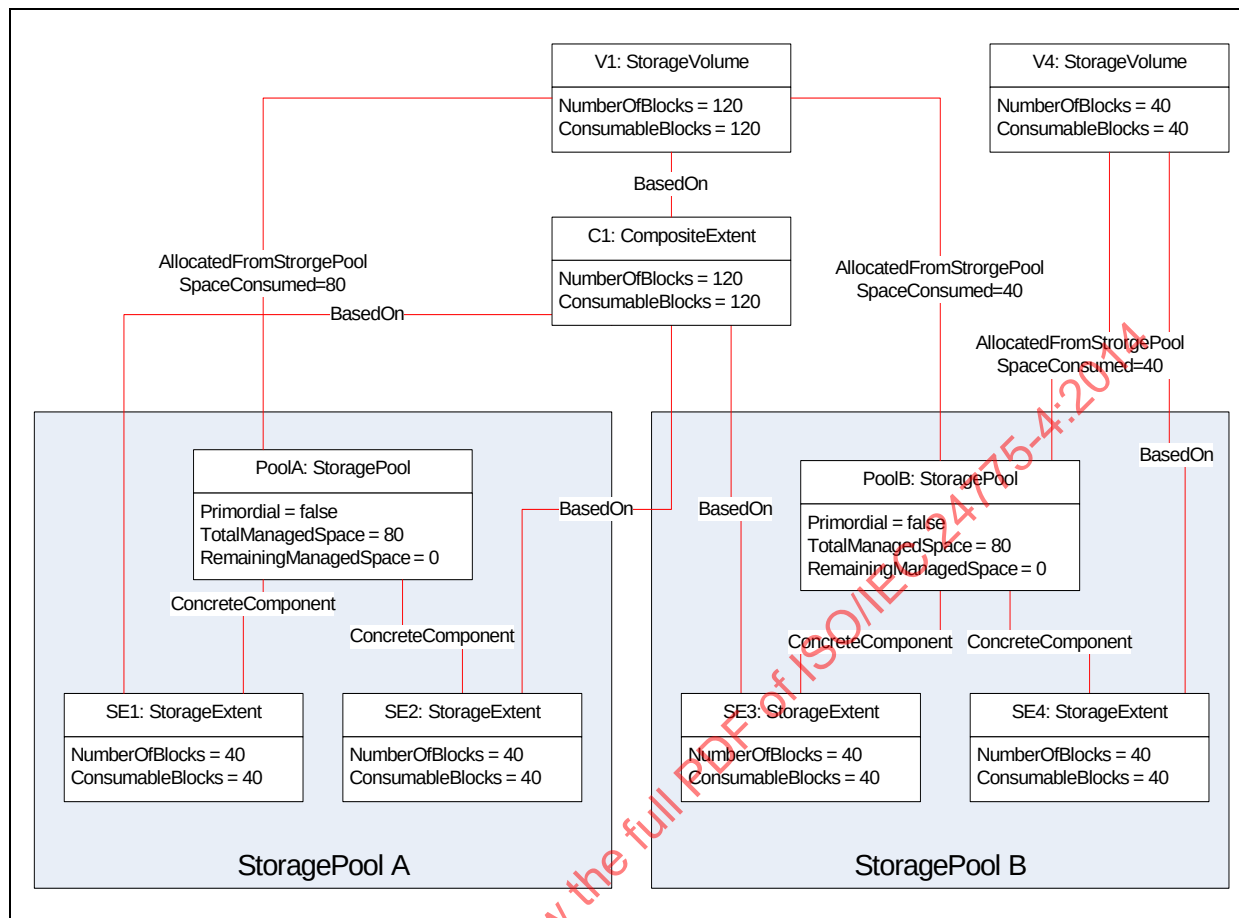


Figure 106 - Example 2 - After Composition

24.2 Striped and Concatenated Composite Volumes

The profile supports a composite volume that consists of striping across some constituent elements and concatenation among the remaining constituent elements, or vice versa. For example, Figure 107 shows the model for a composite volume that combines striping and concatenation. In this example, a composite volume consisting of "vol1" and "vol2" existed. Then, the composite volume was expanded using "vol3" and composite type of *Concatenate*. Therefore, the expanded composite volume now has a composition of "Concatenate+Stripe". It is also possible to start with a composite volume that has a composite type of *Concatenate* and expand it with two or more volumes that are Striped. In this case, the composition is still considered "Concatenate+Stripe".

Use the method 24.6.6 "GetCompositeElements" to determine which constituent elements are striped and which ones are concatenated.

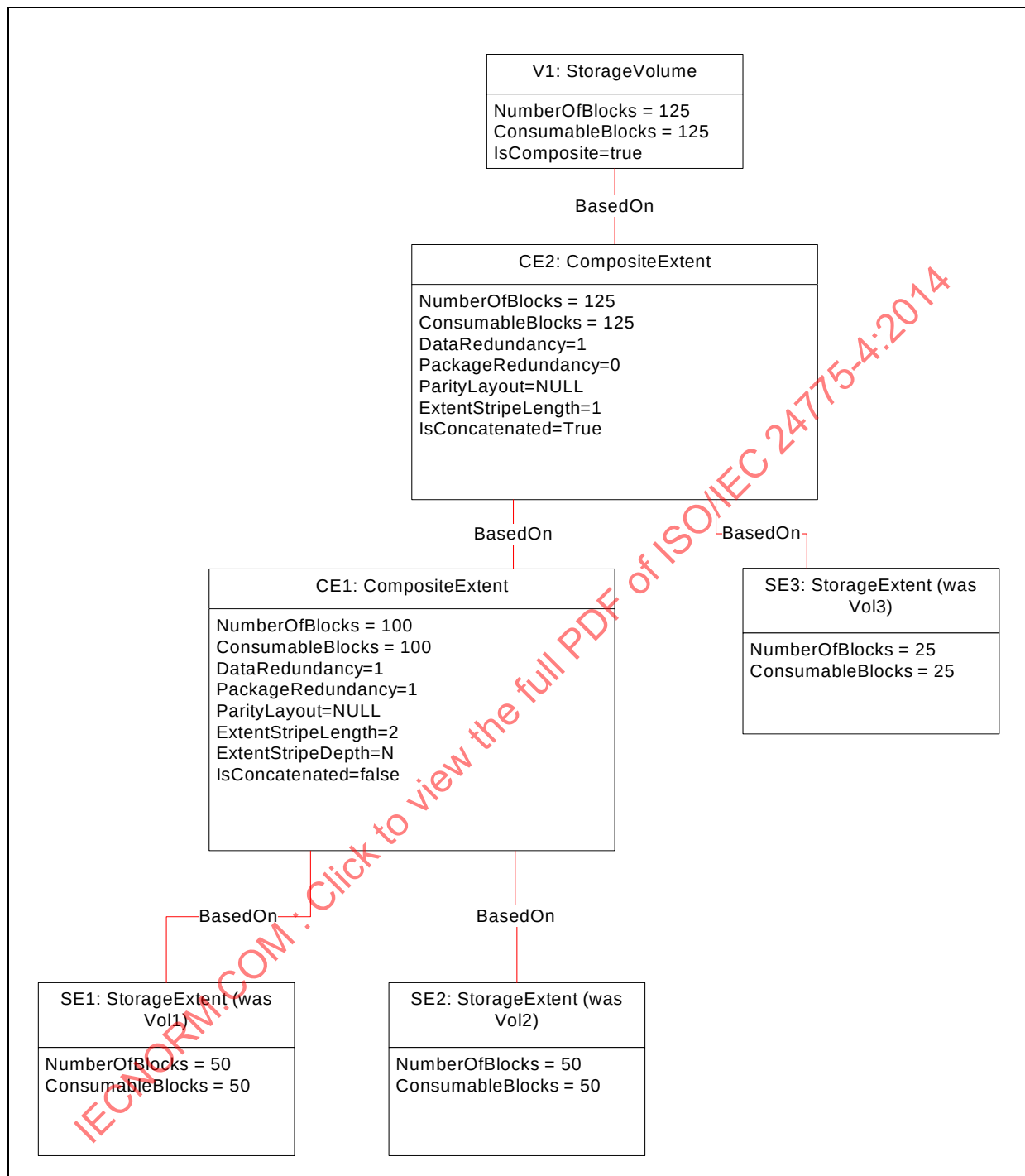


Figure 107 - Striping and Concatenation

24.3 Health and Fault Management Consideration

Not defined in this International Standard.

24.4 Cascading Considerations

None.

24.5 Supported Profiles, Subprofiles, and Packages

Table 434 describes the supported profiles for Volume Composition.

Table 434 - Supported Profiles for Volume Composition

Profile Name	Organization	Version	Requirement	Description
Extent Composition	SNIA	1.5.0	Mandatory	
Block Services	SNIA	1.5.0	Mandatory	

24.6 Methods of the Profile

24.6.1 General

Table 435 describes the methods of the profile.

Table 435 - Method Summary

Method	Created Instances	Modified Instances	Deleted Instances
CreateOrModifyCompositeElement	StorageVolume CompositeExtent	StorageVolume CompositeExtent	StorageVolume CompositeExtent
ReturnElementToElements	StorageVolume CompositeExtent	StorageVolume CompositeExtent	StorageVolume CompositeExtent
GetAvailableElements	N/A	N/A	N/A
GetCompositeElements	N/A	N/A	N/A
GetSupportedStripeLengths	N/A	N/A	N/A
GetSupportedStripeLengthRange	N/A	N/A	N/A
GetSupportedStripeDepths	N/A	N/A	N/A
GetSupportedStripeDepthRange	N/A	N/A	N/A
RemoveElementsFromElement	StorageVolume CompositeExtent	StorageVolume CompositeExtent	N/A

24.6.2 CreateOrModifyCompositeElement

24.6.2.1 General

This method is found in the StorageElementCompositionService. It creates or modifies a composite element. Only like elements (e.g., StorageVolumes) can be combined. In this version of the International Standard, only StorageVolumes may be used to create composite elements.

This method attempts to support vendors' sometimes complicated algorithms for creating and modifying composite storage elements, while simplifying it as much as possible. The key parameters are the Goal, RepresentativeElement, Size, InElements[], and TheElement. Setting one or more of these values will influence what the other values of these key parameters may be. These combinations will be described below. Of the other parameters, they are fairly self-explanatory and are described in Table 436. For this International Standard, ElementType shall only be "StorageVolume".

The Goal parameter specifies a set of generic QoS settings to use when creating the composite. The RepresentativeElement parameter is intended as a more detailed goal or QoS target for the composite. Because vendors have complex rules to create composites, it can be difficult to map those to the standard QoS settings that might be expressed in the usual setting

properties. By passing in a representative element, the client is indicating to the instrumentation that it should use additional vendor-specific information about that storage element when trying to create a composite. This allows for better interoperability because it hides those vendor rules, while still supporting vendor needs. If Goal or RepresentativeElement is non-null, then the other shall be null. InElements[] can also be used to deduce QoS setting to use in case neither Goal or RepresentativeElement is specified. In this case, the QoS for the composite element will be the lowest common denominator of the QoS values for the InElements array.

24.6.2.2 Creating a Composite

When creating a new composite storage element, there are two distinct modes of operation. Regardless of which mode is used, the following values shall apply:

The TheElement parameter shall be NULL. ElementName may be specified if the instrumentation supports naming of composite elements. CompositeType may be specified if the instrumentation supports the setting of this parameter. See the CompositeTypesSupported property in the StorageElementCompositionCapabilities class to determine if this can be set. Job will be non-NULL upon the method return if a Job was created.

The two creation use cases are the following:

- Pass in a non-empty list of extents (e.g., StorageVolumes) in InElements[] and a NULL Size parameter. The RepresentativeElement and Goal parameters may be NULL as the instrumentation will pick up the QoS goal from the InElements. If RepresentativeElement is not NULL, the instrumentation shall attempt to satisfy the QoS settings in the RepresentativeElement. It shall fail if it cannot create a composite that satisfies that QoS. If Goal is not NULL, the instrumentation shall attempt to satisfy the QoS settings in the Goal. It shall fail if it cannot create a composite that satisfies that Goal. The user may specify RepresentativeElement or Goal, but not both. The ElementSource parameter shall be NULL.
- Pass in a Size and a NULL InElements parameter. In this case, the instrumentation shall find the elements to use, based on the value of the ElementSource parameter, which may be NULL, indicating the instrumentation will determine the source of the elements. Goal or RepresentativeElement shall be specified. If RepresentativeElement is not NULL, the instrumentation shall attempt to satisfy the QoS settings in the RepresentativeElement. It shall fail if it cannot create a composite that satisfies that QoS. If Goal is not NULL, the instrumentation shall attempt to satisfy the QoS settings in the Goal. It shall fail if it cannot create a composite that satisfies that Goal. The user may specify RepresentativeElement or Goal, but not both. The size of the composite created shall be equal to or greater than the Size passed in.

24.6.2.3 Modifying a Composite

When modifying a composite, the client should examine the supported capabilities of the instrumentation before modifying a composite, as certain operations may result in data loss, depending upon the capabilities of the instrumentation.

Modifying a composite is similar to creation, with a few differences. The key difference is that TheElement shall be specified. ElementName may be specified if the instrumentation supports naming of composite elements. CompositeType may be specified if the instrumentation supports the setting of this parameter. See the CompositeTypesSupported property in the StorageElementCompositionCapabilities class to determine if this can be set. Job will be non-NULL upon the method return if a Job was created.

The two modification use cases are the following:

- Pass in a non-empty list of extents (e.g., StorageVolumes) in InElements[] and a NULL Size parameter. The RepresentativeElement and Goal parameters may be NULL as the instrumentation will pick up the QoS goal from the existing composite and the InElements. If RepresentativeElement is not NULL, the instrumentation shall attempt to satisfy the QoS settings in the RepresentativeElement. It shall fail if it cannot modify the composite to satisfy that QoS. If Goal is not NULL, the instrumentation shall attempt to satisfy the QoS settings in the Goal. It shall

fail if it cannot modify a composite to satisfy that Goal. The user may specify RepresentativeElement or Goal, but not both. If the Size parameter is NULL, the Instrumentation shall modify the composite size to be the current size plus the sum of the ConsumableBlocks times BlockSize of the InElements[] entries. The ElementSource parameter shall be NULL.

- Pass in a Size and a NULL InElements parameter. In this case, the instrumentation shall find the elements to use, based on the value of the ElementSource parameter, which may be NULL, indicating the instrumentation will determine the source of the elements. Goal or RepresentativeElement shall be specified. If RepresentativeElement is not NULL, the instrumentation shall attempt to satisfy the QoS settings in the RepresentativeElement. It shall fail if it cannot modify the composite to satisfy that QoS. If Goal is not NULL, the instrumentation shall attempt to satisfy the QoS settings in the Goal. It shall fail if it cannot modify a composite to satisfy that Goal. The user may specify RepresentativeElement or Goal, but not both. The size of the composite created shall be equal to or greater than the Size passed in. If Size is smaller than the current composite size, this may mean that volumes in the composite may remove from the composite.

Table 436 describes the return values for the CreateOrModifyCompositeElement method.

Table 436 - CreateOrModifyCompositeElement

Method: CreateOrModifyCompositeElement			
Return Values:			
Value		Description	
0: Success		Job completed with no error.	
1: Not Supported		Not supported.	
2: Unknown		Unknown error occurred.	
3: Timeout		Timeout.	
4: Failed		Method failed.	
5: Invalid Parameter			
6: In Use		Element is in use and cannot be modified.	
4096: Method Parameters Checked - Job started		Job was started.	
4097: Size Not supported			
Parameters:			
Qualifiers	Name	Type	Description/Values.
IN	ElementName	string	End-user relevant name for the element created.
IN	ElementType	uint16	Type of element being created.
OUT	Job	REF ConcreteJob	Reference to the job created.
IN	Goal	REF ManagedElement	The QoS requirements for the composite element to maintain. This parameter may be null. If both Goal and RepresentativeElement are null, the implementation selects an appropriate Goal from the InElements. When a StorageSetting is used, this will include the stripe length and depth.

Table 436 - CreateOrModifyCompositeElement (Continued)

Method: CreateOrModifyCompositeElement			
IN	RepresentativeElement	REF StorageExtent	The instrumentation will use this parameter + Size or InElements to determine the elements used to construct the composite. This parameter may be NULL. If both Goal and RepresentativeElement are null, the implementation selects an appropriate Goal from the InElements.
IN/OUT	Size	uint64	Unit: bytes As an input parameter Size specifies the desired size. If NULL, then InElements shall be supplied. If not NULL, this parameter will supply a new size when creating or modifying an existing element. As an output parameter Size specifies the size achieved.
IN	InElements[]	REF StorageExtent	The elements from which to create the composite element. If this parameter is NULL then Size shall be non-NULL. Once the elements are combined, they will be removed from the model and replaced with a single element. For some instrumentation, this may be one of the InElements, so in effect, all but one are removed.
IN/OUT	TheElement	REF LogicalElement	When used to create a composite, this shall be NULL Upon modification, this shall specify an existing composite element. The method will then modify the specified element. Upon completion (unless a Job is started), a reference to the resulting element shall be returned.
IN	CompositeType	uint16	Type of composite element to create. Possible values are Concatenate, Stripe, Concatenate+Stripe, Vendor specific. If NULL, the instrumentation will decide.
IN	ElementSource	uint16	Tell the instrumentation where to get the elements. Only applies when Size is specified and not InElements. Otherwise it shall be NULL. Possible values are: 1. Use existing elements only 2. Create new elements only 3. Can use existing or create new or both 4. Instrumentation decides If NULL, the instrumentation will decide.

24.6.3 RemoveElementsFromElement

This method is found in the StorageElementCompositionService. It removes selected elements from a composite volume. Note that the elements returned may not match the elements that went into the composite (e.g., VPD page 83h information may not be the same). Also, removing a member element from a composite element may impact the data stored on the remaining

members (see Table 433, "CompositionCharacteristics Property"). Removing all members is the same as calling ReturnElementToElements.

Table 437 describes the return values for the RemoveElementsFromElement method.

Table 437 - RemoveElementsFromElement

Method: RemoveElementsFromElement			
Return Values:			
Value		Description	
0: Success		Job completed with no error.	
1: Not Supported		Not supported.	
2: Unknown		Unknown error occurred.	
3: Timeout		Timeout.	
4: Failed		Method failed.	
5: Invalid Parameter			
6: In Use		Element is in use and cannot be modified.	
4096: Method Parameters Checked - Job started		Job was started.	
Parameters:			
Qualifiers	Name	Type	Description/Values
OUT	Job	REF ConcreteJob	Reference to the job created.
INOUT	TheElement	REF StorageVolume	Composite element to modify. Returns element in case object path changes as a result of removal.
IN	InElements[]	REF StorageExtent	The elements to remove from the composite element. These may be found by calling GetCompositeElements or keeping track of the elements that went into the composite.

24.6.4 ReturnElementToElements

This method is found in the StorageElementCompositionService. It dissolves a composite into its constituent elements. Note that the elements returned may not match the elements that went into the composite (e.g., VPD page 83h information may not be the same).

Table 438 describes the return values for the ReturnElementToElements method.

Table 438 - ReturnElementToElements

Method: ReturnElementToElements	
Return Values:	
Value	Description
0: Success	Job completed with no error.
1: Not Supported	Method not supported.
2: Unknown	Unknown error occurred.
3: Timeout	Operation timed out.
4: Failed	Operation failed.
5: Invalid Parameter	Invalid parameter.

Table 438 - ReturnElementToElements (Continued)

Method: ReturnElementToElements			
6: In use		Element is in use and cannot be dissolved.	
4096: Method Parameters Checked - Job started		Job was started.	
Parameters:			
Qualifiers	Name	Type	Description/Values
OUT	Job	REF ConcreteJob	Reference to the job created.
IN	TheElement	REF LogicalElement	The composite element to dissolve.
OUT	OutElements[]	REF StorageExtent	Elements the composite was dissolved into.

24.6.5 GetAvailableElements

This method, found in the StorageElementCompositionService, queries the set of pools passed in and returns a set of elements (volumes or logical disks) that can be composed together based on the specified goal and element passed in. Since there are usually complicated vendor-specific rules for creating these composite volumes, using the representative element can supply more vendor-specific information than there would be in a interoperable setting. The client can then use some or all of this list in a call to CreateOrModifyCompositeElement().

In this International Standard, only StorageVolumes shall be supported as the ElementType.

Table 439 describes the return values for the GetAvailableElements method.

Table 439 - GetAvailableElements

Method: GetAvailableElements			
Return Values:			
Value		Description	
0: Success		Job completed with no error.	
1: Not Supported		Method not supported.	
2: Unknown		Unknown error occurred.	
3: Timeout		Operation timed out.	
4: Failed		Operation failed.	
5: Invalid Parameter		Invalid parameter.	
6: In use		Element is in use and cannot be dissolved.	
4096: Method Parameters Checked - Job started		Job was started.	
Parameters:			
Qualifiers	Name	Type	Description/Values.
OUT	Job	REF ConcreteJob	Reference to the job created.
IN	InPools[]	REF StoragePool	List of pools to look in.
IN	Goal	REF StorageSetting	The QoS goal requirements for the composite element. Can be NULL. If it is NULL, then RepresentativeElement shall be non-NULL.

Table 439 - GetAvailableElements (Continued)

Method: GetAvailableElements			
IN	ElementType	uint16	Enumeration indicating the type of element being created or modified Values: 2: StorageVolume 3: LogicalDisk
IN	RepresentativeElement	REF StorageExtent	Serves as a guide to help the instrumentation determine which elements to return. It shall be a member of one of the pools passed in. This may be NULL, only if Goal is non-NULL.
OUT	Candidates[]	REF StorageExtent	The elements that can be used to create the composite element. These will be an array of references to StorageVolumes or LogicalDisks.

24.6.6 GetCompositeElements

This method is found in the StorageElementCompositionService. It is used to query an existing composite element to determine the component elements that make up that composite element (i.e., the “parents” of a composite element). If the method is executed under control of a job, examine the AffectedJobElement associations for the list of the constituent elements after the job completes.

Table 440 describes the return values for the GetCompositeElements method.

Table 440 - GetCompositeElements

Method: GetCompositeElements			
Return Values:			
Value	Description		
0: Success	Method completed with no error.		
1: Not Supported	Method not supported.		
2: Unknown	Unknown error occurred.		
3: Timeout	Operation timed out.		
4: Failed	Operation failed.		
5: Invalid Parameter	Invalid parameter.		
6: In use	Element is in use and cannot be accessed.		
4096: Method Parameters Checked - Job started	Job was started.		
Parameters:			
Qualifiers	Name	Type	Description/Values
OUT	Job	REF ConcreteJob	Reference to the job created.
IN	TheElement	REF StorageExtent	The element to query.
IN	RequestType	uint16	Possible values are: Immediate – return the immediate “parent” of TheElement. Primordial – return dependent storage extents of TheElement at the lowest extent hierarchy.
OUT	OutElements[]	REF StorageExtent	The elements that comprise the composite.

Table 440 - GetCompositeElements (Continued)

Method: GetCompositeElements			
OUT	OutElementTypes[]	uint16	A parallel array to OutElements array. Possible values: Member of Stripe Set, and Member of Concatenation.

24.6.7 GetSupportedStripeLengths

This method is found in the StorageElementCompositionService. This method returns the list of possible stripe lengths which can be used in the property StorageSetting.ExtentStripeLength supplied, as the Goal, to the CreateOrModifyCompositeElement method. Note that different implementations may support either the GetSupportedStripeLengths or the GetSupportedStripeLengthRange method. If the system only supports a range of lengths, then the return value will be set to 3.

Table 441 describes the return values for the GetSupportedStripeLengths method.

Table 441 - GetSupportedStripeLengths

Method: GetSupportedStripeLengths			
Return Values			
Value	Description		
0	Method completed with no error.		
1	Method not supported.		
2	ElementType not supported.		
3	Use <code>GetSupportedStripeLengthRange</code> instead.		
Parameters			
Qualifiers	Name	Type	Description/Values
IN	ElementType	uint16	Type of element.
OUT	StripeLengths[]	uint64	List of supported stripe Lengths.

24.6.8 GetSupportedStripeLengthRange

This method is found in the StorageElementCompositionService. For systems that support a range of stripe lengths for composite volumes, this method can be used to retrieve the range of possible stripe lengths which can be used in the property StorageSetting.ExtentStripeLength supplied, as the Goal, to the CreateOrModifyCompositeElement method. Note that different implementations may support either the GetSupportedStripeLengths or the GetSupportedStripeLengthRange method. If the system only supports discrete values, then the return value will be set to 3.

Table 442 describes the return values for the GetSupportedStripeLengthRange method.

Table 442 - GetSupportedStripeLengthRange

Method: GetSupportedStripeLengthRange	
Return Values	
Value	Description
0	Method completed with no error.
1	Method not supported.

Table 442 - GetSupportedStripeLengthRange (Continued)

Method: GetSupportedStripeLengthRange			
2	ElementType not supported.		
3	Use GetSupportedStripeLengths instead.		
Parameters			
Qualifiers	Name	Type	Description/Values
IN	ElementType	uint16	Type of element.
OUT	MinimumStripeLength	uint64	Minimum ExtentStripeLength for a composite element.
OUT	MaximumStripeLength	uint64	Maximum ExtentStripeLength for a composite element.
OUT	StripeLengthDivisor	uint64	Composite element's stripe length shall be a multiple of this value.

24.6.9 GetSupportedStripeDepths

This method is found in the StorageElementCompositionService. This method returns the list of possible stripe depths which can be used in the property StorageSetting.UserDataStripeDepth supplied, as the Goal, to the CreateOrModifyCompositeElement method for systems that support discrete stripe depths. For systems that require the stripe depth to be on a given boundary, such as 512, the stripe length will be rounded up to the next higher value that is a multiple of the required boundary. Note that different implementations may support either the GetSupportedStripeDepths or the GetSupportedStripeDepthRange method. If the system only supports a range of stripe depths, then the return value will be set to 3.

Table 443 describes the return values for the GetSupportedStripeDepths method.

Table 443 - GetSupportedStripeDepths

Method: GetSupportedStripeDepths			
Return Values			
Value	Description		
0	Method completed with no error.		
1	Method not supported.		
2	ElementType not supported.		
3	Use GetSupportedStripeDepthRange instead.		
Parameters			
Qualifiers	Name	Type	Description/Values
IN	ElementType	uint16	Type of element.
OUT	StripeDepths[]	uint64	List of supported stripe depths.

24.6.10 GetSupportedStripeDepthRange

This method is found in the StorageElementCompositionService. For systems that support a range of stripe depths for composite volumes, this method can be used to retrieve the range of possible stripe depths which can be used in the property StorageSetting.UserDataStripeDepth supplied, as the Goal, to the CreateOrModifyCompositeElement method. Note that different implementations may support either the GetSupportedStripeDepths or the

GetSupportedStripeDepthRange method. If the system only supports discrete values, then the return value will be set to 3.

Table 444 describes the return values for the GetSupportedStripeDepthRange method.

Table 444 - GetSupportedStripeDepthRange

Method: GetSupportedStripeDepthRange			
Return Values			
Value	Description		
0	Method completed with no error.		
1	Method not supported.		
2	ElementType not supported.		
3	Use GetSupportedStripeDepths instead.		
Parameters			
Qualifiers	Name	Type	Description/Values
IN	ElementType	uint16	Type of element.
OUT	MinimumStripeDepth	uint64	Minimum User-DataStripeDepth for a composite element.
OUT	MaximumStripeDepth	uint64	Maximum User-DataStripeDepth for a composite element.
OUT	StripeDepthDivisor	uint64	Composite element's stripe depth shall be a multiple of this value.

24.7 Client Considerations and Recipes

24.7.1 Indications

When storage elements are combined into a composite or a composite is dissolved, indications shall be sent. When a composite is created, the instrumentation shall send an InstDelete indication for all volumes that no longer exist as StorageVolumes. The AllocatedFromStoragePool association shall be deleted, as well as the ElementSettingData association and its associated StorageSetting. Indications shall not be required to be sent for those deletions. If the storage element still exists but is no longer accessible, the provider may send an InstModification indication for the StorageVolume depending upon whether or not there are any changes to the storage element itself. If the instrumentation creates a new storage element, then it shall send an InstCreation indication for the new element. If the instrumentation modifies an existing element and it becomes the element to represent a composite, an InstModification indication shall be sent. InstModification indications for the AllocatedFromStoragePool association, ElementSettingData association, and associated StorageSetting shall not be required.

When a composite is dissolved, the instrumentation shall send an InstCreation indication for each storage element created. It shall send an InstDeletion indication if the composite element is deleted and an InstModification indication if the composite element is merely modified. Indications for the AllocatedFromStoragePool associations, ElementSettingData associations, and associated StorageSettings that are created, deleted, or modified as a result of the dissolution of the composite shall not be required.

The user is advised to check the StorageSetting for the storage elements they are interested in after composite creation or deletion as those settings may have changed from what they were before.

24.7.2 Recipe 1: Create Composite Volume

In this use case, all available storage is consumed in StorageVolumes. The client wishes to create a larger volume from volumes that are not being used. The client will call CreateOrModifyElementFromElements(), passing in a set storage volumes. The provider will then create the concatenated volume.

```
// DESCRIPTION:
//
// Create a composite volume
//
// PRE-EXISTING CONDITIONS AND ASSUMPTIONS
//
// 1. The StorageElementCompositionService has been found and the object
//    path
//    value is stored in $CompositionService->
// 2. The list of elements (volumes) to use to create the composite has
//    been
//    identified and the object path values are stored in $volumes->[]
// 3. The StorageSetting to use has been identified and the object path
//    values are stored in $Goal->
// 4. The type of element to create has been selected (LogicalDisk or
//    StorageVolume and it's value stored in #ElementType
// 5. A representative element (LogicalDisk or StorageVolume instance)
//    has been identified and it's object path stored in $SampleElement->
//    Note: this is allowed to be null
// 6. The StorageElementCompositionCapabilities associated to the
//    StorageElementCompositionService has been found and the instance
//    stored in $CompositionCapabilities
// 7. The type of composite to create has been identified and the value
//    stored in #CompositeType

// Determine if there is a job created by method
// and wait for the job to complete
// Input:
// #ReturnCode : The return code of the method
// $ConcreteJob-> :The output parameter that may have a ConcreteJob
//                 REF.
// This method will return control if the recipe was not exited because
// of error
sub void WaitForJob(#ReturnCode, $ConcreteJob->) {
  if (4096 == #ReturnCode) {
    if ($ConcreteJob-> != null) {
      /*Wait until the completion of the job using $ConcreteJob-> as
      a filter Verify that the OperationalStatus contains 2 ("OK"),
      or 17 ("Completed") */
      $JobInstance = GetInstance($ConcreteJob->,
        false, false, false, null)
      if ($JobInstance.JobState != 7) { // 7 - Completed
        <ERROR! Job failed! >
      }
    }
  } else {
    <ERROR! Missing Job reference>
  }
}
```

```

    }
  }
}

// Step 1. See if creation is supported

if ($CompositionCapabilities.SupportsComposites == false) {
  <ERROR! Volume composition not supported>
}
if ((contains("CreateOrModifyCompositeElement",
  $CompositionCapabilities.SupportedAsynchronousActions[]) ==
  FALSE)
  && (contains("CreateOrModifyCompositeElement",
  $CompositionCapabilities.SupportedSynchronousActions[]) ==
  FALSE)) {
  <ERROR! Volume composition creation not supported>
}

// Step 2. Subscribe to indications
#Filter1 = "SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA
          CIM_StorageVolume";
#Filter2 = "SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA
          CIM_StorageVolume";
// Determine if the Indication filters already exist
// If they don't, create them

// Step 3. Create the composite
%InputArguments["ElementName"] = {"Test"}
%InputArguments["ElementType"] = #ElementType
%InputArguments["Goal"] = $Goal->
%InputArguments["InElements"] = {$volumes->}
%InputArguments["ElementType"] = #CompositeType

#ReturnCode = InvokeMethod($CompositionService->,
  "CreateOrModifyCompositeElement",
  %InputArguments,
  %OutputArguments)
// 0 is "Success" and 4096 is "Method Parameters Checked - Job Started"
if (#ReturnCode != 0 || #ReturnCode != 4096) {
  <ERROR! Method failure>
}

$Job-> = %OutputArguments["Job"]
if ($Job-> == null) {
  $CompositeCreated-> = %OutputArguments["TheElement"]
}
else {
  // Wait until job is finished
  &WaitForJob(#ReturnCode, $Job->)

  // Now get the composite just created

```

```
$CompositeCreated-> = Associators($Job->,
    "CIM_AffectedJobElement",
    "CIM_StorageExtent",
    "AffectingElement",
    "AffectedElement",
    false, false, null)
}

// Step 4: Verify the volumes that went into the composite are no longer
// accessible
for #i in $Candidates->[] {
    if ($Candidates->[#i] == $CompositeCreated->) {
        // It's allowed for the created composite to take over the
        // identity of a volume that went into creating it
    }
    else {
        $Refs->[] = AssociatorNames(
            $Candidates->[#i],
            "CIM_SystemDevice", // AssocClass
            "CIM_ComputerSystem", // ResultClass
            "PartComponent", // Role
            null )
        if( $Refs->[].length != 0 )
        {
            <"ERROR! Composite volume component still exists">
        }
    }
}
}
```

24.7.3 Recipe 2: Delete Composite Volume

In this use case, the client wishes to return a concatenated volume to its individual component storage volumes. The client calls `ReturnElementToElements()` to dissolve the concatenated volume.

```
// DESCRIPTION:
//
// Delete a composite volume
//
// PRE-EXISTING CONDITIONS AND ASSUMPTIONS
//
// 1. The StorageElementCompositionService has been found and the object
//    path
//    value is stored in $CompositionService->
// 2. A composite volume has been created and the object path stored in
//    $compositeVolume->
// 3. The StorageElementCompositionCapabilities associated to the
//    StorageElementCompositionService has been found and the instance
//    stored in $CompositionCapabilities

// Determine if there is a job created by method
```

```

// and wait for the job to complete
// Input:
//   #ReturnCode : The return code of the method
//   $ConcreteJob-> :The output parameter that may have a ConcreteJob
//                   REF.
// This method will return control if the recipe was not exited because
// of error
sub void WaitForJob(#ReturnCode, $ConcreteJob->) {
  if (4096 == #ReturnCode) {
    if ($ConcreteJob-> != null) {
      /*Wait until the completion of the job using $ConcreteJob-> as
      a filter Verify that the OperationalStatus contains 2 ("OK"),
      or 17 ("Completed") */
      $JobInstance = GetInstance($ConcreteJob->,
        false, false, false, null)
      if ($JobInstance.JobState != 7) { // 7 - Completed
        <ERROR! Job failed! >
      }
    } else {
      <ERROR! Missing Job reference>
    }
  }
}

// Step 1. Subscribe to indications
#Filter1 = "SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA
          CIM_StorageVolume";
#Filter2 = "SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA
          CIM_StorageVolume";
// Determine if the Indication filters already exist
// If they don't, create them

// Step 2. Dissolve the composite volume

if ($CompositionCapabilities.SupportsComposites == false) {
  <ERROR! Volume composition not supported>
}

%InputArguments["TheElement"]          = $CompositeVolume->

#ReturnCode = InvokeMethod($CompositionService->,
  "ReturnElementToElement",
  %InputArguments,
  %OutputArguments)
// 0 is "Success" and 4096 is "Method Parameters Checked - Job Started"
if (#ReturnCode != 0 || #ReturnCode != 4096) {
  <ERROR! Method failure>
}

$Job-> = %OutputArguments["Job"]
if ($Job-> == null) {
  $Volumes->[] = %OutputArguments["OutElements"]
}

```

```

}
else {
    // Wait until job is finished
    &WaitForJob(#ReturnCode, $Job->)

    // Now get the SPCs
    $Volumes->[] = Associators(
        $Job->,
        "CIM_AffectedJobElement",
        "CIM_StorageExtent",
        "AffectingElement",
        "AffectedElement",
        false, false, null)
}

// Step 2. Show the elements from the former composite
for #v in $Volumes->[] {
    $AnInstance = GetInstance($Volumes->[#i],
                                false, false, false, null)

    // Display instance
}

```

24.8 Registered Name and Version

Volume Composition version 1.5.0 (Component Profile).

24.9 CIM Elements

24.9.1 General

Table 445 describes the CIM elements for Volume Composition.

Table 445 - CIM Elements for Volume Composition

Element Name	Requirement	Description
24.9.2 CIM_CompositeExtent	Mandatory	
24.9.3 CIM_CompositeExtentBasedOn (Volume Composition)	Mandatory	
24.9.4 CIM_ElementCapabilities	Mandatory	
24.9.5 CIM_ElementSettingData	Mandatory	
24.9.6 CIM_HostedService (Associates ComputerSystem and the ElementCompositionService)	Mandatory	
24.9.7 CIM_StorageElementCompositionCapabilities	Mandatory	
24.9.8 CIM_StorageElementCompositionService	Mandatory	

Table 445 - CIM Elements for Volume Composition (Continued)

Element Name	Requirement	Description
24.9.9 CIM_StorageSetting	Mandatory	
24.9.10 CIM_StorageVolume	Conditional	Conditional requirement: Storage Volumes used as storage elements.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_StorageVolume	Conditional	Conditional requirement: Storage Volumes used as storage elements. Modification of a StorageVolume upon creation or deletion of a composite.

24.9.2 CIM_CompositeExtent

Created By: Extrinsic

Modified By: Extrinsic

Deleted By: Extrinsic

Requirement: Mandatory.

Table 446 describes class CIM_CompositeExtent.

Table 446 - SMI Referenced Properties/Methods for CIM_CompositeExtent

Properties	Flags	Requirement	Description and Notes
IsConcatenated		Mandatory	Indicates data is concatenated across extents in the group.
BlockSize		Mandatory	Size in bytes of the blocks which form this StorageExtent.
NumberOfBlocks		Mandatory	
ConsumableBlocks		Mandatory	The maximum number of blocks, of size BlockSize, which are available for consumption.
SystemCreationClassName		Mandatory	The scoping System CreationClassName.
SystemName		Mandatory	The scoping System Name.
CreationClassName		Mandatory	The name of the concrete subclass.
Name		Mandatory	Unique identifier for the Service.

24.9.3 CIM_CompositeExtentBasedOn (Volume Composition)

Created By: Extrinsic: CreateOrModifyCompositeElement, ReturnElementToElements

Modified By: External

Deleted By: Extrinsic: CreateOrModifyCompositeElement, ReturnElementToElements

Requirement: Mandatory.

Table 447 describes class CIM_CompositeExtentBasedOn (Volume Composition).

Table 447 - SMI Referenced Properties/Methods for CIM_CompositeExtentBasedOn (Volume Composition)

Properties	Flags	Requirement	Description and Notes
Dependent		Mandatory	
Antecedent		Mandatory	

24.9.4 CIM_ElementCapabilities

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 448 describes class CIM_ElementCapabilities.

Table 448 - SMI Referenced Properties/Methods for CIM_ElementCapabilities

Properties	Flags	Requirement	Description and Notes
ManagedElement		Mandatory	
Capabilities		Mandatory	

24.9.5 CIM_ElementSettingData

Created By: Extrinsic: CreateOrModifyCompositeElement, ReturnElementToElements

Modified By: Static

Deleted By: Extrinsic: CreateOrModifyCompositeElement, ReturnElementToElements

Requirement: Mandatory

Table 449 describes class CIM_ElementSettingData.

Table 449 - SMI Referenced Properties/Methods for CIM_ElementSettingData

Properties	Flags	Requirement	Description and Notes
ManagedElement		Mandatory	StorageVolume or LogicalDisk.
SettingData		Mandatory	The composite setting data object associated with the composite element.

24.9.6 CIM_HostedService (Associates ComputerSystem and the ElementComposition-Service)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 450 describes class CIM_HostedService (Associates ComputerSystem and the ElementCompositionService).

Table 450 - SMI Referenced Properties/Methods for CIM_HostedService (Associates ComputerSystem and the ElementCompositionService)

Properties	Flags	Requirement	Description and Notes
Antecedent		Mandatory	
Dependent		Mandatory	

24.9.7 CIM_StorageElementCompositionCapabilities

Created By: Static

Requirement: Mandatory.

Table 451 describes class CIM_StorageElementCompositionCapabilities.

Table 451 - SMI Referenced Properties/Methods for CIM_StorageElementCompositionCapabilities

Properties	Flags	Requirement	Description and Notes
ElementName		Mandatory	User friendly name for this instance of Capabilities.
InstanceID		Mandatory	Unique identifier for the instance.
SupportsComposites		Mandatory	Indicates if instrumentation supports composite elements.
MaxCompositeSize		Mandatory	Indicates the largest composite element that can be created in bytes.
MaxCompositeElements		Mandatory	Indicates the most elements that can be combined into a composite element.
CompositionCharacteristics		Mandatory	Composition characteristics supported by this system.
SupportedAsynchronousActions		Mandatory	Indicates which methods are executed asynchronously.
SupportedSynchronousActions		Mandatory	Indicates which methods are executed synchronously.
SupportedStorageElements		Mandatory	Managed element types that can be composited. Currently only StorageVolume.
CompositionMethodsSupported		Mandatory	Composition methods supported.
CompositeSourcesSupported		Mandatory	Composition sources supported.
SupportsCompositeNaming		Mandatory	Can the user name the composite.
SupportsRepresentativeElement		Mandatory	Can the user specify the RepresentativeElement in CreateOrModifyComposite and GetAvailableElements.

24.9.8 CIM_StorageElementCompositionService

Created By: Static

Requirement: Mandatory.

Table 452 describes class CIM_StorageElementCompositionService.

Table 452 - SMI Referenced Properties/Methods for CIM_StorageElementCompositionService

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	The scoping System CreationClassName.
SystemName		Mandatory	The scoping System Name.
CreationClassName		Mandatory	The name of the concrete subclass.
Name		Mandatory	Unique identifier for the Service.
CreateOrModifyCompositeElement()		Mandatory	This method creates or modifies a composite element. Only like elements (e.g. StorageVolumes) can be combined.
ReturnElementToElements()		Mandatory	Dissolve the composite. All elements in the composite are restored.
RemoveElementsFromElement()		Optional	Removes one or more constituent elements from a composite volume.

Table 452 - SMI Referenced Properties/Methods for CIM_StorageElementCompositionService

Properties	Flags	Requirement	Description and Notes
GetAvailableElements()		Optional	This method queries the set of pools passed in and returns a set of volumes or logical disks that can be composed together based on the specified goal and element passed in.
GetCompositeElements()		Optional	Returns list of volumes/logical disks that were combined into this composite volume. Since (usually) all but one of these volumes/logical disks disappear when the composite is created, this is an essential method to help the client figure out what is in the composite. Remember that a particular client may not have been the one to create the composite.
GetSupportedCompositeStripeDepths()		Optional	This method returns the list of possible stripe depths (also known as stripe size) to use in the CreateOrModifyCompositeElement method.
GetSupportedCompositeStripeDepthRange()		Optional	This method returns the range of possible stripe depths (also known as stripe size) to use in the CreateOrModifyCompositeElement method.

24.9.9 CIM_StorageSetting

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 453 describes class CIM_StorageSetting.

Table 453 - SMI Referenced Properties/Methods for CIM_StorageSetting

Properties	Flags	Requirement	Description and Notes
InstanceID		Mandatory	
ElementName		Mandatory	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		Mandatory	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.
DataRedundancyMin		Mandatory	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.
DataRedundancyMax		Mandatory	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.
DataRedundancyGoal		Mandatory	

Table 453 - SMI Referenced Properties/Methods for CIM_StorageSetting (Continued)

Properties	Flags	Requirement	Description and Notes
PackageRedundancyMin		Mandatory	PackageRedundancyMin describes the minimum number of spindles or logical devices to be used. Package redundancy describes how many disk spindles or logical devices 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.
PackageRedundancyMax		Mandatory	PackageRedundancyMax describes the maximum number of spindles or logical devices to be used. Package redundancy describes how many disk spindles or logical devices 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.
PackageRedundancyGoal		Mandatory	
ExtentStripeLength		Optional	Number of underlying Storage volumes in a composite volume that data is striped across.
ExtentStripeLengthMin		Optional	ExtentStripeLengthMin describes the minimum acceptable stripe length.
ExtentStripeLengthMax		Optional	ExtentStripeLengthMax describes the maximum acceptable stripe length.
ParityLayout		Optional	ParityLayout describes the desired parity layout. The value may be 1 or 2 (Non-rotated Parity or Rotated Parity).
UserDataStripeDepth		Optional	The number of bytes forming a stripe (also known as stripe size).
UserDataStripeDepthMin		Optional	UserDataStripeDepthMin describes the minimum acceptable stripe depth.
UserDataStripeDepthMax		Optional	UserDataStripeDepthMax describes the maximum acceptable stripe depth.
ChangeableType		Mandatory	This property informs a client if the setting can be modified. It also tells the client how long this setting is expected to remain in the model. If the implementation allows it, the client can use the property to request that the setting's existence be not transient.
StorageExtentInitialUsage		Optional	The Usage value to be used when creating a new storage element.
StoragePoolInitialUsage		Optional	The Usage value to be used when creating a new storage pool.

24.9.10 CIM_StorageVolume

Created By: Extrinsic: ReturnElementToElements

Modified By: External

Deleted By: Extrinsic: CreateOrModifyCompositeElement, ReturnElementToElements

Requirement: Storage Volumes used as storage elements.

Table 454 describes class CIM_StorageVolume.

Table 454 - SMI Referenced Properties/Methods for CIM_StorageVolume

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	
BlockSize		Mandatory	
NumberOfBlocks		Mandatory	The number of blocks as reported by the hardware.
ConsumableBlocks		Mandatory	The number of usable blocks.
ExtentStatus		Mandatory	
OperationalStatus		Mandatory	

EXPERIMENTAL

IECNORM.COM : Click to view the full PDF of ISO/IEC 24775-4:2014

DEPRECATED

25 Volume Management Profile

25.1 Description

25.1.1 General

The host Volume Management (VM) Profile addresses block storage virtualization and presents virtual block devices to clients. The model represents virtualization for host volume management where LogicalDisks are exported.

A host volume manager is a software storage management subsystem that allows one to manage physical disks as logical devices called volumes. A volume is a logical device that appears to data management systems as a physical disk. Through support of RAID, the volume manager provides similar features as many disk arrays. Therefore, CIM administration of a volume manager is similar to that of an array. Embedded volume managers, like in a switch, should use the Virtualization Profile.

The Volume Management Profile uses existing classes from the Array Profile and Block Services Subprofile, and optionally uses the Host Discovered Resources Subprofile to bind with the disks in the host operating system.

25.1.2 Instance Diagram

Figure 108 shows an instance diagram for the Volume Management Profile.

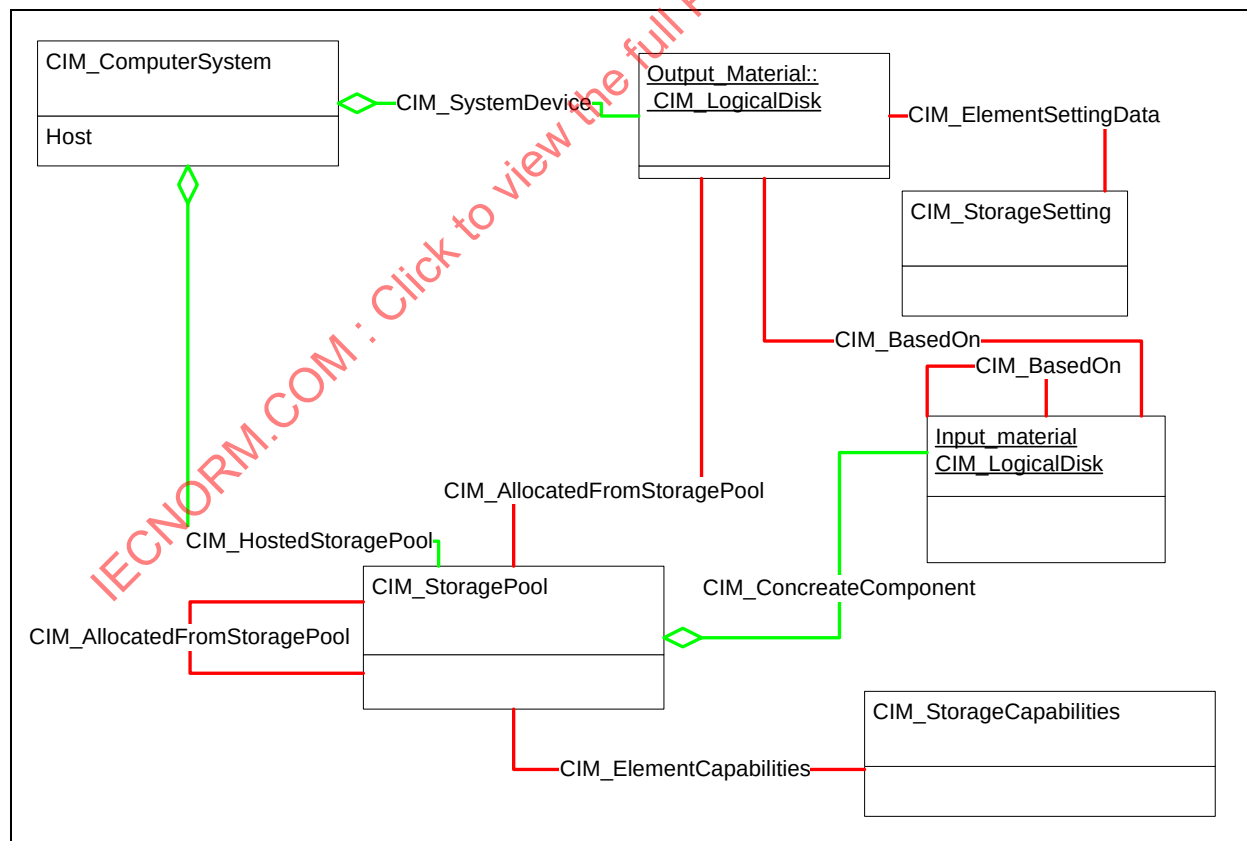


Figure 108 - Volume Management Instance Diagram

25.1.3 Input Class of the Volume Manager

The host operating system provides a unique name for each disk via a special file name. Typically, these are device file names: drive letters on Windows systems or `/dev/dsk/device1`

on UNIX systems. A LogicalDisk can be based on a disk partition or created by the operating system to represent a discovered volume and would have an operating system device name. The volume manager provider will place into a primordial pool all disks that it discovers as a LogicalDisk and uses the Name property to specify the operating system device file name.

25.1.4 Export Class of the Volume Manager

The Volume Management Profile exports LogicalDisk, which may be referred to as a volume in a typical host volume manager. For host volume managers, this is treated as a virtual disk or volume, and is where a file system or database would reside.

25.1.5 Initializing OS Disks for Volume Manager Use

All disks initially discovered by the volume manager from the host's device tree are added to a Primordial Pool by creating an association between the Primordial Pool and a LogicalDisk instance. Typically, these discovered disks are those listed in the /dev directory. Disks on a host are not immediately available for volume manager use. They are first initialized for volume manager use by writing metadata to the disk. Any disks that are not yet initialized for volume manager use will become initialized as a side effect of creating a concrete StoragePool.

25.1.6 Creating Pools and Logical Volumes

Concrete StoragePools are created by the Block Services CreateOrModifyStoragePool method.

Any uninitialized disks that are added to the concrete StoragePool are initialized as a side-effect of adding the disk to the pool.

The Block Services methods CreateOrModifyElementFromStoragePool or CreateOrModifyElementFromElements are used to create and modify volumes. When specifying a primordial pool or uninitialized disks to create or modify volumes, any disks that are not yet initialized will be initialized as a side effect of adding the disks to a concrete pool and creating the volume. See 6.1 Description for more details on methods for creating pools and logical volumes.

25.1.7 Storage Settings for Volumes

Providers need to map a Quality of Service and any Storage Settings to a particular volume's redundancy or raid level. This is similar to creating StorageVolumes in the Block Services Subprofile.

The StorageSetting, StorageSettingWithHints, and StorageCapabilities classes may be used to specify striping parameters such as number of stripe columns, or the extent stripe length. See Clause 6 "Block Services Package" for a description of these settings. The StorageSettingsDescription string should be updated with an appropriate string describing the volume's settings. The Exported value in ExtentStatus[] of the LogicalDisk should be set if it is intended for application use.

25.1.8 Durable Names and Other Correlatable ids of the Profile

Each object's Name in the volume manager is not durable. The names can be changed at any time. However, names will always be unique and correlatable. The provider will present names that the underlying volume manager software creates using its own naming heuristics. When available, the Host Discovered Resources Profile provides the connectivity and correlatable IDs of the host resources.

25.2 Health and Fault Management Considerations

Not defined in this International Standard.

25.3 Cascading Considerations

The Cascading Subprofile may be used when the Host Discovered Resources Profile is available on the host, where the Host Discovered Resource Profile would be the leaf profile. In this case, all discovered disks by the provider are still placed in the primordial pool. Therefore,

the behavior of what is in the primordial pool should not change based on the presence of another profile. The content should be consistent regardless of the presence of the Host Discovered Resources Profile. The description of the Cascading Subprofile for usage with the Security Resource Ownership Subprofile.

25.4 Supported Subprofiles and Packages

Table 455 describes the supported profiles for Volume Management.

Table 455 - Supported Profiles for Volume Management

Profile Name	Organization	Version	Requirement	Description
Access Points	SNIA	1.3.0	Optional	
Extent Composition	SNIA	1.5.0	Optional	
Location	SNIA	1.4.0	Optional	
Software	SNIA	1.4.0	Optional	
Disk Sparing	SNIA	1.5.0	Optional	
Job Control	SNIA	1.5.0	Optional	
Block Server Performance	SNIA	1.5.0	Optional	
Block Services	SNIA	1.5.0	Mandatory	
Health	SNIA	1.2.0	Mandatory	
Indication	SNIA	1.5.0	Mandatory	

25.5 Methods of the Profile

None.

25.6 Client Considerations and Recipes

25.6.1 General

Use Clause 6 Block Services Package to create and modify volumes.

See recipes for creating volumes in 6.6.6 "Conditional: Create StoragePool and Storage Element on Block Server (e.g., Array or Volume Manager)" in Clause 6 Block Services Package.

Replacing a disk is done by using the Sparing Subprofile. Newly added disks are first made and then are used to replace the old disk.

25.6.2 Storage Configuration

The Volume Management Profile uses the StorageConfigurationService in the Block Services Subprofile for creating and modifying objects in a StoragePool. Creating volumes with specified extents shall be done using the CreateorModifyElementFromElements method. When specifying extents, or when using the InExtents[] parameter of CreateOrModifyStoragePool for creating storage pools as well as adding disks, then the specified extents shall be from among the extents returned from the StoragePool.GetAvailableExtents method. Any other extents may cause the operation to fail.

25.7 Registered Name and Version

Volume Management version 1.2.0 (Autonomous Profile).

25.8 CIM Elements

25.8.1 General

Table 456 describes the CIM elements for Volume Management.

Table 456 - CIM Elements for Volume Management

Element Name	Requirement	Description
25.8.2 CIM_AllocatedFromStoragePool (LogicalDisk from Pool)	Mandatory	
25.8.3 CIM_AllocatedFromStoragePool (Pool from Pool)	Mandatory	
25.8.4 CIM_ComputerSystem	Mandatory	Associated to RegisteredProfile.
25.8.5 CIM_ElementCapabilities	Mandatory	
25.8.6 CIM_ElementSettingData	Mandatory	
25.8.7 CIM_HostedStoragePool	Mandatory	
25.8.8 CIM_LogicalDisk	Mandatory	
25.8.9 CIM_StorageCapabilities	Mandatory	
25.8.10 CIM_StoragePool (Concrete)	Mandatory	Logical Disks are allocated from 'concrete' pools.
25.8.11 CIM_StoragePool (Primordial)	Mandatory	At least one primordial pool shall exist for a host. This is the 'unallocated storage' of the host, and contains unused disks.
25.8.12 CIM_StorageSetting	Mandatory	
25.8.13 CIM_SystemDevice	Mandatory	
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_LogicalDisk	Mandatory	Addition of a new logical disk instance.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_LogicalDisk	Mandatory	Deletion of a logical disk instance.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_LogicalDisk AND SourceInstance.OperationalStatus <> PreviousInstance.OperationalStatus	Mandatory	Change of status of a Logical Disk.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_LogicalDisk AND SourceInstance.CIM_LogicalDisk::OperationalStatus <> PreviousInstance.CIM_LogicalDisk::OperationalStatus	Mandatory	CQL - Change of status of a Logical Disk.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_StoragePool AND SourceInstance.CIM_StoragePool::OperationalStatus <> PreviousInstance.CIM_StoragePool::OperationalStatus	Mandatory	CQL - Change of status of a storage pool.
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_StoragePool	Mandatory	Addition of a storage pool instance.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_StoragePool	Mandatory	Deletion of a storage pool instance.

25.8.2 CIM_AllocatedFromStoragePool (LogicalDisk from Pool)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 457 describes class CIM_AllocatedFromStoragePool (LogicalDisk from Pool).

Table 457 - SMI Referenced Properties/Methods for CIM_AllocatedFromStoragePool (LogicalDisk from Pool)

Properties	Flags	Requirement	Description and Notes
SpaceConsumed		Mandatory	
Antecedent		Mandatory	
Dependent		Mandatory	

25.8.3 CIM_AllocatedFromStoragePool (Pool from Pool)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 458 describes class CIM_AllocatedFromStoragePool (Pool from Pool).

Table 458 - SMI Referenced Properties/Methods for CIM_AllocatedFromStoragePool (Pool from Pool)

Properties	Flags	Requirement	Description and Notes
SpaceConsumed		Mandatory	
Dependent		Mandatory	
Antecedent		Mandatory	

25.8.4 CIM_ComputerSystem

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Shall be associated to RegisteredProfile using ElementConformsToProfile association. The RegisteredProfile instance shall have RegisteredName set to 'Volume Management', RegisteredOrganization set to 'SNIA', and RegisteredVersion set to '1.2.0'.

Table 459 describes class CIM_ComputerSystem.

Table 459 - SMI Referenced Properties/Methods for CIM_ComputerSystem

Properties	Flags	Requirement	Description and Notes
CreationClassName		Mandatory	
Name		Mandatory	Unique identifier for the Host. IP address.
ElementName		Mandatory	User friendly name.
OperationalStatus		Mandatory	Overall status of the Host.
NameFormat		Mandatory	Format for Name property.

Table 459 - SMI Referenced Properties/Methods for CIM_ComputerSystem

Properties	Flags	Requirement	Description and Notes
Dedicated		Mandatory	This should include 0. This indicates that this computer system is not dedicated to volume management.
PrimaryOwnerContact		Optional	
PrimaryOwnerName		Optional	

25.8.5 CIM_ElementCapabilities

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 460 describes class CIM_ElementCapabilities.

Table 460 - SMI Referenced Properties/Methods for CIM_ElementCapabilities

Properties	Flags	Requirement	Description and Notes
Capabilities		Mandatory	
ManagedElement		Mandatory	

25.8.6 CIM_ElementSettingData

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 461 describes class CIM_ElementSettingData.

Table 461 - SMI Referenced Properties/Methods for CIM_ElementSettingData

Properties	Flags	Requirement	Description and Notes
SettingData		Mandatory	
ManagedElement		Mandatory	

25.8.7 CIM_HostedStoragePool

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 462 describes class CIM_HostedStoragePool.

Table 462 - SMI Referenced Properties/Methods for CIM_HostedStoragePool

Properties	Flags	Requirement	Description and Notes
PartComponent		Mandatory	
GroupComponent		Mandatory	

25.8.8 CIM_LogicalDisk

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 463 describes class CIM_LogicalDisk.

Table 463 - SMI Referenced Properties/Methods for CIM_LogicalDisk

Properties	Flags	Requirement	Description and Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	Opaque identifier.
ElementName		Mandatory	User friendly name.
Name		Mandatory	Should be a durable name. As yet any name.
ExtentStatus		Mandatory	
OperationalStatus		Mandatory	
BlockSize		Mandatory	
NumberOfBlocks		Mandatory	
IsBasedOnUnderlyingRedundancy		Mandatory	
NoSinglePointOfFailure		Mandatory	
DataRedundancy		Mandatory	
PackageRedundancy		Mandatory	
DeltaReservation		Mandatory	
ConsumableBlocks		Mandatory	

25.8.9 CIM_StorageCapabilities

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 464 describes class CIM_StorageCapabilities.

Table 464 - SMI Referenced Properties/Methods for CIM_StorageCapabilities

Properties	Flags	Requirement	Description and Notes
InstanceID		Mandatory	
ElementName		Mandatory	
NoSinglePointOfFailure		Mandatory	
NoSinglePointOfFailureDefault		Mandatory	
DataRedundancyMin		Mandatory	
DataRedundancyMax		Mandatory	
DataRedundancyDefault		Mandatory	
PackageRedundancyMin		Mandatory	
PackageRedundancyMax		Mandatory	
PackageRedundancyDefault		Mandatory	
DeltaReservationDefault		Mandatory	
DeltaReservationMax		Mandatory	
DeltaReservationMin		Mandatory	

25.8.10 CIM_StoragePool (Concrete)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 465 describes class CIM_StoragePool (Concrete).

Table 465 - SMI Referenced Properties/Methods for CIM_StoragePool (Concrete)

Properties	Flags	Requirement	Description and Notes
InstanceID		Mandatory	
ElementName		Mandatory	
PoolID		Mandatory	
TotalManagedSpace		Mandatory	
RemainingManagedSpace		Mandatory	
Primordial		Mandatory	Set to false.

25.8.11 CIM_StoragePool (Primordial)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 466 describes class CIM_StoragePool (Primordial).

Table 466 - SMI Referenced Properties/Methods for CIM_StoragePool (Primordial)

Properties	Flags	Requirement	Description and Notes
InstanceID		Mandatory	
ElementName		Mandatory	
PoolID		Mandatory	
TotalManagedSpace		Mandatory	
RemainingManagedSpace		Mandatory	
Primordial		Mandatory	Set to true.

25.8.12 CIM_StorageSetting

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 467 describes class CIM_StorageSetting.

Table 467 - SMI Referenced Properties/Methods for CIM_StorageSetting

Properties	Flags	Requirement	Description and Notes
InstanceID		Mandatory	Opaque identifier.
ElementName		Mandatory	User friendly name.... can be used for 'potted' settings for specific RAID levels.
DataRedundancyMin		Mandatory	
DataRedundancyMax		Mandatory	
DataRedundancyGoal		Mandatory	
PackageRedundancyMin		Mandatory	
PackageRedundancyMax		Mandatory	
PackageRedundancyGoal		Mandatory	
DeltaReservationGoal		Mandatory	
DeltaReservationMax		Mandatory	
DeltaReservationMin		Mandatory	

25.8.13 CIM_SystemDevice

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 468 describes class CIM_SystemDevice.

Table 468 - SMI Referenced Properties/Methods for CIM_SystemDevice

Properties	Flags	Requirement	Description and Notes
PartComponent		Mandatory	
GroupComponent		Mandatory	

DEPRECATED

IECNORM.COM : Click to view the full PDF of ISO/IEC 24775-4:2014

EXPERIMENTAL

26 Storage Element Protection SubProfile

26.1 Description

26.1.1 Overview

The Storage Element Protection Subprofile defines classes and methods for managing access permission to a storage element—either a storage volume or logical disk. This subprofile also defines how long the protection shall stay in effect. It allows a client to protect data as required by changeable business and operational policies. Clients may modify access to a storage element for various reasons, including:

- *Regulatory Compliance* - Ensure that vital records are available, unaltered (immutable) and protected from accidental or malicious destruction. The degree of exposure and the retention period depend on the nature of the records.
- *Protection of Fixed Content* - Maintain in "Read-only" mode between cyclic refreshes of the data content.
- *Protection of Recovery Assets* - Protect data from accidental reuse. For example, make recovery logs "Read-only" or immutable.
- *Reclamation of Expired (Archive) Capacity* - After migration, delete or destroy data when elements are released for re-use.

26.1.2 Use Cases

In a typical scenario, a storage element is allocated with Read/Write permission. At a later time, when the element holds data that requires protection, the access permission is changed to Read-only with a retention period.

Changes in regulations, audit or litigation may require that the storage element be retained for a longer period. In this case, the retention period may be extended or alternatively set to a "never to expire" value. This new setting retains the current protection for an indefinite period – until litigation is resolved, for example.

Company policy may dictate that archived data, although still protected and retained for legal purposes, be unavailable even for Read-only. In this case, the element may be hidden from read-and-write access. It will be visible only to a storage administrator.

26.1.3 Functionality

A management application will interact with this subprofile in two ways—(1) the management application can retrieve and modify the access permission attribute and (2) the management application may define the period for which the access permission will remain in effect (the retention period). During the retention period, other functions shall be disabled to prevent the storage element from being reformatted, erased or otherwise (logically) destroyed. While this retention period is in effect, the access permission cannot be modified except to make it more restrictive. Once this period expires, the access permissions remain in effect, but they may now be modified. The management application may extend this retention period but shall not be able to shorten it.

26.1.4 Class Model

In order to support the desired protection functionality, this profile defines a new method, `Protect`, for the `StorageProtectionService` class. This method allows the client to set the protection-related configurations of a storage element, either a `StorageVolume` or `LogicalDisk`. When first called for a storage element, it creates a `StorageProtectionSetting` instance with the client requested configuration and associates it to the target element by the `ElementProtectionSettingData` association. If the target element already has a

StorageProtectionSetting associated via ElementProtectionSettingData, then it modifies the properties of the existing instance of StorageProtectionSetting, as shown in Figure 109: "Storage Element Protection Class Model". After the retention period has expired and every

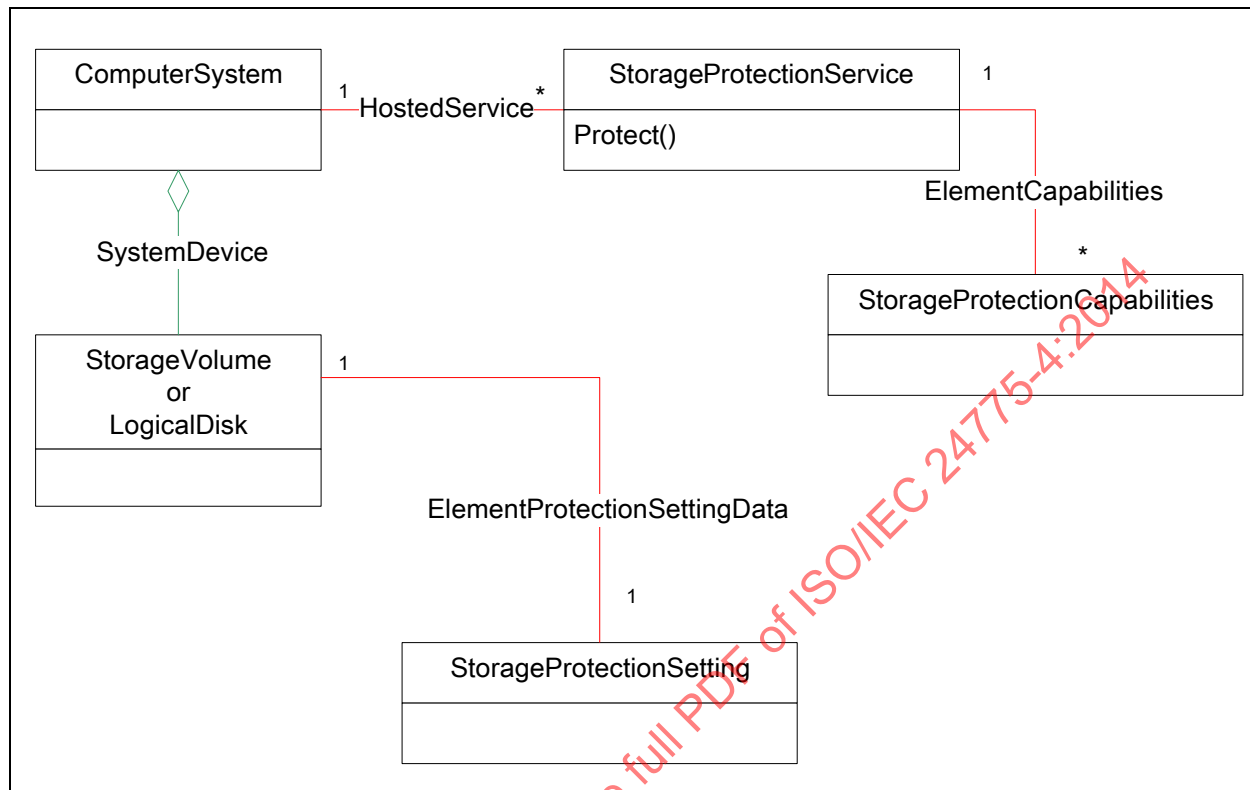


Figure 109 - Storage Element Protection Class Model

protection configuration has been released, the StorageProtectionSetting instance will not automatically be removed by the instrumentation. However, a state change indication will be sent to the management application so that it may remove the instance by using the DeleteInstance operation if needed.

Table 469 shows properties this profile defines for the StorageProtectionCapabilities class, which indicates the capability of the element protection feature of the associated StorageProtectionService, including the granularity of the retention period.

Table 469 - Properties for StorageProtectionCapabilities

Property	Flags	Type	Descriptions and Notes
ProtectionTimeGranularity		uint16	Granularity for the time period of StorageProtectionSetting.RemainingProtectionTime. Possible values are: 0 (Unknown), 1 (Other), 2 (Second), 3 (Minute), 4 (Hour), 5 (Day)
SupportedStorageElementFeatures		uint16[]	Enumeration indicating which storage elements can be protected. Possible values: 1 - StorageVolume Protection 2 - LogicalDisk Protection
SupportedSynchronousActions		uint16[]	Methods that will not create a job. Possible values: 1 - Storage Element Protection
SupportedAsynchronousActions		uint16[]	Methods that will create a job. Possible values: 1 - Storage Element Protection

This profile also defines a new Setting class, `StorageProtectionSetting`, which contains the protection-related properties for a particular `StorageVolume` or `LogicalDisk` storage element, shown in Table 470. This class is associated to a storage element instance via the `ElementProtectionSettingData` association. A client can retrieve the protection-related configurations and statuses of a `StorageVolume` or `LogicalDisk` by traversing the `ElementProtectionSettingData` association if it exists. If that association is not found, no protection management is applied for the `StorageVolume` or `LogicalDisk`.

Table 470 - Properties for `StorageProtectionSetting`

Property	Flags	Type	Descriptions and Notes
<code>ProtectionControlled</code>		boolean	Whether the storage element is under protection control or not. If this property is FALSE that indicates the storage device has protection feature or used to have, but currently the service has been withdrawn or is not available to obtain protection attributes by some accident.
<code>Access</code>		uint16	Read and write accessibility of the storage element. 1: Read/Write Enabled 2: Read Only 3: Write Once 4: Read/Write Disabled While it is not possible to use <code>Protect()</code> to transition to "Write Once", it is still needed for correct reporting of status.
<code>InquiryProtection</code>		Uint16[]	Protected responses for SCSI inquiry commands. 1: No SCSI Inquiry Protection 2: Inquiry Disabled 3: Zero Capacity Returned This property is utilized in the protection of a <code>StorageVolume</code> and it is optional to implement.
<code>DenyAsCopyTarget</code>		boolean	Whether the storage element can be specified as a copy target or not. If this property is TRUE then this storage element will not be selectable as a target of copy pair.
<code>LUNMappingConfigurable</code>		boolean	Whether LU assignment to the storage element is configurable or not. This property is utilized in the protection of a <code>StorageVolume</code> and is optional to implement.
<code>ProtectExpirationSpecified</code>		uint16	Duration type of the storage element protection. 1: None 2: Limited Expiration 3: Permanent.
<code>RemainingProtectionTime</code>		datetime	Amount of remaining time before a management application can change the access permission.

26.1.5 Access permission

The overall state of the `StorageVolume` or `LogicalDisk` protection is indicated by the combination of several properties. Table 471, Table 472, Table 473, Table 474, and Table 475

show the possible values of each property listed in Table 470. These tables apply to properties in the StorageProtectionSetting class.

Table 471 - Values for ProtectionControlled

Value	Description
TRUE	Storage element is under protection control.
FALSE	Storage element is NOT under protection control.

Table 472 - Values for Access

Value	Description
0 (Unknown)	Accessibility status is unknown.
1 (Read/Write Enabled)	Both read and write commands are allowed.
2 (Read Only)	Read command is allowed; write command is prohibited.
3 (Write Once)	Read command is allowed; overwrite command is prohibited.
4 (Read/Write Disabled)	Both read and write commands are prohibited.

Table 473 - Values for InquiryProtection

Value	Description
0 (Unknown)	Status is unknown.
1 (No SCSI Inquiry Protection)	Protection method by the SCSI inquiry commands is not performed.
2 (Inquiry Disabled)	All SCSI inquiry commands are rejected.
3 (Zero Capacity Returned)	Size 0 is returned as a reply of SCSI read capacity command

Table 474 - Values for DenyAsCopyTarget

Value	Description
TRUE	Storage element can not be specified as a copy target.
FALSE	Storage element can be specified as a copy target.

Table 475 - Values for LUNMappingConfigurable

Value	Description
TRUE	LU assignment to the storage volume is configurable.
FALSE	LU assignment to the storage volume is not configurable.

26.1.6 Retention period

The Retention period (the amount of time that the settings are to remain locked) is also indicated by the combination of several properties. Table 476 and Table 477 show the meaning of each property value. These tables apply to properties in the StorageProtectionSetting class.

Table 476 - Values for ProtectExpirationSpecified

Value	Description
0 (Unknown)	Status is unknown.
1 (None)	The protection duration is not specified.
2 (Limited Expiration)	The protection expires after the time period.
3 (Permanent)	The protection is permanent.

Table 477 - Values for RemainingProtectionTime

Value	Description
datetime	Amount of remaining time before a management application can change the access permission. It is a dynamic value which keeps decreasing by the time progress until it reaches the datetime equivalent of 0. The value will be decreased by the time period indicated by the StorageProtectionCapabilities.ProtectionTimeGranularity property.

There are two ways to designate the duration of access permission, shown in Figure 110: "Retention Time Line":

- Expiration Date - Defines a future date/time when access permission may be modified.
- Remaining Retention Period - Defines the remaining length of time for access permission.

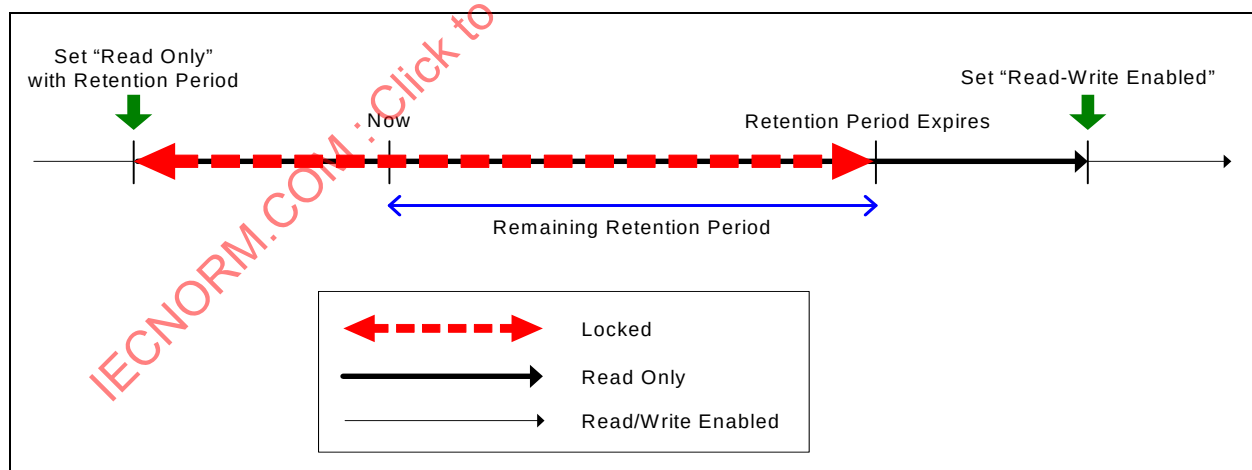


Figure 110 - Retention Time Line

The use of an *Expiration Date* requires a reference to an agreed-upon reference clock. Without a trusted external date/time reference, the retention period will be open to spoofing, conflicts between individual component clocks (e.g., server and storage) and time zones issues. The inevitable nuances of individual implementations may require variations in the client application.

The use of *Remaining Retention Period* does not require a reference clock. There is no question of interpretation of whether or when the retention period will expire - it is either zero (expired) or not. The implementation is the responsibility of the provider and is hidden from the client.

Providers may implement the retention function that works best for that provider, while remaining interoperable.

26.1.7 Protection State Transition

Figure 111: "Protection State Transition Diagram" shows storage element protection state transition. When the retention period is not specified or expired, the storage element may transition to any state except *Write Once* permission by using the Protect method. Once a retention period is specified to a storage element, it may transition to a more restricted state only via the Protect method. It may transition to the other states only when the retention period has expired. Generally, a storage element starts with a protection state of "Access = Read/Write Enabled, Retention = None/Expired" and Protect is used to set the protection to be more restrictive. If the storage element is write-once media such as a CD-ROM it will have a protection state of "Access = Write Once, Retention = Permanent".

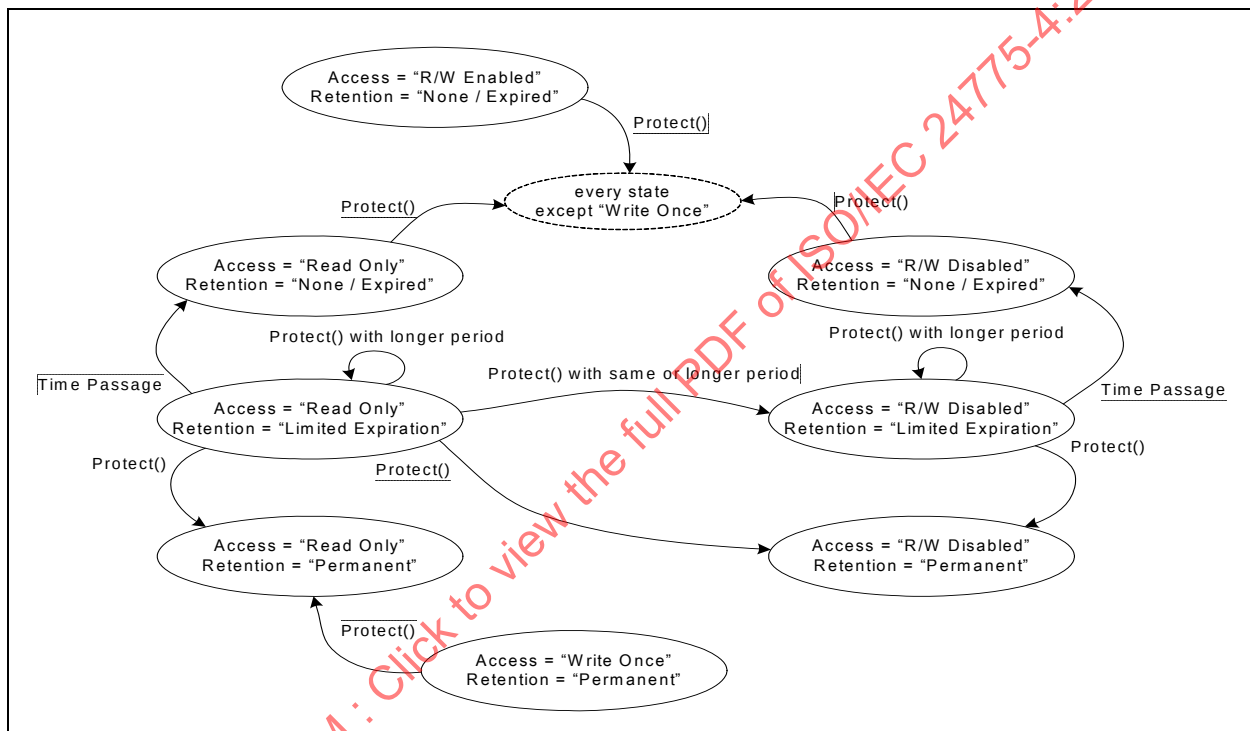


Figure 111 - Protection State Transition Diagram

26.1.8 Sample Usage Scenario

26.1.8.1 General

Figure 112: "Step 1 - Initial State", Figure 113: "Step 2 - Volume Set to Read-only", Figure 114: "Step 3 - Second Volume Set to Read-only", Figure 115: "Step 4 - Volume Set to Read/Write Disabled", and Figure 116: "Step 5 Volume Access Changed" show the progression of a typical usage scenario for StorageVolume protection.

26.1.8.2 Step 1: StorageVolume not protected

Figure 112: "Step 1 - Initial State" shows the initial state of a StorageVolume that does not have protection enabled yet. In this situation, no instance of StorageProtectionSetting exists. However, it shows that the instrumentation has the capability to support the setting of the element protection properties because the StorageProtectionCapabilities SupportedStorageElementFeatures property includes the value 1 (StorageVolume Protection) and the SupportedAsynchronousActions property includes the value 1 (Storage Element Protection). The StorageProtectionCapabilities instance also has a value of 5 (Day) for the ProtectionTimeGranularity property which indicates the retention period specified on this device will be decreased by the granularity of a day.

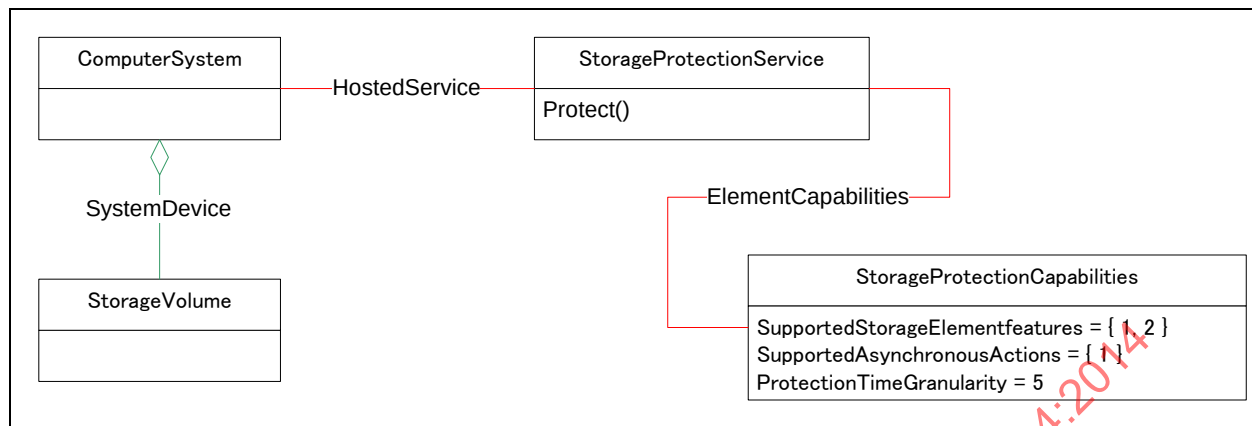


Figure 112 - Step 1 - Initial State

26.1.8.3 Step 2: Volume Set to Read-only

In Figure 113: "Step 2 - Volume Set to Read-only", the **StorageVolume** is set to Read-only permission for a specific period of time. In this example, there are two **StorageVolumes**, 'V1' and 'V2'. By using the **Protect()** method of **StorageProtectionService**, volume 'V1' is set to Read-only access permission and a 365-day retention period. This operation creates new instance of **StorageProtectionSetting** ('SPS1') and associates it with the target **StorageVolume** 'V1'. After the **Protect** method completes, the **Access** property is now set to the value 2 (Read Only), and the **RemainingProtectionTime** is set to the value of 365 days.

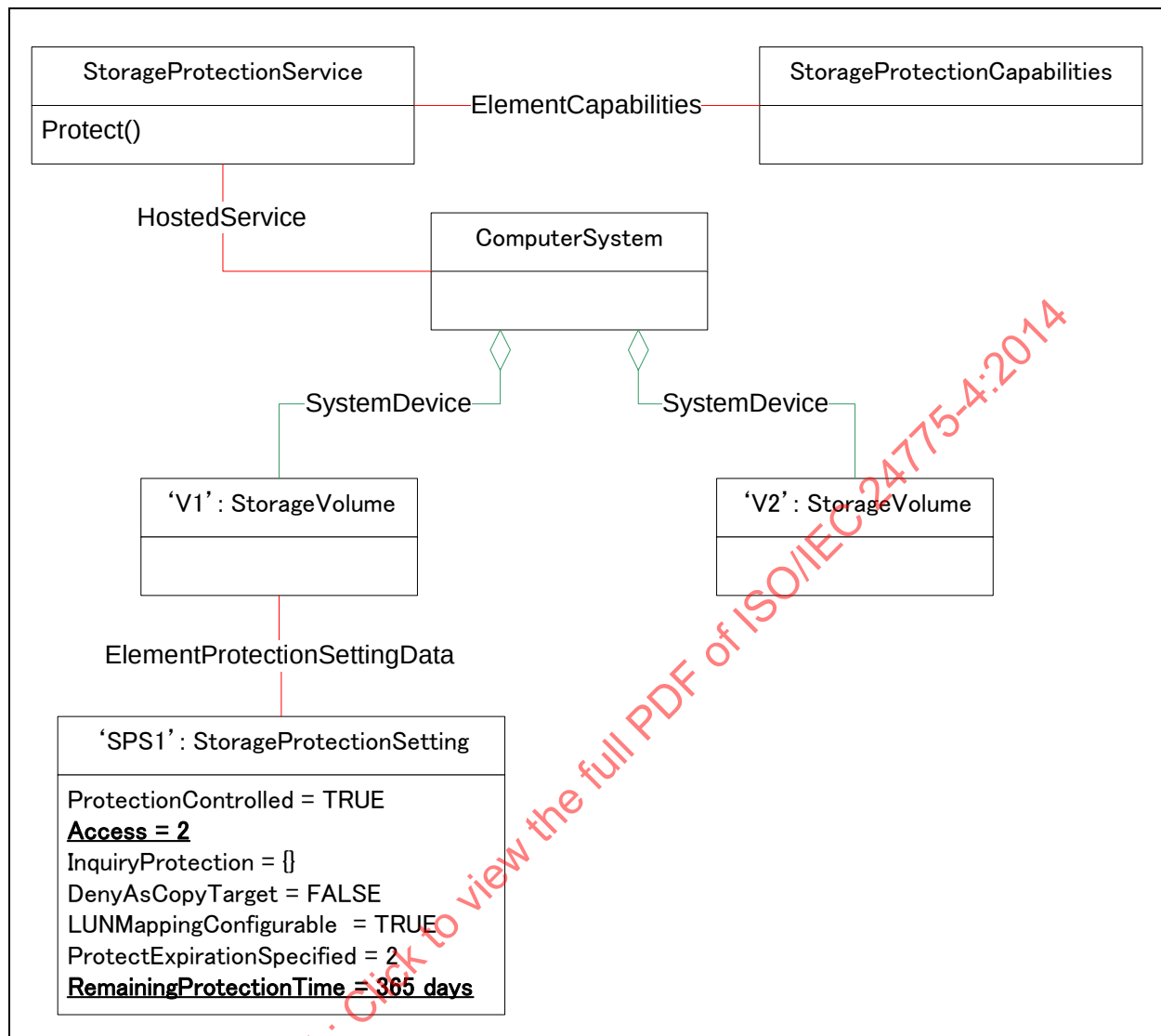


Figure 113 - Step 2 - Volume Set to Read-only

26.1.8.4 Step 3: Second Volume Set to Read-only

Figure 114: "Step 3 - Second Volume Set to Read-only" shows Set Read-only permission to another StorageVolume 'V2' after some amount of time.

After 30 days, the client decides to protect StorageVolume 'V2' by setting it to Read-only with a retention time of 365 days, same as 'V1'. A new instance of StorageProtectionSetting is created by the instrumentation to the target StorageVolume 'V2'. A single StorageProtectionSetting instance will not be shared because it has a different RemainingProtectionTime although both are configured with the same access permission.

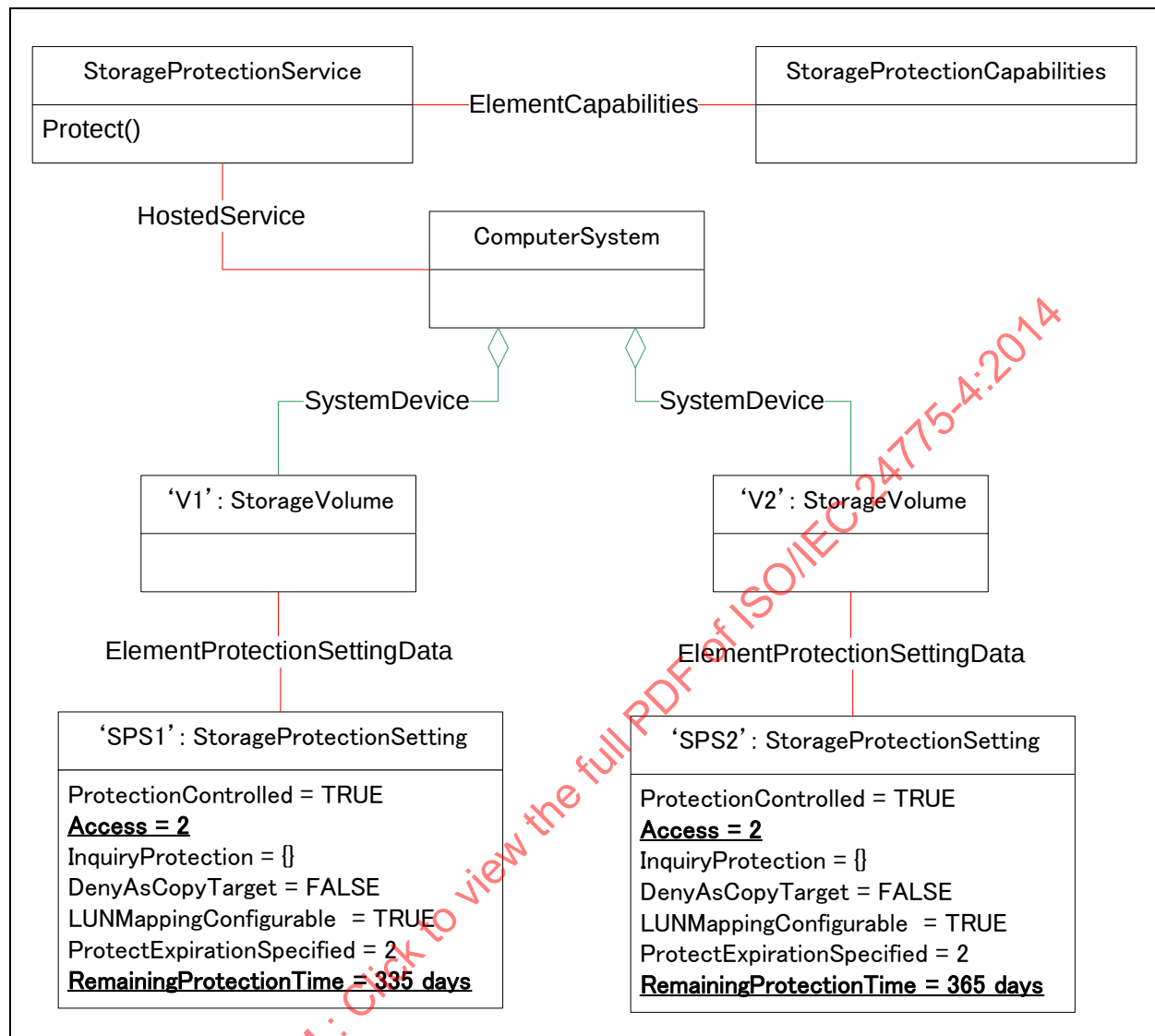


Figure 114 - Step 3 - Second Volume Set to Read-only

26.1.8.5 Step 4: Volume Set to Read/Write Disabled

Figure 115: "Step 4 - Volume Set to Read/Write Disabled" shows access permission of StorageVolume 'V1' changed to Read/Write Disabled.

Within the retention period, the access permission may not be changed except to be made more restricted. Because StorageVolume 'V1' was set to Read-only permission, it is possible to modify it to Read/Write Disabled permission within its retention period because this setting is more restrictive than Read-only.

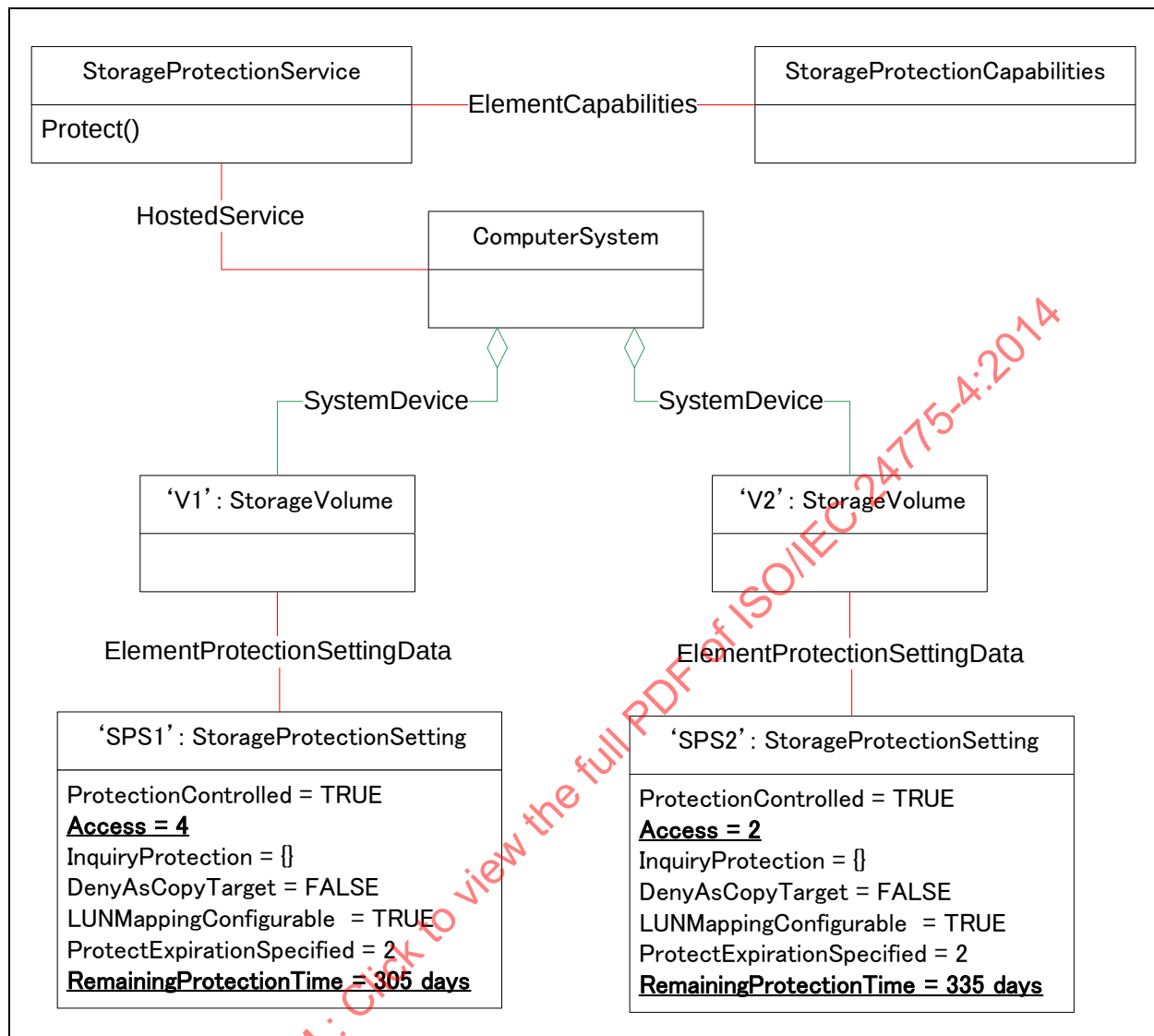


Figure 115 - Step 4 - Volume Set to Read/Write Disabled

26.1.8.6 Step 5: Volume Access Change

Figure 116: "Step 5 Volume Access Changed" shows change of access permission of StorageVolume 'V1' to "Read/Write Enabled" after expiration.

After the passage of the specified time, the retention period of StorageVolume will expire. Therefore, its access permission can be modified to any level. The StorageProtectionSetting instance is not automatically deleted when the retention period has expired. The StorageVolume maintains its access permission configuration.

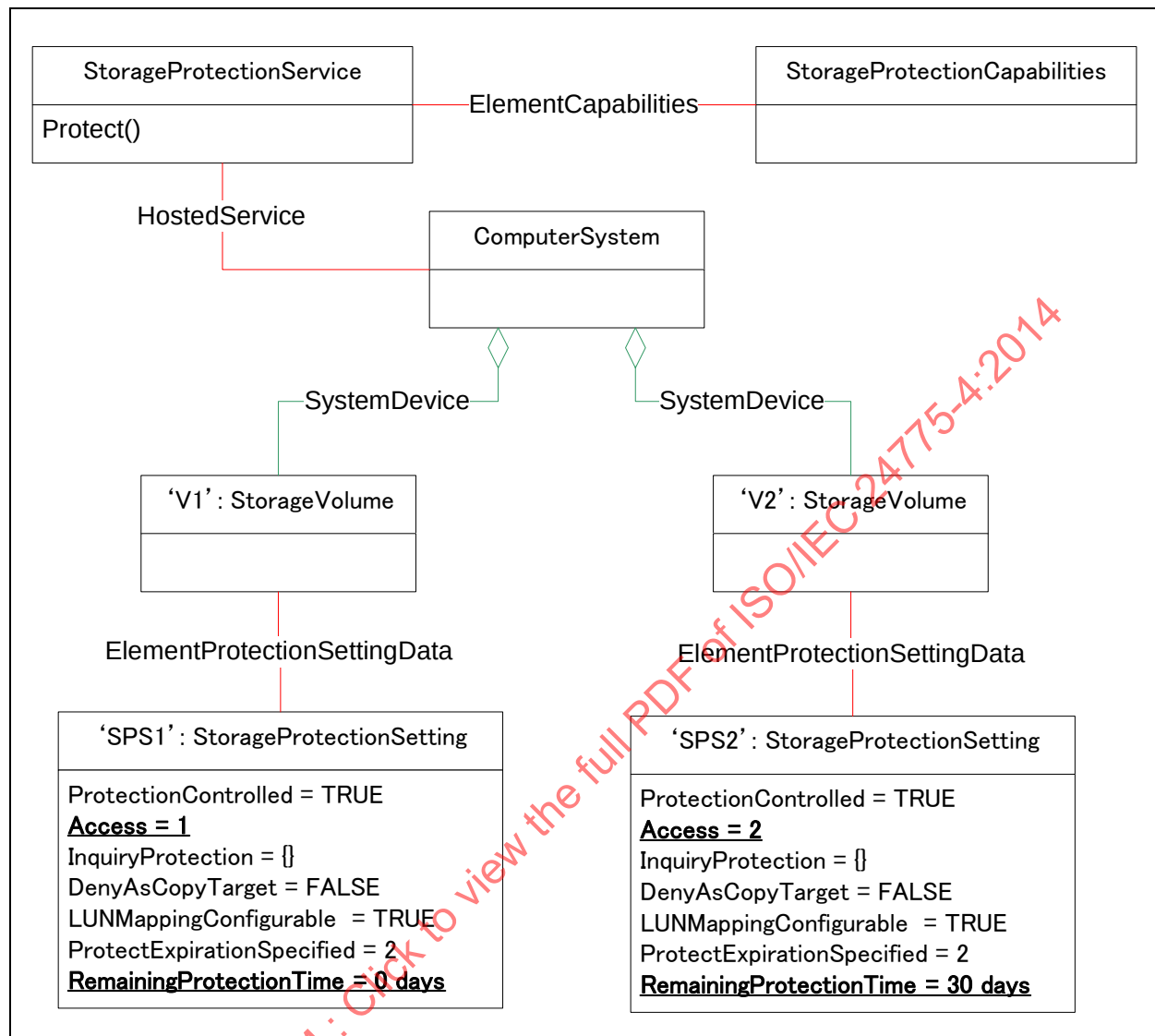


Figure 116 - Step 5 Volume Access Changed

26.2 Health and Fault Management Consideration

Not defined in this International Standard.

26.3 Cascading Considerations

Not applicable.

26.4 Supported Profiles, Subprofiles, and Packages

Related Profiles for Storage Element Protection: Not defined in this International Standard.

26.5 Methods of the Profile – Protect

This method, defined in Table 478, is found in the StorageProtectionService. It configures the protection attributes of StorageVolumes and LogicalDisks, which prevents them from being modified for a specific period of time. Values specified for this method shall be set as properties of the StorageProtectionSetting instance that is associated to the specified StorageVolume or LogicalDisk. This method can be used to extend the retention period, but not decrease it. The instrumentation shall always create a new instance of

StorageProtectionSetting when protection is first applied, but it shall reuse the existing setting when modifying the protection setting.

Table 478 - Methods of the Storage Element Protection Profile

Method: Protect			
Return Values:			
Value		Description	
0: Success		Method completed with no error.	
1: Not Supported		Method is not supported.	
2: Unspecified Error		Unspecified error.	
3: Timeout		Timeout happened during processing.	
4: Failed		Method failed.	
5: Invalid Parameter		Specified parameter is not allowed.	
6: Invalid State Transition		Specified access permission or retention period is not allowed in the current status.	
4096: Method parameters checked - job started		A Job was started.	
Errors:			
Not defined in this International Standard			
Parameters:			
Qualifiers	Name	Type	Description/Values
OUT	Job	CIM_Job REF	Reference to the job created, if any.
IN	Element	CIM_StorageExtent REF	StorageVolume or LogicalDisk to be configured.
IN	ElementType	uint16	The type of element being protected. 1: StorageVolume 2: LogicalDisk.
IN	Access	uint16	Read and write accessibility of the storage element. 1: Read/Write Enabled 2: Read Only 4: Read/Write Disabled Note that it is not possible to transition to "3: Write Once" from other state.
IN	InquiryProtection	uint16[]	The inquiry protection method for SCSI inquiry commands. 1: No SCSI Inquiry Protection 2: Inquiry Disabled 3: Zero Capacity Returned This may be specified when protecting a StorageVolume.

Table 478 - Methods of the Storage Element Protection Profile

Method: Protect			
IN	DenyAsCopyTarget	boolean	Whether the storage element can be specified as a copy target or not. If this property is TRUE then the storage element will not be selectable as a target of copy pair.
IN	LUNMappingConfigure	boolean	Whether LU assignment to the StorageVolume is configurable or not. This may be specified when protecting a StorageVolume.
IN	ProtectExpirationType	uint16	Duration type of the storage element protection. 1: None 2: Limited Expiration 3: Permanent.
IN	TimePeriod	datetime	Amount of remaining time before a management application can change the access permission.

26.6 Client Considerations and Recipes

26.6.1 Start Volume Protection

In this use case, a StorageVolume is used to store business data that needs to be protected from overwriting. There is a regulation that this kind of data should be held for three years. Therefore, a management application requests the instrumentation to set the StorageVolume for Read-only access permission and a three-year retention period.

```
// DESCRIPTION
//
// Start StorageVolume protection
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
//
// 1. Reference to the CIM_StorageVolume to be protected
// has been found and the object path value is stored in
// $StorageVolume->
// 2. Reference to the SNIA_StorageProtectionService
// has been found and the object path value is stored in
// $StorageProtectionService->
// 3. The SNIA_StorageProtectionCapabilities for this service
// has been found and the instance value is stored in
// $StorageProtectionCapabilities
// 4. Variable #ThreeYearsValue and #TwoYearsValue are the
// type of datetime and have the value of a three and two year
// time period, respectively

// Check for the capability
#SupportedFeatures[] =
    $StorageProtectionCapabilities.SupportedStorageElementFeatures
```

```
if (contains(1, #SupportedFeatures) == false) {
    <ERROR! StorageVolume protection feature is not supported>
}

// Invoke the protection method.
%InputArguments["Element"]           = $StorageVolume->
%InputArguments["ElementType"]       = 1// StorageVolume
%InputArguments["Access"]            = 2// Read Only
%InputArguments["ProtectExpirationType"] = 2// Limited Expiration
%InputArguments["TimePeriod"]       = #ThreeYearsValue// 3 years

#ReturnCode = InvokeMethod(
    $StorageProtectionService->,
    "Protect",
    %InputArguments,
    %OutputArguments )

if (#ReturnCode != 0) {
    <ERROR! CIM_StorageProtectionSetting has not been created.>
}

// Verify the protection setting.
$StorageProtectionSettings->[] = Associators(
    $StorageVolume->,
    "SNIA_ElementProtectionSettingData",
    "SNIA_StorageProtectionSetting",
    "Dependent",
    "Antecedent",
    false, false, NULL )

for #i in $StorageProtectionSettings->[] { // should be only one item.
    if( $StorageProtectionSetting->[#i].ProtectionControlled == false ) {
        <ERROR! CIM_StorageVolume is not under protection controlled.>
    }

    if( $StorageProtectionSetting->[#i].Access == 2 &&
        $StorageProtectionSetting->[#i].ProtectExpirationSpecified == 2
        &&
        $StorageProtectionSetting->[#i].RemainingProtectionTime >
            #TwoYearsValue &&
        $StorageProtectionSetting->[#i].RemainingProtectionTime <=
            #ThreeYearsValue )
    {
        <EXIT: StorageVolume Protection configuration successful>
    }
}

// if we get to this point, it was not set
<ERROR! StorageProtectionSetting has not been created.>
```

```
// end of the recipe
```

26.6.2 Extend the Retention Period

In this use case, a regulation has changed and the business data now need to be held for five years. A management application retrieves the value of the RemainingProtectionTime property of StorageProtectionSetting instance, which is associated to the target StorageVolume, and calculates the new value by adding two years to it. Then the application uses the StorageProtectionService.Protect() method to configure a new retention period.

```
// DESCRIPTION
//
// Extend the retention period
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
//
// 1. Reference to the CIM_StorageVolume to be protected
//   has been found and the object path value is stored in
//   $StorageVolume->
// 2. Reference to the SNIA_StorageProtectionService
//   has been found and the object path value is stored in
//   $StorageProtectionService->
// 3. The SNIA_StorageProtectionCapabilities for this service
//   has been found and the instance value is stored in
//   $StorageProtectionCapabilities
// 4. Variable #TwoYearsValue is the type of datetime and
//   has two year period of value.

// Check for the capability
#SupportedFeatures[] =
    $StorageProtectionCapabilities.SupportedStorageElementFeatures

if (contains(1, #SupportedFeatures) == false) {
    <ERROR! StorageVolume protection feature is not supported>
}

// Get current value for remaining protection time.
$StorageProtectionSettings[] = Associators(
    $StorageVolume->,
    "SNIA_ElementProtectionSettingData",
    "SNIA_StorageProtectionSetting",
    "Dependent",
    "Antecedent",
    false, false, NULL )

for #i in $StorageProtectionSettings[] { // should be only one item.
    if( $StorageProtectionSetting[#i].ProtectionControlled == false ) {
        <ERROR! CIM_StorageVolume is not under protection controlled.>
    }
    #RemainingProtectionTime =
        $StorageProtectionSetting[#i].RemainingProtectionTime
```

```
}

// Invoke the protection method.
// Set the time to protect the StorageVolume to the current time
    remaining + 2 more years
%InputArguments["Element"]           = $StorageVolume->
%InputArguments["ElementType"]       = 1// StorageVolume
%InputArguments["Access"]            = 2// Read Only
%InputArguments["ProtectExpirationType"] = 2// Limited Expiration
%InputArguments["TimePeriod"]        = #RemainingProtectionTime +
    #TwoYearsValue

#ReturnCode = InvokeMethod(
    $StorageProtectionService->,
    "Protect",
    %InputArguments,
    %OutputArguments )

if (#ReturnCode != 0) {
    <ERROR! SNIA_StorageProtectionSetting has not been created.>
}

// Verify the protection setting using prior found instance
for #i in $StorageProtectionSettings->[] {
    if( $StorageProtectionSetting->[#i].ProtectionControlled == false ) {
        <ERROR! CIM_StorageVolume is not under protection controlled.>
    }

    if( $StorageProtectionSetting->[#i].Access == 2 &&
        $StorageProtectionSetting->[#i].ProtectExpirationSpecified == 2
        &&
        $StorageProtectionSetting->[#i].RemainingProtectionTime
            > #TwoYearsValue &&
        $StorageProtectionSetting->[#i].RemainingProtectionTime
            <= #RemainingProtectionTime + #TwoYearsValue )
    {
        <EXIT: StorageVolume Protection configuration successful>
    }
}

// if we get to this point, it was not set
<ERROR! SNIA_StorageProtectionSetting was not created>

// end of the recipe
```

26.7 Registered Name and Version

Storage Element Protection version 1.4.0 (Component Profile).

26.8 CIM Elements

26.8.1 General

Table 479 describes the CIM elements for Storage Element Protection.

Table 479 - CIM Elements for Storage Element Protection

Element Name	Requirement	Description
26.8.2 CIM_ElementCapabilities	Mandatory	Associates the capabilities to the service.
26.8.3 CIM_HostedService	Mandatory	Associates the service to the system providing the service.
26.8.4 SNIA_ElementProtectionSettingData	Mandatory	SNIA_ElementProtectionSettingData represents the association between the storage element to be protected and applicable protection setting.
26.8.5 SNIA_StorageProtectionCapabilities	Mandatory	
26.8.6 SNIA_StorageProtectionService	Mandatory	
26.8.7 SNIA_StorageProtectionSetting	Mandatory	SNIA_StorageProtectionSetting class holds properties for the protection-related configuration and statuses of a storage element. It is associated to the StorageVolume or LogicalDisk class by SNIA_ElementProtectionSettingData. A management application can retrieve the protection-related information by traversing the ElementProtectionSettingData association. If is not found, it indicates no protection management is applied for the storage element.

26.8.2 CIM_ElementCapabilities

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 480 describes class CIM_ElementCapabilities.

Table 480 - SMI Referenced Properties/Methods for CIM_ElementCapabilities

Properties	Flags	Requirement	Description and Notes
ManagedElement		Mandatory	The service.
Capabilities		Mandatory	The associated capabilities.

26.8.3 CIM_HostedService

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory.

Table 481 describes class CIM_HostedService.

Table 481 - SMI Referenced Properties/Methods for CIM_HostedService

Properties	Flags	Requirement	Description and Notes
Dependent		Mandatory	The protection service.
Antecedent		Mandatory	The system providing the service.

26.8.4 SNIA_ElementProtectionSettingData

Created By: Extrinsic: Protect

Modified By: Static

Deleted By: External

Requirement: Mandatory.

Table 482 describes class SNIA_ElementProtectionSettingData.

Table 482 - SMI Referenced Properties/Methods for SNIA_ElementProtectionSettingData

Properties	Flags	Requirement	Description and Notes
ManagedElement		Mandatory	The storage element to be protected.
SettingData		Mandatory	The protection setting and status of the storage element.

26.8.5 SNIA_StorageProtectionCapabilities

Created By: Static

Requirement: Mandatory.

Table 483 describes class SNIA_StorageProtectionCapabilities.

Table 483 - SMI Referenced Properties/Methods for SNIA_StorageProtectionCapabilities

Properties	Flags	Requirement	Description and Notes
ProtectionTimeGranularity		Mandatory	Granularity for the time period of StorageProtectionSetting.RemainingProtectionTime. 0: Unknown 1: Other 2: Second 3: Minute 4: Hour 5: Day.
SupportedStorageElementFeatures		Mandatory	Value for storage element protection. 1 (StorageVolume Protection), 2 (LogicalDisk protection).
SupportedSynchronousActions		Mandatory	Value for storage element protection. 1 (Storage Element Protection).
SupportedAsynchronousActions		Mandatory	Value for element protection. 1 (Storage Element Protection).

26.8.6 SNIA_StorageProtectionService

Created By: Static

Requirement: Mandatory.

Table 484 describes class SNIA_StorageProtectionService.

Table 484 - SMI Referenced Properties/Methods for SNIA_StorageProtectionService

Properties	Flags	Requirement	Description and Notes
Protect()		Mandatory	Configures the protection attributes of the storage element and prevents modification for a specific period of time. Values specified for this method will be set as properties of StorageProtectionSetting instance which is associated to the specified storage element. This method can be used to extend the retention period, but not for decreasing it.

26.8.7 SNIA_StorageProtectionSetting

Created By: Extrinsic: Protect

Modified By: Extrinsic: Protect

Deleted By: DeleteInstance

Requirement: Mandatory.

Table 485 describes class SNIA_StorageProtectionSetting.

Table 485 - SMI Referenced Properties/Methods for SNIA_StorageProtectionSetting

Properties	Flags	Requirement	Description and Notes
ProtectionControlled		Optional	Whether the storage element is under protection control or not.
Access		Mandatory	Read and write accessibility of the StorageVolume. 0: Unknown 1: Read/Write Enabled 2: Read Only 3: Write Once 4: Read/Write Disabled.
InquiryProtection		Conditional	Conditional requirement: Storage Volumes used as storage elements. StorageVolume protection method for SCSI inquiry commands. 0: Unknown 1: No SCSI Inquiry Protection 2: Inquiry Disabled 3: Zero Capacity Returned.
DenyAsCopyTarget		Optional	Whether the storage element can be specified as a copy target or not.
LUNMappingConfigurable		Conditional	Conditional requirement: Storage Volumes used as storage elements. Whether LU assignment to the StorageVolume is configurable or not.
ProtectionExpirationSpecified		Mandatory	Duration type of the storage element protection. 1: None 2: Limited Expiration 3: Permanent.
RemainingProtectionTime		Mandatory	Amount of remaining time before a management application can change the access permission.

EXPERIMENTAL

EXPERIMENTAL

27 Replication Services Profile

27.1 Description

27.1.1 Synopsis

Profile Name: Replication Services

Version: 1.5.0

Organization: SNIA

CIM schema version: 2.23

Central Class: ReplicationService

Scoping Class: ComputerSystem

27.1.2 Supported Profiles, Subprofiles, and Packages

Table 486 describes the supported profiles for Replication Services.

Table 486 - Supported Profiles for Replication Services

Profile Name	Organization	Version	Requirement	Description
Block Services	SNIA	1.5.0	Mandatory	
Copy Services	SNIA	1.5.0	Mandatory	
Job Control	SNIA	1.5.0	Optional	
Indication	SNIA	1.5.0	Optional	
Cascading	SNIA	1.3.0	Mandatory	Deprecated. This is a deprecated profile. Related cascading elements are marked as Optional.

27.1.3 Overview

The Replication Services, a *component* profile, specifies attributes and methods to copy data from a source element to a target element. The copy operations may be performed on elements from the same storage system or across a connection to a different storage system. Elements may be placed into a group in order to facilitate copy operations on many elements at the same time. The elements of a group may be declared as *Consistent*.

Two types of synchronization views are supported. A replica may be synchronized to the current view of the source element or may be synchronized to a point-in-time view. Snapshots and clones always represent a point-in-time view, while a mirror represents a current view.

Two copy operation modes are supported – synchronous and asynchronous. In the synchronous mode, the write operations to the source elements are reflected to the target elements before signalling the host that a write operation is complete. In the asynchronous mode, the host is signaled as soon as the write operations to the source elements are complete. However, the writes to the target elements may take place at a later time.

Replication Services supports local and remote replication. Local replication specifies that both the source and target elements are contained in a single managed system, such as an array platform. Remote replication specifies the source and the target elements are contained in separate systems. For remote replication, the client may interact with both the source and the target systems. However, the client only invokes the replication methods to a single Replication Service.

Replication Services supports “copying” thinly provisioned elements. Unlike fully provisioned elements, a thinly provisioned element has fewer actual allocated storage blocks than the advertised capacity of the element.

The Replication Service supports copy operations to and from *undiscovered resources*. An undiscovered resource is an addressable entity without a known object model.

Replication Services includes the methods to create the necessary access point and shared secret instances that may be required for copy operations to remote resources.

The Replication Service generally relies on the underlying implementation to perform the actual copy operations. However, the profile can expose the “copy methodology” if that information is available.

Throughout this profile, there are specific references to class properties and methods pertaining to each section. Refer to 27.8 “CIM Elements” for a complete list of all properties and methods, including their description.

27.1.4 Key Features

The following is a brief list of key features of the Replication Services

- The ability to specify individual or *Groups* of elements to manage replication
- The ability to copy to and from undiscovered resources
- The ability to support *Consistency Management*
- The ability to handle local and remote replication seamlessly
- The ability to replicate *Thinly Provisioned* elements
- The ability to offer different *Copy Methodologies*
- The ability to efficiently retrieve replication relationships
- The ability to reduce the potential to receive many unwanted indications

27.1.5 Replication Services and Copy Services Profiles

The Replication Services Profile extends the functionality of the Copy Services Subprofile by including enhanced local replication for thinly provisioned storage objects, remote replication, and support for replication groups and consistency groups.

Any action taken via a Copy Services conformant interface shall be reflected correctly in the applicable Replication Services properties. Furthermore, any action taken via a Replication Services conformant interface shall be reflected correctly in the applicable Copy Services properties, as if the similar action was taken by the Copy Services. Refer to 27.5.2 “Replication Services and Copy Services Properties and Methods Mapping” for mapping between Copy Services specific properties and properties introduced for Replication Services.

27.1.6 Key Components

Table 487 shows a list of key classes used by Replication Services. Refer to 27.5 “Methods of the Profile” and “CIM Elements” for additional details on methods and properties of these classes.