Storage Management Technical Specification, Part 3 Block Devices

Version 1.6.0, Revision 4

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

SNIA Technical Position

10 February, 2012

Revision History

Revision 1

Date

16 July 2010

SCRs Incorporated and other changes

Block Storage Views Profile (DRM-SMIS-SCR00209)
- Changed the version of profile to 1.6.0 and the CIM-VERSION to 2.27
- Replaced the SNIA_BaseInstance with CIM_ElementView
- Promoted all SNIA_ classes to CIM_ classes
- Added a SNIA_DiskDriveView, which is subclassed from CIM_DiskDriveView and adds 3 properties
- Added a SNIA_VolumeView which is subclassed from CIM_VolumeView and adds the CanDelete property (from SNIA_StorageVolume)
- Did a lot of editing of diagrams and text for the promotion to CIM_ classes

Masking and Mapping Subprofile (DRM-SMIS-SCR00212)
- Added AssociatedPrivilege association and deprecated AuthorizedSubject, AuthorizedTarget, AuthorizedPrivilege, and PrivilegeManagementService

Replication Services Profile (DRM-SMIS-SCR00210)
- Added properties CopyRecoveryMode, FailedCopyStopsHostIO, UnequalListsAction, and DeltaUpdateBlocks to class ReplicationSettingData
- Added Urgent to the value map of CopyPriority of classes ReplicationSettingData and StorageSynchronized
- Expanded value map of ReplicationSettingData.TargetElementSupplier to include "Client must supply" a target element
- Added properties CopyRecoveryMode and FailedCopyStopsHostIO to class Synchronized
- Expanded the value map of Synchronized.CopyState and added Partitioned and Invalid states
- Added property ConsistentPointInTime to class ReplicationGroup
- Added methods CreateGroupReplicaFromElements, GetReplicationRelationshipInstances, and ModifyListSettingsDefineState to class ReplicationService
- Introduced class RemoteReplicationCollection, inherited for class ConnectivityColllection, and includes properties Active and DeleteOnUnassociated

Thin Provisioning Profile (DRM-SMIS-SCR00213)
- Put in a note that this profile is being superceded by the Block Services with Thin Provisioning Profile

Storage Relocation profile (DRM-SMIS-SCR00208)
- A **new** profile that adds services for relocating data in storage

Block Services with Thin Provisioning Profile (DRM-SMIS-SCR00213)
- A **new** profile that extends Block Services for thin provisioning

Comments

Editorial notes and DRAFT material are displayed.

Revision 2

Date

7 October 2010

SCRs Incorporated and other changes

Array (SMIS-160-Draft-SCR00003)
- Updated the profile to include the ImplementationCapabilities of the Array implementation

Block Server Performance (SMIS-150-Errata-SCR00017)
-Added the "N" (NULL OK) qualifier to the CSVSequence properties of the Predefined and Client Defined BlockStatisticsManifest class tables

Replication Services (DRM-SMIS-SCR00218)
- Introduced the class RemoteReplicationCollection
- Converted all the SNIA classes used by the Replication Services Profile to CIM classes.

Storage Virtualizer (SMIS-160-Draft-SCR00003)
- Updated the profile to include the ImplementationCapabilities of the Array implementation

Storage Relocation profile (DRM-SMIS-SCR-00215)
- Several updates were made throughout the profile

Automated Storage Tiering Profile (SMIS-160-Draft-SCR00005)
- Added **new** profile that adds Storage Tiers to block devices

Comments

Editorial notes and DRAFT material are displayed.

Revision 3

Date

10 March 2011

SCRs Incorporated and other changes

Array (SMIS-160-Draft-SCR00003)
- Promoted the profile to include the ImplementationCapabilities of the Array implementation

Block Services Package (DRM-SMIS-SCR00215, DRM-SMIS-SCR00219 & DRM-SMIS-SCR00223)
- Updated the version to be 1.6.0
- Added OperationalStatus to StoragePool and made it conditional on Storage Relocation
- Added OperationalStatus tables for StorageVolumes, LogicalDisks and StoragePools to the Health and Fault Management section
- Moved CanDelete to CIM_StorageVolume and deleted the SNIA_StorageVolume
- Added CapacityInMigratingSource and CapacityInMigratingTarget to StoragePool
- Updated OperationalStatus for StorageVolume and LogicalDisk to include the 19 (Relocating) as a secondary value.
- Added NumExtentsMigrating to StorageVoume and LogicalDisk
- Added the Storage Relocation enumerations to StorageConfigurationCapabilities properties and made them conditional on Storage Relocation (rather than optional)
- Added the method CreateElementsFromStoragePool for creating multiple elements
- Added the method ReturnElementsToStoragePool for deleting multiple elements

Block Storage Views Profile (DRM-SMIS-SCR00209)
- Changed the CIM-VERSION of profile to 2.29 (from 2.27)
- Moved the SVCanDelete property from SNIA_VolumeView to CIM_VolumeView and deleted SNIA_VolumeView
- Added an SVIsComposite property to CIM_VolumeView
- Rewrote the section the Class Diagrams for Block Storage View Classes
- Changed the name of the CIM_ExposedView class to CIM_MaskingMappingExposedDeviceView
- Changed the SupportedViews enumeration for CIM_MaskingMappingExposedDeviceView to be "SNIA:MaskingMappingExposedDeviceView" instead of "SNIA:ExposedView"
- In CIM_ProtocolControllerForUnitView, changed PCFUDeviceNumber to DeviceNumber
- Adjusted the subclassing of associations (not subclassed from CIM_View)
- Changed the references in CIM_ProtocolControllerForUnitView back to Antecedent and Dependent
- Changed the references in CIM_HostedStoragePoolView back to GroupComponent and PartComponent
- Renamed the BasedOnView properties to StartingAddress, EndingAddress, OrderIndex from BOStartingAddress, BOEndingAddress, BOOrderIndex
- Edited Figures for other changes

Storage Virtualizer (SMIS-160-Draft-SCR00003)
- Promoted the profile to include the ImplementationCapabilities of the Array implementation

Replication Services (DRM-SMIS-SCR00222)
- Added explicit support for creating "Read Only" elements
- Added the property StorageSynchronized.ReadOnly to indicate when an element is "Read Only"

Thin Provisioning (DRM-SMIS-SCR00213)
- Promoted "Block Services with Thin Provisioning" to be "Thin Provisioning" as an Experimental Profile
- Removed the old "Thin Provisioning" Profile
- Added the Thin Provisioning predefined FilterCollection and IndicationFilters with their related associations

Storage Relocation (DRM-SMIS-SCR00215)
- Promoted the profile to Experimental

Automated Storage Tiering (DRM-SMIS-SCR00221)
- Updated the profile based on reviews and feedback from vendors offering storage tiering
- Adjusted the profile based on related classes in CIM 2.28
- Expanded the notion of a storage tier to include additional storage objects such as storage pools and volumes
- Included a method to locate candidate storage objects that can comprise a storage tier
- Promoted the profile to Experimental

SMI-S Information Model (TSG-SMIS-SCR00294)
- Added this Annex

Comments

Editorial notes are displayed.
DRAFT material was hidden.

Revision 4

Date

28 September 2011

SCRs Incorporated and other changes

Array (TSG-SMIS-SCR-00307)
- Updated the predefined filter collection name to be "SNIA:Array:Predefined"
- Added an optional FilterCollection: "SNIA:Array:ProfileSpecificLifecycleIndicationFilterCollection" and its associations
- Changed the references to reference the new Indications profile (specialization of the DMTF Indications Profile)

Block Services Package (TSG-SMIS-SCR-00307)
- Updated the predefined filter collection name to be "SNIA:Block Services:Predefined"
- Added an optional FilterCollection: "SNIA:Block Services:ProfileSpecificLifecycleIndicationFilterCollection" and its associations
- Changed the references to reference the new Indications profile (specialization of the DMTF Indications Profile)
- Updated subclause "5.1.18 Block Services Predefined Indications" to reflect the above changes

Disk Drive Lite (TSG-SMIS-SCR00307)
- Updated the predefined filter collection name to be "SNIA:Disk Drive Lite:Predefined"
- Added an optional FilterCollection: "SNIA:Disk Drive Lite:ProfileSpecificLifecycleIndicationFilterCollection" and its associations
- Changed the references to reference the new Indications profile (specialization of the DMTF Indications Profile)

Extent Composition (TSG-SMIS-SCR00307)
- Updated the predefined filter collection name to be "SNIA:Extent Composition:Predefined"

Storage Virtualizer (TSG-SMIS-SCR00307)
- Updated the predefined filter collection name to be "SNIA:Storage Virtualizer:Predefined"
- Added an optional FilterCollection: "SNIA:Storage Virtualizer:ProfileSpecificLifecycleIndicationFilterCollection" and its associations
- Changed the references to reference the new Indications profile (specialization of the DMTF Indications Profile)

Storage Relocation (TSG-SMIS-SCR00307)
- Added an optional FilterCollection "SNIA:Storage Relocation:ProfileSpecificAlertIndicationFilterCollection" and its associations
- Changed the references to reference the new Indications profile (specialization of the DMTF Indications Profile)

Thin Provisioning (TSG-SMIS-SCR00307)
- Updated the predefined filter collection name to be "SNIA:Thin Provisioning:Predefined"
- Added an optional FilterCollection: "SNIA:Thin Provisioning:ProfileSpecificAlertIndicationFilterCollection" and its associations
- Changed the references to reference the new Indications profile (specialization of the DMTF Indications Profile)

Automated Storage Tiering (DRM-SMIS-SCR00225)
- Adjusted the associations SystemComponent and Component
- Addressed the comments from 1.6 rev 3 ballot
- Updated diagrams to conform to the new associations and classes

Comments

Editorial notes and DRAFT material are hidden.

INTENDED AUDIENCE

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

DISCLAIMER

CHANGES TO THE SPECIFICATION

Each publication of this specification is uniquely identified by a three-level identifier, comprised of a version number, a release number and an update number. The current identifier for this specification is version 1.2.0. Future publications of this specification are subject to specific constraints on the scope of change that is permissible from one publication to the next and the degree of interoperability and backward compatibility that should be assumed between products designed to different publications of this standard. The SNIA has defined three levels of change to a specification:

- Major Revision: A major revision of the specification represents a substantial change to the underlying scope or architecture of the SMI-S API. A major revision results in an increase in the version number of the version identifier (e.g., from version 1.x.x to version 2.x.x). There is no assurance of interoperability or backward compatibility between releases with different version numbers.

- Minor Revision: A minor revision of the specification represents a technical change to existing content or an adjustment to the scope of the SMI-S API. A minor revision results in an increase in the release number of the specification’s identifier (e.g., from x.1.x to x.2.x). Minor revisions with the same version number preserve interoperability and backward compatibility.

- Update: An update to the specification is limited to minor corrections or clarifications of existing specification content. An update will result in an increase in the third component of the release identifier (e.g., from x.x.1 to x.x.2). Updates with the same version and minor release levels preserve interoperability and backward compatibility.

TYPOGRAPHICAL CONVENTIONS

Maturity Level

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

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

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

Experimental Maturity Level

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

EXPERIMENTAL

Experimental content appears here.

EXPERIMENTAL

Figure 1 - Experimental Maturity Level Tag

Implemented Maturity Level

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

IMPLEMENTED

Implemented content appears here.

IMPLEMENTED

Figure 2 - Implemented Maturity Level Tag

Stable Maturity Level

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

STABLE

Stable content appears here.

STABLE

Figure 3 - Stable Maturity Level Tag

Finalized Maturity Level

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

Deprecated Material

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

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

DEPRECATED

Content that has been deprecated appears here.

DEPRECATED

Figure 4 - Deprecated Tag

Contents

List of Tables

List of Figures

Foreword

The Block Devices part of the *Storage Management Technical Specification* 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 standard is subdivided in the following parts:

- *Storage Management Technical Specification, Overview, 1.6.0 Rev 3*
- *Storage Management Technical Specification, Part 1 Common Architecture, 1.6.0 Rev 4*
- *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4*
- *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4*
- *Storage Management Technical Specification, Part 4 Filesystems, 1.6.0 Rev 4*
- *Storage Management Technical Specification, Part 5 Fabric, 1.6.0 Rev 4*
- *Storage Management Technical Specification, Part 6 Host Elements, 1.6.0 Rev 4*
- *Storage Management Technical Specification, Part 7 Media Libraries, 1.6.0 Rev 4*

Acknowledgments

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

Organization Represented	*Name of Representative*
Brocade Communications Systems	John Crandall
EMC Corporation	Tony Fiorentino
	Mike Hadavi
Hitachi Data Systems	Steve Quinn
IBM	Jun Wei Zhang
Individual Contributor	Mike Walker
Individual Contributor	Paul von Behren
WBEM Solutions, Inc	Jim Davis

SNIA Web Site

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

SNIA Address

Requests for interpretation, suggestions for improvement and addenda, or defect reports are welcome. They should be sent via the SNIA Feedback Portal at http://www.snia.org/feedback/ or by mail to the Storage Networking Industry Association, 425 Market Street, Suite 1020, San Francisco, CA 94105, U.S.A.

Clause 1: Scope

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

Clause 2: Normative References

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

2.1 Approved references

ISO/IEC 14776-452, SCSI Primary Commands - 2 (SPC-2) [ANSI INCITS.351-2001]

2.2 References under development

Storage Management Technical Specification, Part 1 Common Architecture, 1.6.0 Rev 4

Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4

ISO/IEC 14776-452, SCSI Primary Commands - 3 (SPC-3) [ANSI INCITS.351-2005]

DMTF WBEM URI Mapping Specification (DSP0207) 1.0.01 (preliminary)

2.3 Other references

DMTF DSP0214:2004 CIM Operations over HTTP

Clause 3: Definitions, Symbols, Abbreviations, and Conventions

For the purposes of this document, the definitions, symbols, abbreviations, and conventions given in *Storage Management Technical Specification, Part 1 Common Architecture, 1.6.0 Rev 4* apply.

STABLE

Clause 4: Array Profile

4.1 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 8: Host Hardware RAID Controller Profile in *Storage Management Technical Specification, Part 6 Host Elements, 1.6.0 Rev 4*.

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 StorageVolmes 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

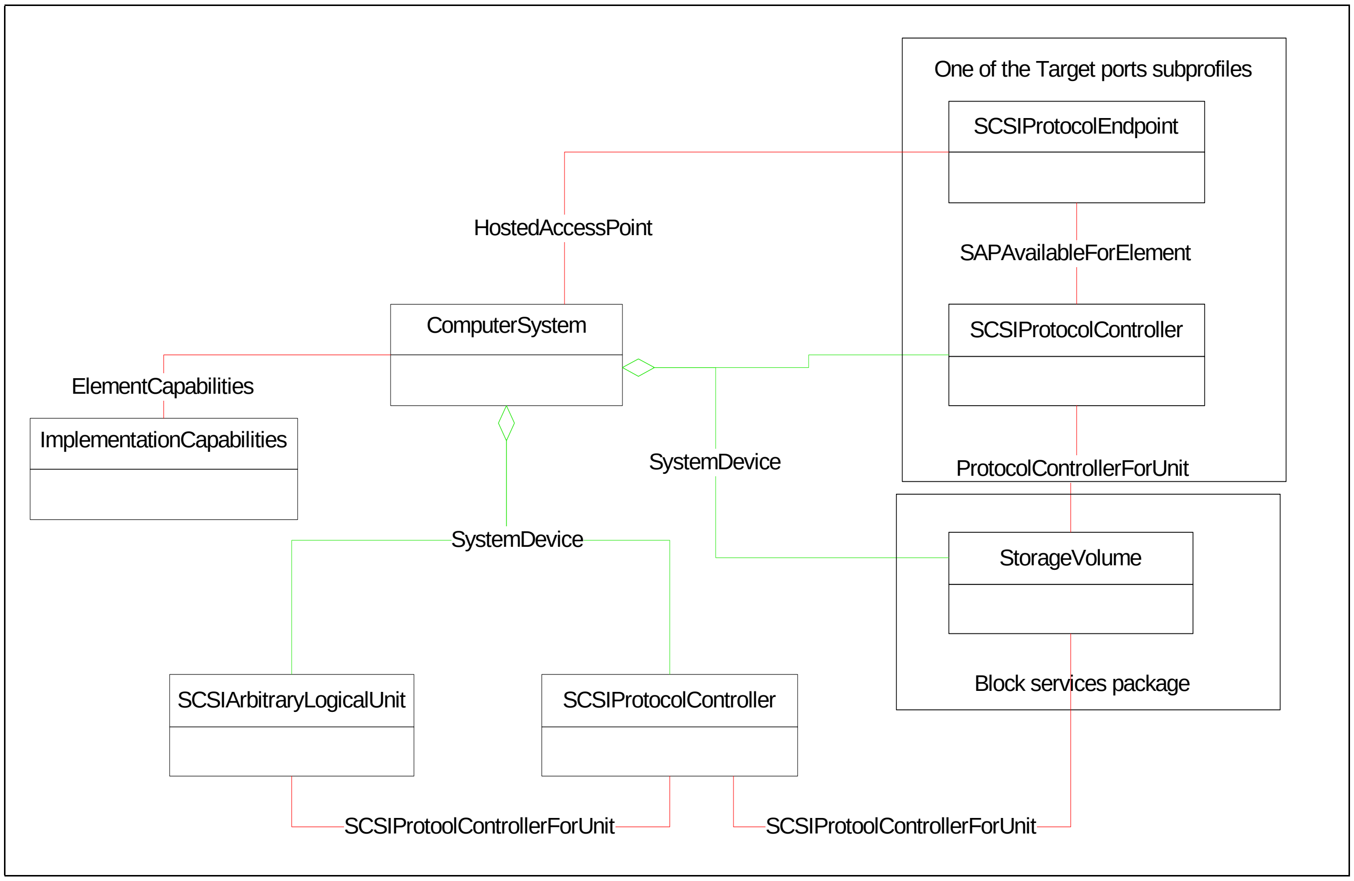

Figure 5 - Array Profile Instance Diagram

At the minimum, the Array Profile provides a high level read-only ‘view’ of an array. Clause 5: Block Services Package includes the basic description of how storage is managed.

EXPERIMENTAL

The capabilities of the Array implementation are identified in an instance of CIM_ImplementationCapabilities, which is associated to the top level Array ComputerSystem via ElementCapabilities. This includes information on the capacity optimization techniques supported by the Array.

EXPERIMENTAL

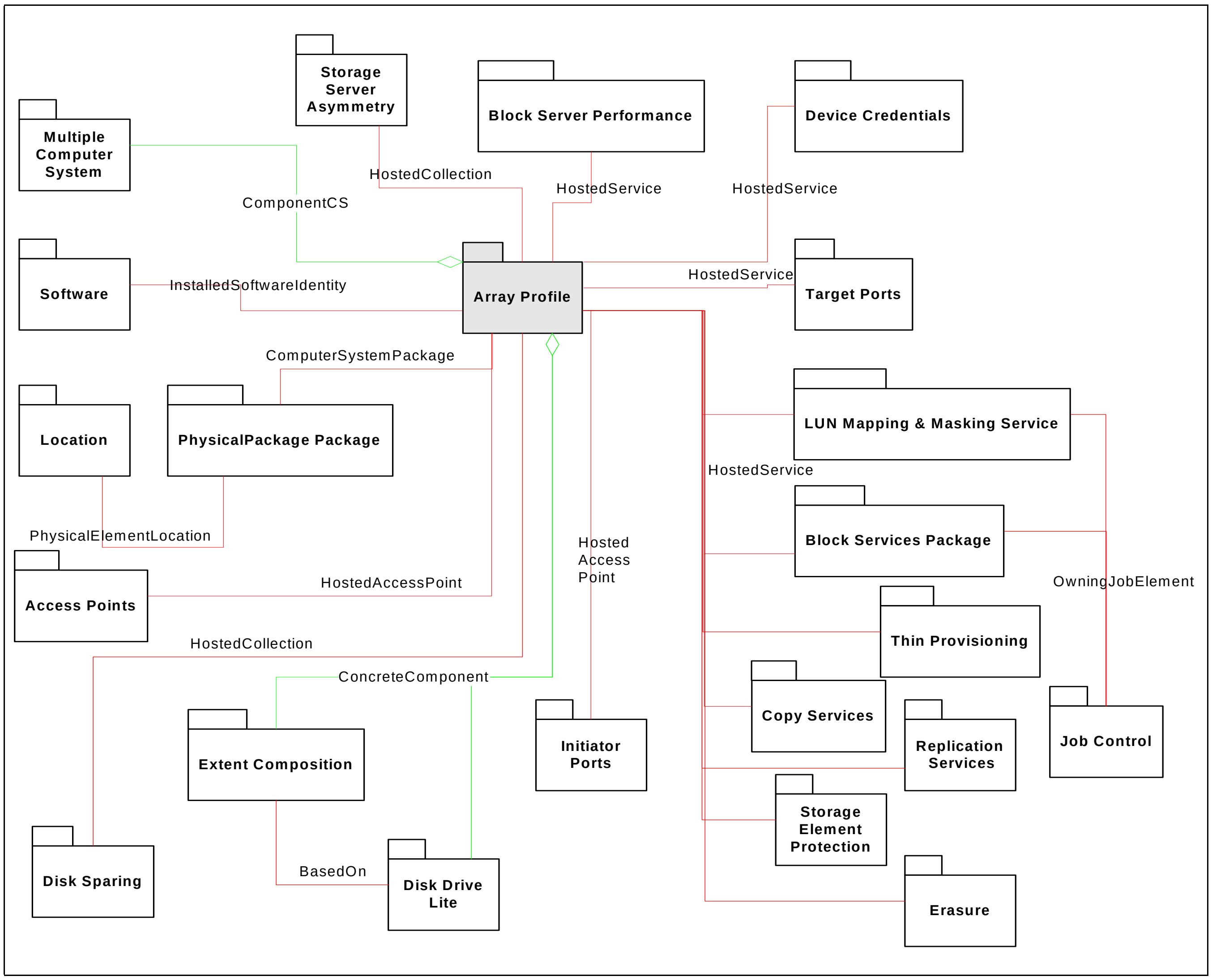

Figure 6 - Array Package Diagram

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 Clause 5: 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 Clause 14: 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 14: Generic Initiator Ports Profile) and the Disk Drive Lite Subprofile (see Clause 11: 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.

Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4, Clause 30: 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.

Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4 Clause 31: Physical Package Package describes the physical layout of the array and includes product identification information.

4.2 Health and Fault Management

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

4.3 Cascading Considerations

Not defined in this standard.

4.4 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. See Multiple Computer System
Extra Capacity Set	SNIA	1.0.2	Optional	Deprecated. See Multiple Computer System
Disk Drive	SNIA	1.0.2	Optional	Deprecated. See Disk Drive Lite
Disk Drive Lite	SNIA	1.6.0	Optional	

Table 1 - Supported Profiles for Array

Profile Name	Organization	Version	Requirement	Description
Extent Mapping	SNIA	1.0.2	Optional	Deprecated. See Extent Composition
Extent Composition	SNIA	1.6.0	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	1.02	Optional	Deprecated. See Block Services
LUN Creation	SNIA	1.0.2	Optional	Deprecated. See Block Services
Device Credentials	SNIA	1.3.0	Optional	
LUN Mapping and Masking	SNIA	1.0.2	Optional	Deprecated. See Masking and Mapping
Masking and Mapping	SNIA	1.6.0	Optional	
Disk Sparing	SNIA	1.5.0	Optional	
Block Services	SNIA	1.6.0	Mandatory	
CKD Block Services	SNIA	1.3.0	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	1.6.0	Optional	Experimental.
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	1.4.0	Optional	Experimental.
Replication Services	SNIA	1.6.0	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

Table 1 - Supported Profiles for Array

Profile Name	Organization	Version	Requirement	Description
FC Target Ports	SNIA	1.4.0	Support for at least one is mandatory.	
iSCSI Target Ports	SNIA	1.6.0		
SAS Target Ports	SNIA	1.4.0		Experimental.
SB Target Ports	SNIA	1.2.0		
FC Initiator Ports	SNIA	1.6.0		
SAS Initiator Ports	SNIA	1.4.0		Experimental.
ATA Initiator Ports	SNIA	1.4.0		Experimental.
Backend Ports	SNIA	1.0.2		Deprecated. See specific Initiator Ports
Indication	SNIA	1.5.0	Support for at least one is mandatory.	Deprecated. See the SNIA Indications Profile
Indications	SNIA	1.6.0		Experimental.
Indications	DMTF	1.2.0		Experimental. See DSP1054, version 1.2.0

4.5 Methods of the Profile

None.

4.6 Use Cases

EXPERIMENTAL

4.6.1 Discover the Capacity Optimization Support in an Array

4.6.1.1 Summary

From a list of available Array devices, determine which devices support any capacity optimization techniques.

4.6.1.2 Basic Course of Events

1) Administrator identifies an available array device.

2) Administrator determines if the array advertises implementation capabilities.

3) System responds with an implementation capabilities.

4) Administrator inspects the capacity optimization techniques supported by the array

4.6.1.3 Alternative Paths

none

4.6.1.4 Exception Paths

FAILED:

- The Array System does not report implementation capabilities
- The Array System reports implementation capabilities, but reports “none” for supported capacity optimizations.

4.6.1.5 Triggers

Device selection for provisioning storage for an application.

4.6.1.6 Assumptions

The administrator has a list of candidate array system names for doing provisioning.

4.6.1.7 Preconditions

The systems are available.

EXPERIMENTAL

4.7 Registered Name and Version

Array version 1.6.0 (Autonomous Profile)

4.8 CIM Elements

Table 2 describes the CIM elements for Array.

Table 2 - CIM Elements for Array

Element Name	Requirement	Description
4.8.1 CIM_ComputerSystem (Top Level System)	Mandatory	'Top level' system that represents the whole array. Associated to RegisteredProfile.
4.8.2 CIM_ElementCapabilities (ImplementationCapabilities to System)	Optional	Experimental. Associates the conformant Array ComputerSystem to the CIM_ImplementationCapabilities supported by the implementation.
4.8.3 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.
4.8.4 CIM_FilterCollection (Array ProfileSpecificLifecycleIndicationFilterCollection)	Optional	Experimental. This is a collection of Lifecycle IndicationFilters defined by the Array Profile.

Table 2 - CIM Elements for Array

Element Name	Requirement	Description
4.8.5 CIM_HostedCollection (Array to ProfileSpecificLifecycleIndicationFilterCollection)	Optional	Experimental.
4.8.6 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).
4.8.7 CIM_ImplementationCapabilities (ImplementationCapabilities)	Optional	Experimental. The capabilities of the profile implementation.
4.8.8 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.
4.8.9 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.
4.8.10 CIM_MemberOfCollection (Array ProfileSpecificLifecycleIndicationFilterCollection to Array Filters)	Optional	Experimental. This associates the Array ProfileSpecificLifecycleIndicationFilterCollection to the Filters defined by the Array Profile.
4.8.11 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.
4.8.12 CIM_ProtocolControllerForUnit (Arbitrary LU for All LUNs View)	Conditional	Conditional requirement: Elements that are mandatory if Masking and Mapping is not implemented.
4.8.13 CIM_ProtocolControllerForUnit (Storage volumes for 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
4.8.14 CIM_SCSIArbitraryLogicalUnit (Arbitrary LU)	Optional	A SCSI Logical Unit that exists only for management of the array.
4.8.15 CIM_SCSIProtocolController (All LUNs View)	Conditional	Conditional requirement: Elements that are mandatory if Masking and Mapping is not implemented.
4.8.16 CIM_SystemDevice (System to SCSIArbitraryLogicalUnit)	Conditional	Conditional requirement: Elements that are mandatory if SCSIArbitraryLogicalUnit is instantiated. This association links SCSIArbitraryLogicalUnit to the scoping system.
4.8.17 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 section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 4.8.8 CIM_IndicationFilter (Array System Creation).
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_ComputerSystem	Mandatory	Deletion of an array instance. See section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 4.8.9 CIM_IndicationFilter (Array System Deletion).

4.8.1 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 '1.6.0'.

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 & Notes
CreationClassName		Mandatory	
Name		Mandatory	Unique identifier for the array. Eg IP address.

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

Properties	Flags	Requirement	Description & Notes
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.

4.8.2 CIM_ElementCapabilities (ImplementationCapabilities to System)

Experimental. Associates the conformant Array ComputerSystem to the CIM_ImplementationCapabilities supported by the implementation.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 4 describes class CIM_ElementCapabilities (ImplementationCapabilities to System).

Table 4 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (ImplementationCapabilities to System)

Properties	Flags	Requirement	Description & Notes
Capabilities		Mandatory	The ImplementationCapabilities.
ManagedElement		Mandatory	The conformant Array ComputerSystem that has ImplementationCapabilities.

4.8.3 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 5 describes class CIM_FilterCollection (Array Predefined FilterCollection).

Table 5 - SMI Referenced Properties/Methods for CIM_FilterCollection (Array Predefined Filter-Collection)

Properties	Flags	Requirement	Description & 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:Predefined'.

4.8.4 CIM_FilterCollection (Array ProfileSpecificLifecycleIndicationFilterCollection)

Experimental. This is a collection of Lifecycle IndicationFilters defined by the Array Profile.

Requirement: Optional

Table 6 describes class CIM_FilterCollection (Array ProfileSpecificLifecycleIndicationFilterCollection).

Table 6 - SMI Referenced Properties/Methods for CIM_FilterCollection (Array ProfileSpecificLife-cycleIndicationFilterCollection)

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	See the InstanceID definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.13 CIM_FilterCollection (StaticFilterCollection).
CollectionName		Mandatory	The value of CollectionName shall be 'SNIA:Array:ProfileSpecificLifecycleIndicationFilterCollection'.

4.8.5 CIM_HostedCollection (Array to ProfileSpecificLifecycleIndicationFilterCollection)

Experimental.

Requirement: Optional

Table 7 describes class CIM_HostedCollection (Array to ProfileSpecificLifecycleIndicationFilterCollection).

Table 7 - SMI Referenced Properties/Methods for CIM_HostedCollection (Array to ProfileSpecifi-cLifecycleIndicationFilterCollection)

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	Reference to the ProfileSpecificLifecycleIndicationFilterCollection for the Array.
Antecedent		Mandatory	Reference to the 'Top level' Array System.

4.8.6 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 8 describes class CIM_HostedCollection (Array to predefined FilterCollection).

Table 8 - SMI Referenced Properties/Methods for CIM_HostedCollection (Array to predefined FilterCollection)

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

4.8.7 CIM_ImplementationCapabilities (ImplementationCapabilities)

Experimental. The capabilities (features) of the profile implementation.

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

Table 9 describes class CIM_ImplementationCapabilities (ImplementationCapabilities).

Table 9 - SMI Referenced Properties/Methods for CIM_ImplementationCapabilities (ImplementationCapabilities)

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	An opaque, unique id for the implementation capability of an implementation.
ElementName		Optional	A provider supplied user-friendly name for this CIM_ImplementationCapabilities element.
SupportedCapacityOptimizations		Mandatory	This array of strings lists the capacity optimization techniques that are supported by the implementation. Valid string values are "none" | "SNIA:Thin Provisioning" | "SNIA:Data Compression" | "SNIA:Data Deduplication".
SupportedViews		Mandatory	This array of strings lists the view classes that are supported by the implementation. Valid string values are "none" | "SNIA:VolumeView" | "SNIA:DiskDriveView" | "SNIA:ExposedView" | "SNIA:MaskingMappingView" | "SNIA:MappingProtocolControllerView" | "SNIA:StoragePoolView" | "SNIA:ReplicaPairView" .

4.8.8 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 10 describes class CIM_IndicationFilter (Array System Creation).

Table 10 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Array System Creation)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Name		Mandatory	This shall be 'SNIA:Array:SystemCreation'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
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 *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).

4.8.9 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 11 describes class CIM_IndicationFilter (Array System Deletion).

Table 11 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Array System Deletion)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Name		Mandatory	This shall be 'SNIA:Array:SystemDeletion'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
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 *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).

4.8.10 CIM_MemberOfCollection (Array ProfileSpecificLifecycleIndicationFilterCollection to Array Filters)

Experimental. This associates the Array ProfileSpecificLifecycleIndicationFilterCollection to the Filters defined by the Array Profile.

Requirement: Optional

Table 12 describes class CIM_MemberOfCollection (Array ProfileSpecificLifecycleIndicationFilterCollection to Array Filters).

Table 12 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Array Profile-SpecificLifecycleIndicationFilterCollection to Array Filters)

Properties	Flags	Requirement	Description & Notes
Collection		Mandatory	Reference to the Array ProfileSpecificLifecycleIndicationFilterCollection.
Member		Mandatory	Reference to the IndicationFilters defined by the Array Profile.

4.8.11 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 13 describes class CIM_MemberOfCollection (Predefined Filter Collection to Array Filters).

Table 13 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Predefined Filter Collection to Array Filters)

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

4.8.12 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 14 describes class CIM_ProtocolControllerForUnit (Arbitrary LU for All LUNs View).

Table 14 - SMI Referenced Properties/Methods for CIM_ProtocolControllerForUnit (Arbitrary LU for All LUNs View)

Properties	Flags	Requirement	Description & Notes
DeviceNumber		Mandatory	Address (e.g. LUN) of the associated Device. Shall be formatted as unseparated uppercase hexadecimal digits, with no leading 0x.
DeviceAccess		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.

4.8.13 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 15 describes class CIM_ProtocolControllerForUnit (Storage volumes for All LUNs View).

Table 15 - SMI Referenced Properties/Methods for CIM_ProtocolControllerForUnit (Storage volumes for All LUNs View)

Properties	Flags	Requirement	Description & Notes
DeviceNumber		Mandatory	Address (e.g. LUN) of the associated Device. Shall be formatted as unseparated uppercase hexadecimal digits, with no leading 0x.
DeviceAccess		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).

4.8.14 CIM_SCSIArbitraryLogicalUnit (Arbitrary LU)

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

Table 16 describes class CIM_SCSIArbitraryLogicalUnit (Arbitrary LU).

Table 16 - SMI Referenced Properties/Methods for CIM_SCSIArbitraryLogicalUnit (Arbitrary LU)

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

4.8.15 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 17 describes class CIM_SCSIProtocolController (All LUNs View).

Table 17 - SMI Referenced Properties/Methods for CIM_SCSIProtocolController (All LUNs View)

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

4.8.16 CIM_SystemDevice (System to SCSIArbitraryLogicalUnit)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Elements that are mandatory if SCSIArbitraryLogicalUnit is instantiated.

Table 18 describes class CIM_SystemDevice (System to SCSIArbitraryLogicalUnit).

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

Properties	Flags	Requirement	Description & Notes
PartComponent		Mandatory	
GroupComponent		Mandatory	

4.8.17 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 19 describes class CIM_SystemDevice (System to SCSIProtocolController).

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

Properties	Flags	Requirement	Description & Notes
PartComponent		Mandatory	
GroupComponent		Mandatory	

STABLE

STABLE

Clause 5: Block Services Package

5.1 Description

5.1.1 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.

5.1.2 Storage Capacity States

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

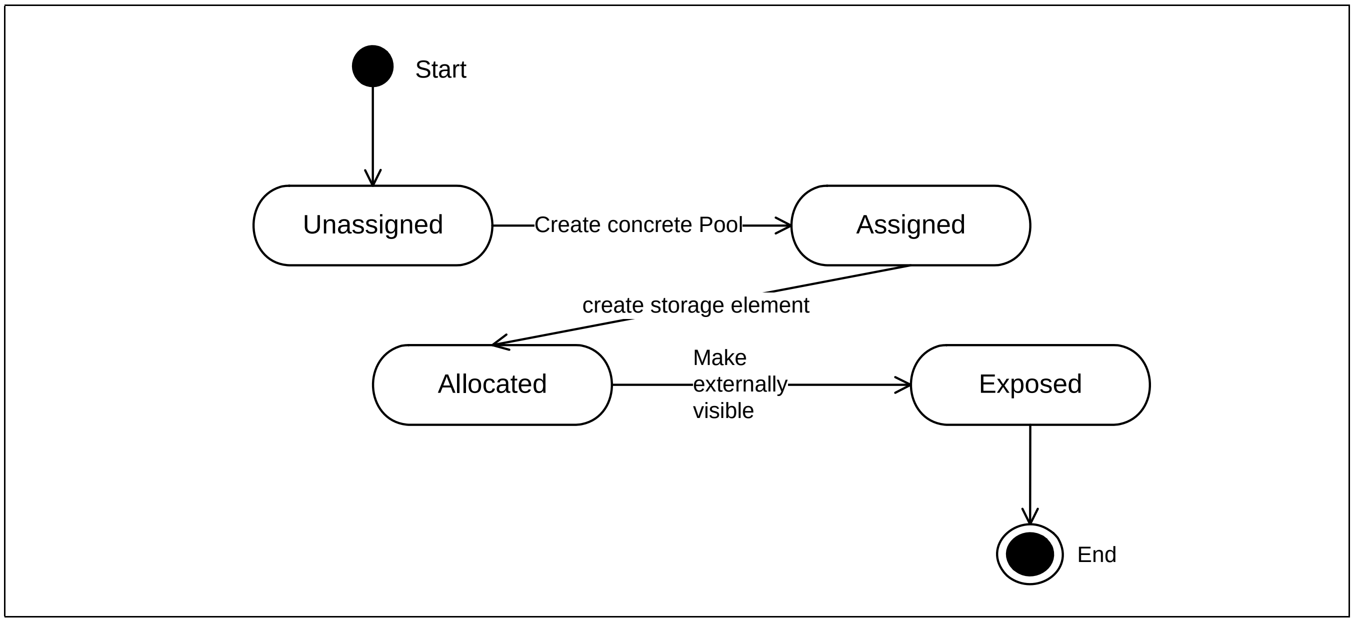

Figure 7 - Storage Capacity State

Each block of capacity within a storage device or application has a state. StorageVolumes and LogicalDisks, the storage elements described in this section, 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.

5.1.3 StoragePools

5.1.3.1 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".

5.1.3.2 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 exists 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.

5.1.3.3 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.

5.1.4 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.

5.1.5 StoragePool Management Instance Diagram

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

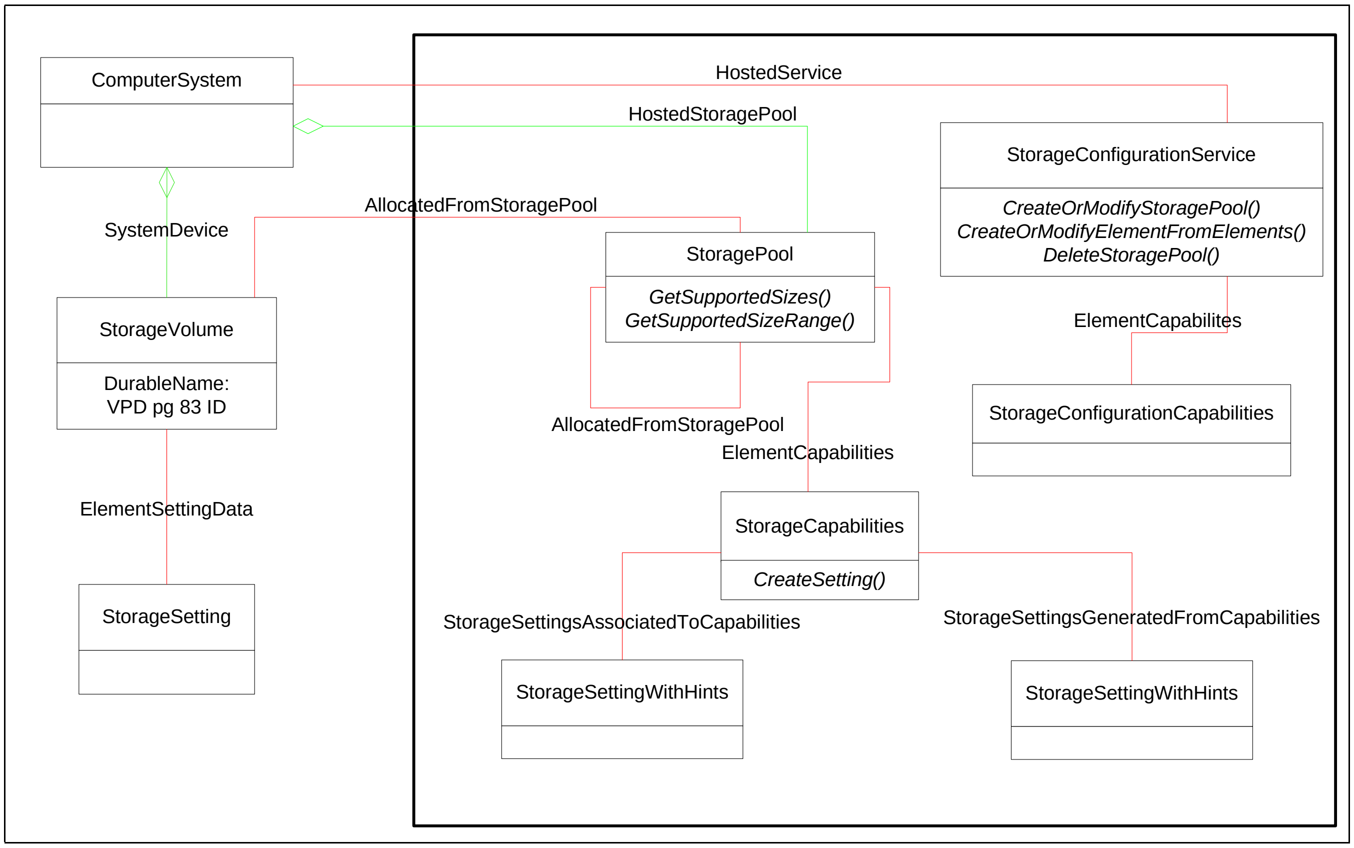

Figure 8 - StoragePool Manipulation Instance Diagram

5.1.6 StoragePool, StorageVolume and LogicalDisk Manipulation

5.1.6.1 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".

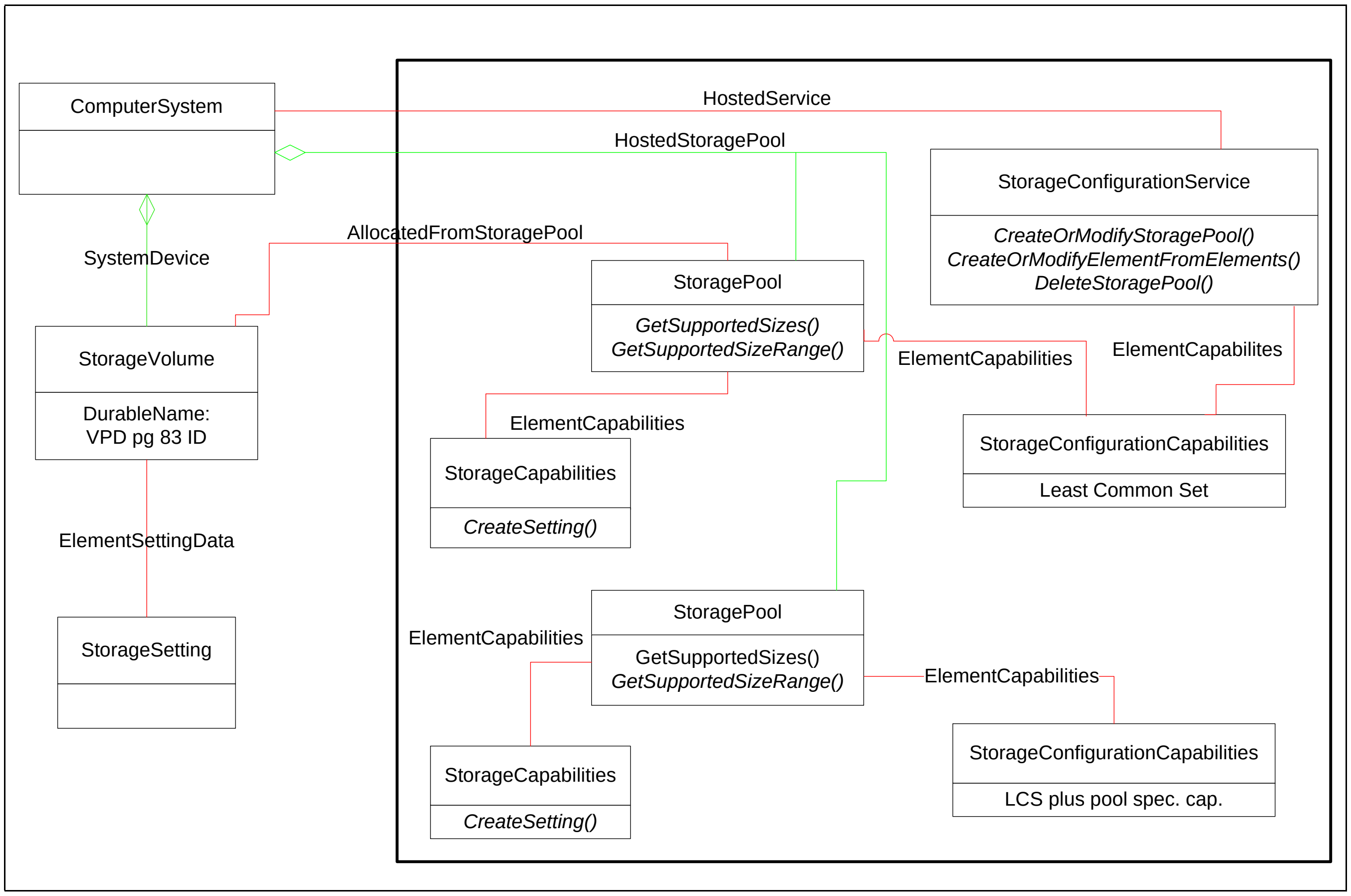

Figure 9 - Capabilities Specific to a StoragePool

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 StorgeVolume 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 must 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 that is no longer contributing capacity to a pool can be deleted. In other words, the expectation is that the value of CanDelete shall change dynamically.

EXPERIMENTAL

5.1.6.2 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.

5.1.6.3 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.

EXPERIMENTAL

- CreateElementsFromStoragePool: Create one or more StorageVolumes or LogicalDisks in a single method call, possibly with a specific StorageSetting, from a source StoragePool.

EXPERIMENTAL

- 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

- ReturnElementsToStoragePool: In a single method call, return one or more elements previously created with CreateOrModifyElementFromStoragePool or CreateElementsFromStoragePool to the originating StoragePool.
- To locate Pools, Volumes, or Logical Disks based on their current usage, use the method StorageConfigurationService.GetElementsBasedOnUsage.

EXPERIMENTAL

5.1.6.4 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 5.5.3 for details on the associations from Setting to Capabilities.

5.1.6.5 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 discontiguous 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.

5.1.7 Declaring Storage Configuration Options

If no StorageConfigurationService is present, then the implementation offers no standard configuration capability (see section 5.1.4 "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 20 defines how the SupportedSynchronousActions and SupportedAsynchonousActions array values map to methods in the StorageConfigurationService class. The presence of an ‘Action’ from Table 20 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 20 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 20 also gives the specific meanings of a supported actions in the context of the associated pool (“Pool x”).

EXPERIMENTAL

Table 20 - 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

Table 20 - Mapping: Supported Actions to Methods

Action	Associated to “Pool x” Meaning	SCS Method
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	
22 “Multiple Storage Element Creation”	“Pool x” may be used as the InPool parameter of CreateElementsFromStoragePool	CreateElementsFromStoragePool
23 “Multiple Storage Element Return”	No meaning specified.	ReturnElementsToStoragePool

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 21.

Table 21 - 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 elements 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 22.

Table 22 - 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 specification, 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 23.

Table 23 - SupportedStoragePoolFeatures Array

Supported Special Features	Explanation
3 "StorageVolume Creation", 5 "StorageVolume Modification"	The SMI-S implementation can create or modify StorageVolumes respectively.

Table 23 - SupportedStoragePoolFeatures Array

Supported Special Features	Explanation
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 specification, 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.

5.1.8 StorageVolume Creation Instance Diagram

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

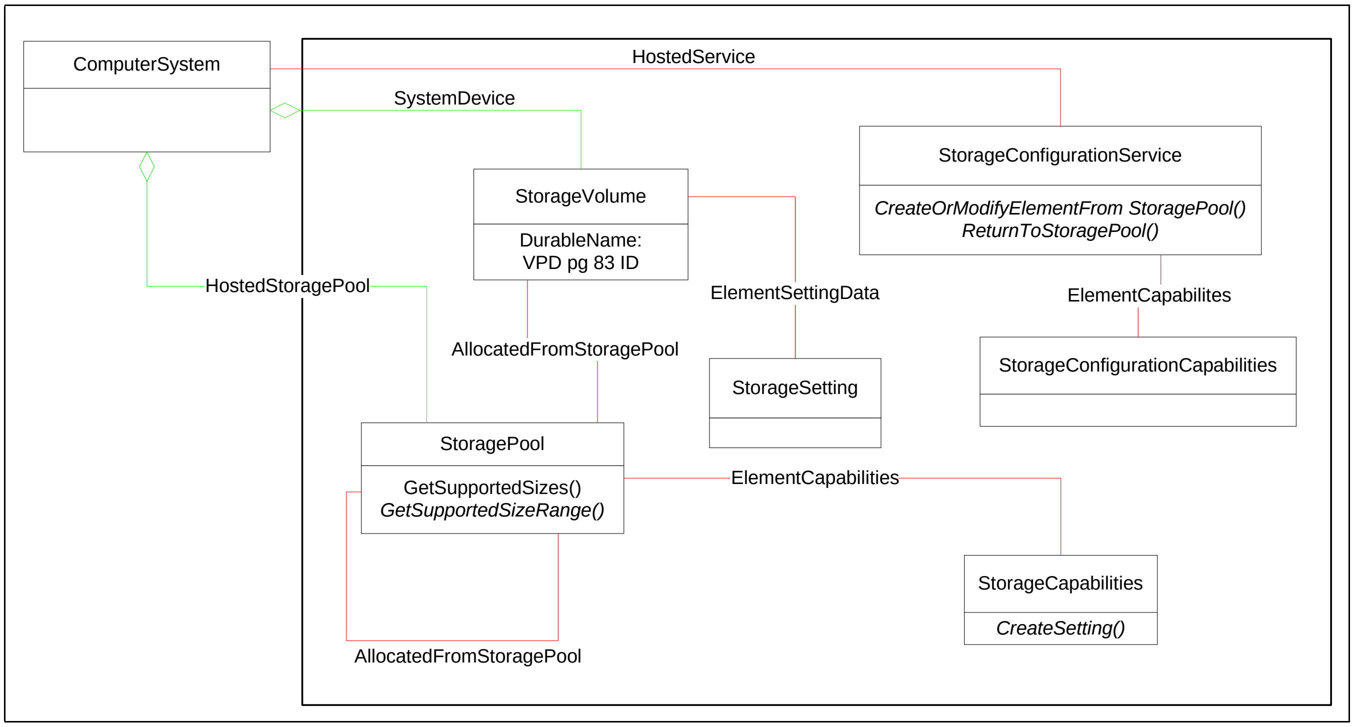

Figure 10 - StorageVolume Creation Instance Diagram

5.1.9 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 *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* Clause 40:, "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

5.1.10 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".

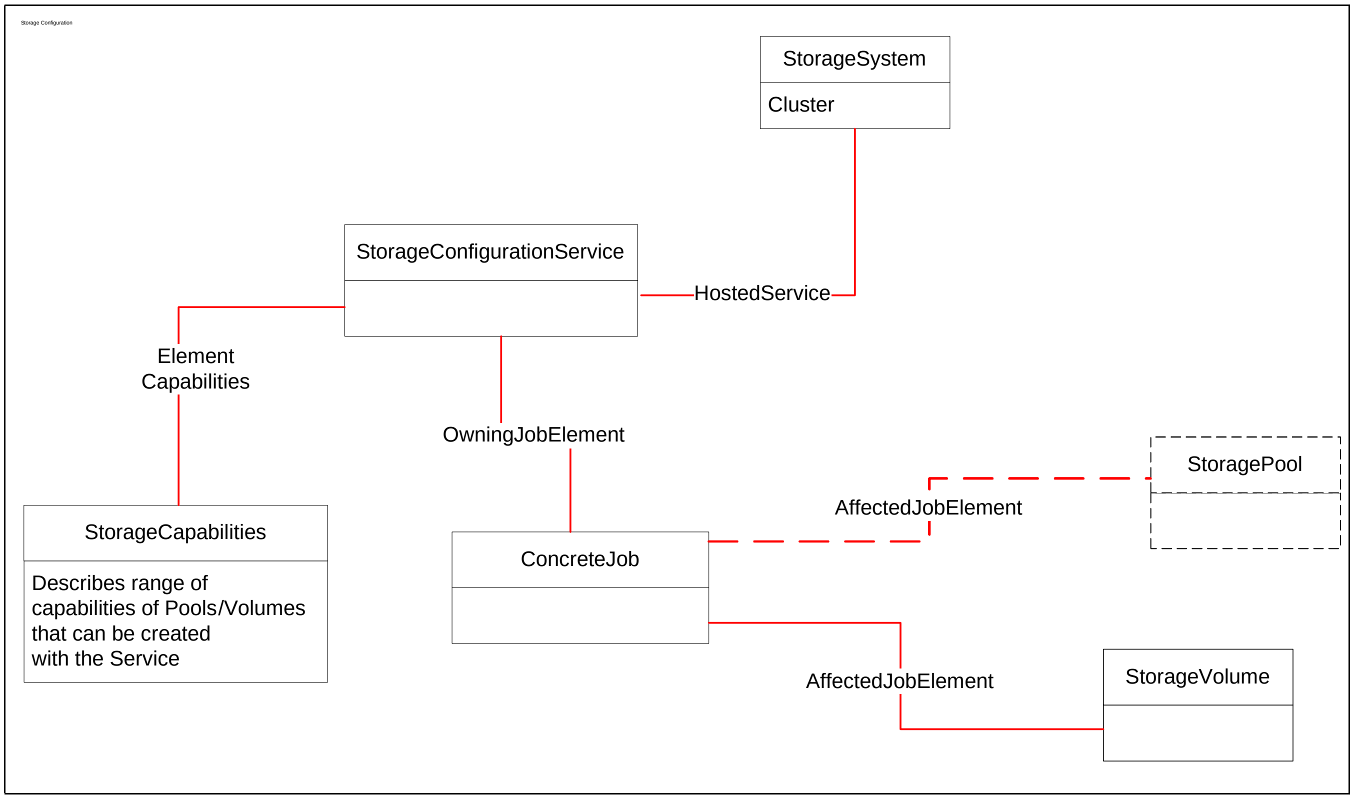

Figure 11 - Storage Configuration

See *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* Clause 26: 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 24.

5.1.11 Mapping of RAID levels to Data Redundancy and Package Redundancy

Table 24 reflects available definitions of RAID levels.

Table 24 - 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 "ChangableType" 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".

5.1.12 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 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

5.1.13 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 is 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 25 describes some of the representative values of the Usage property (storage element refers to a StorageVolume, LogicalDisk, or StoragePool):

Table 25 - 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 StorageVolume or LogicalDisk is now acting as a StorageExtent. In this case, the storage element no longer appears in the list of these element types. Use the method GetElementsBasedOnUsage to locate such storage elements.

EXPERIMENTAL

5.1.14 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 Global StorageConfigurationCapabilities. 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 be a *read-only* implementation. In such a model, only classes listed in Table 26 are required. Classes not explicitly listed are not required for *read-only* implementations.

Table 26 - 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

5.1.15 StorageExtent Conservation

5.1.15.1 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.

5.1.15.2 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 14: 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 14: Extent Composition Subprofile will tolerate these additional extents. The modeling of free/unused extents is defined only in 5.1.15 StorageExtent Conservation.

Support of the GetAvailableExtents and CreateOrModifyElementFromElements methods are not required by the Block Services package nor Clause 14: Extent Composition Subprofile. An implementation may support the representation of StorageVolume or LogicalDisk structure through Clause 14: 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 14: Extent Composition Subprofile. See 5.5.3. 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

5.1.15.3 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 14: 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.

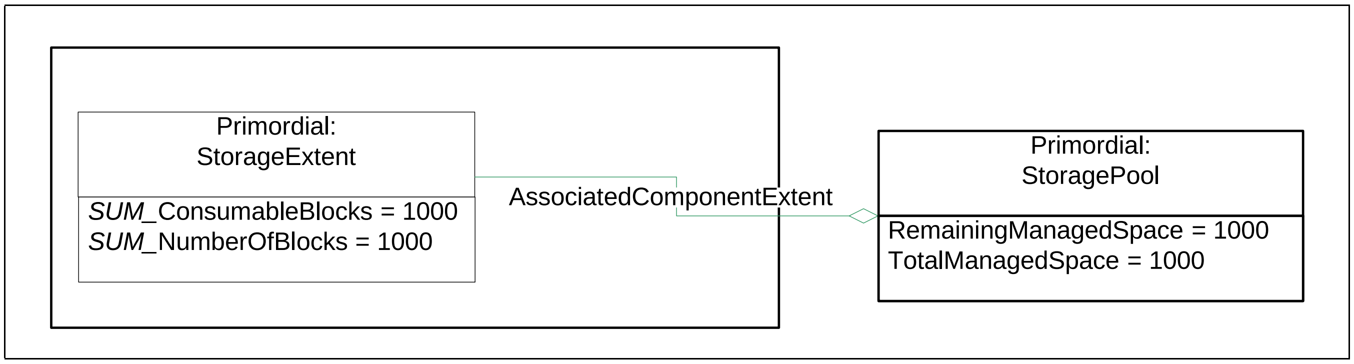
Primordial:
StorageExtent
*SUM*_ConsumableBlocks = 1000
*SUM*_NumberOfBlocks = 1000
AssociatedComponentExtent
Primordial:
StoragePool
RemainingManagedSpace = 1000
TotalManagedSpace = 1000

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 14: 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.

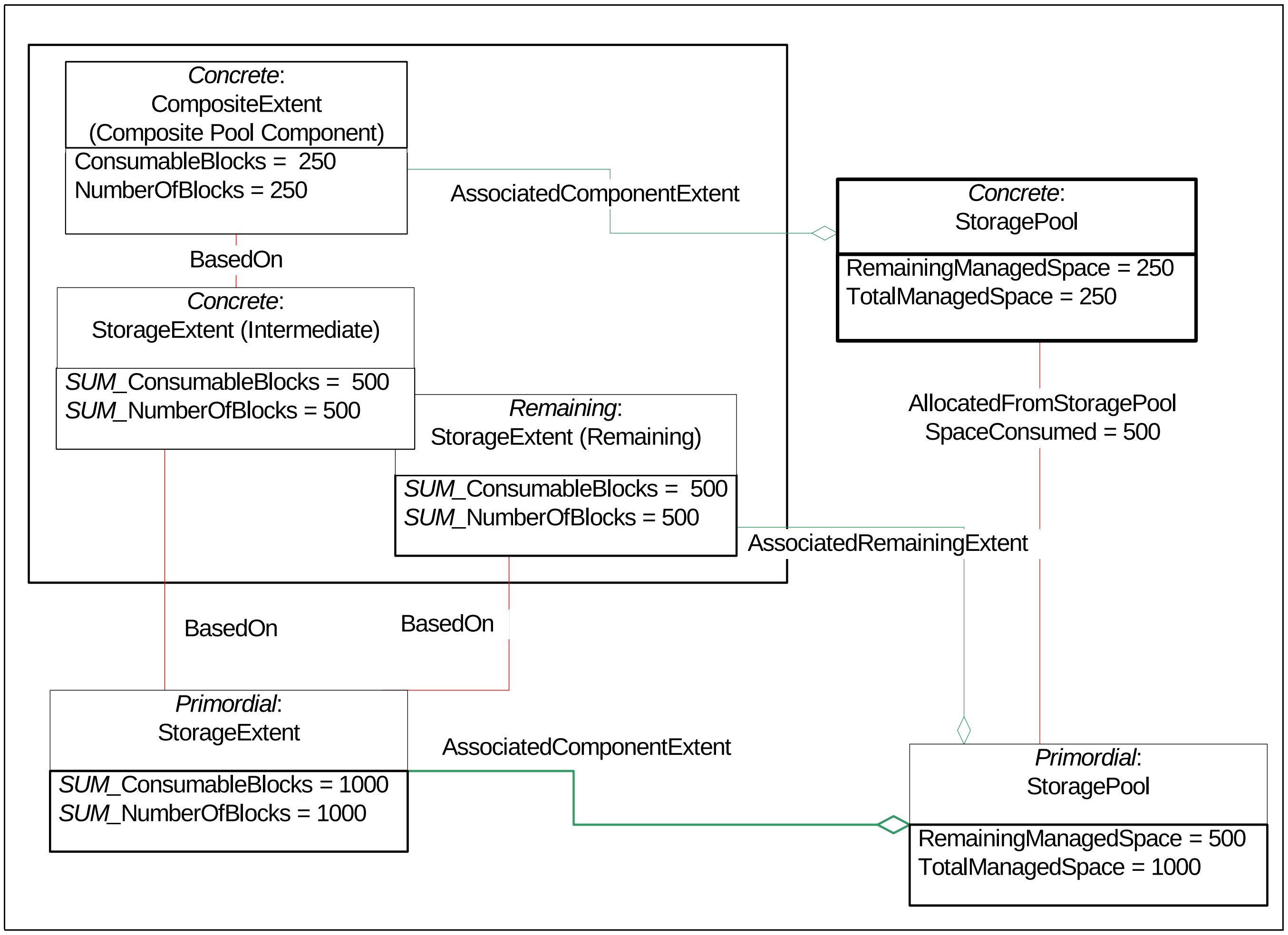
Concrete:
CompositeExtent
(Composite Pool Component)
ConsumableBlocks = 250
NumberOfBlocks = 250
AssociatedComponentExtent
Concrete:
StoragePool
RemainingManagedSpace = 250
TotalManagedSpace = 250
BasedOn
Concrete:
StorageExtent (Intermediate)
*SUM*_ConsumableBlocks = 500
*SUM*_NumberOfBlocks = 500
Remaining:
StorageExtent (Remaining)
*SUM*_ConsumableBlocks = 500
*SUM*_NumberOfBlocks = 500
AllocatedFromStoragePool
SpaceConsumed = 500
AssociatedRemainingExtent
BasedOn
BasedOn
Primordial:
StorageExtent
*SUM*_ConsumableBlocks = 1000
*SUM*_NumberOfBlocks = 1000
AssociatedComponentExtent
Primordial:
StoragePool
RemainingManagedSpace = 500
TotalManagedSpace = 1000

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 14: 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.

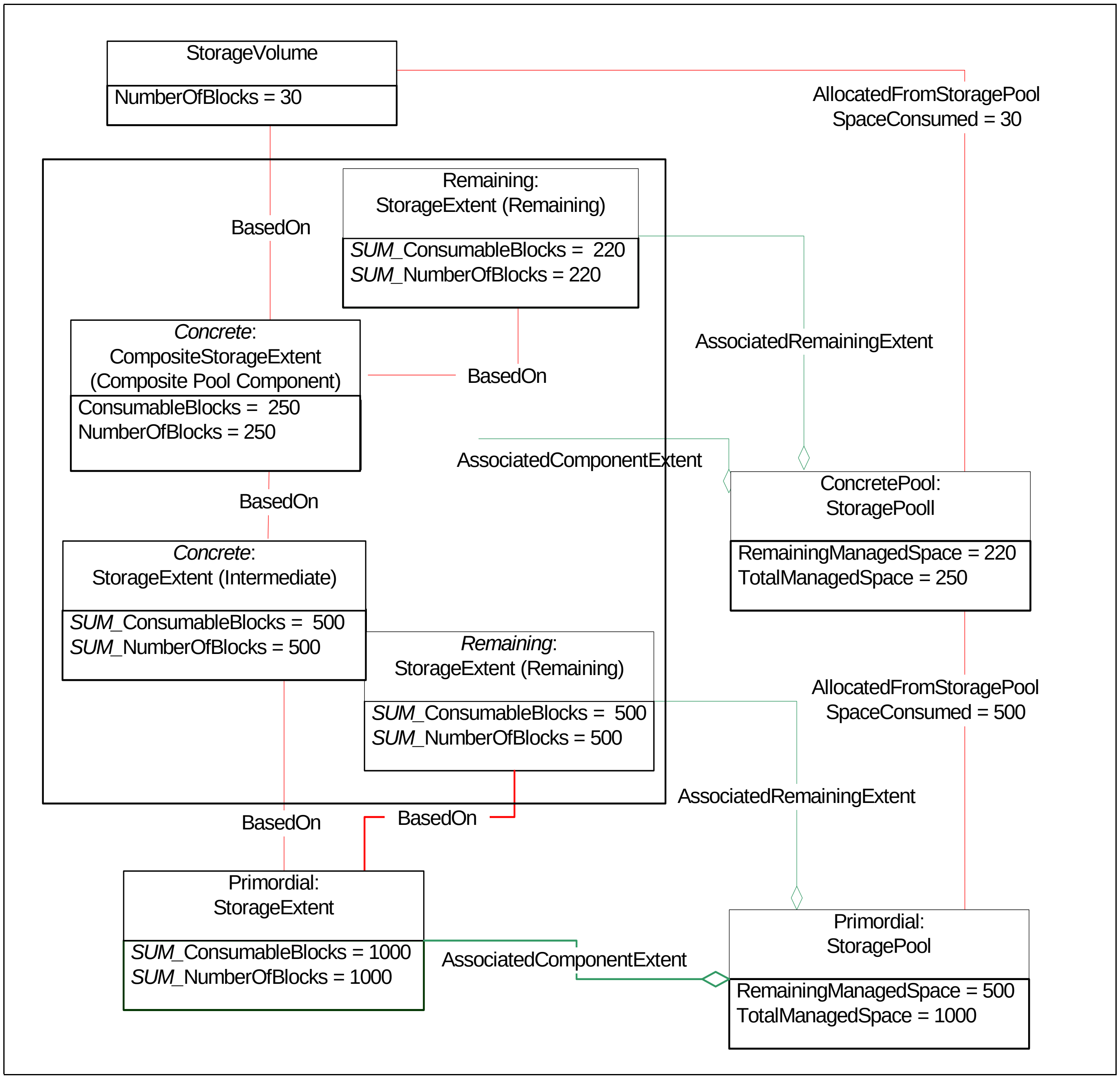

Figure 14 - StorageExtent Conservation - Step 3

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.

5.1.16 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 14: Extent Composition Subprofile is implemented:

IMPLEMENTED

1) The StoragePool's TotalManagedSpaceshall be equal to the sum of all the AssociatedComponentExtent 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:

   a) 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.

   b) The set of StorageExtents associated to the StoragePool via AssociatedRemainingExtent.

IMPLEMENTED

5.1.17 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, CreateElementsFromStoragePool, 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( ), CreateElementsFromStoragePool() methods. A StorageVolume creation operation takes time, and a Client needs to be aware that the operation is not complete until the StorageVoume.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 32, “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()

EXPERIMENTAL

- CreateElementsFromStoragePool

EXPERIMENTAL

- 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.RequstUsageChange method.

EXPERIMENTAL

EXPERIMENTAL

5.1.18 Block Services Predefined Indications

If the Indications 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.

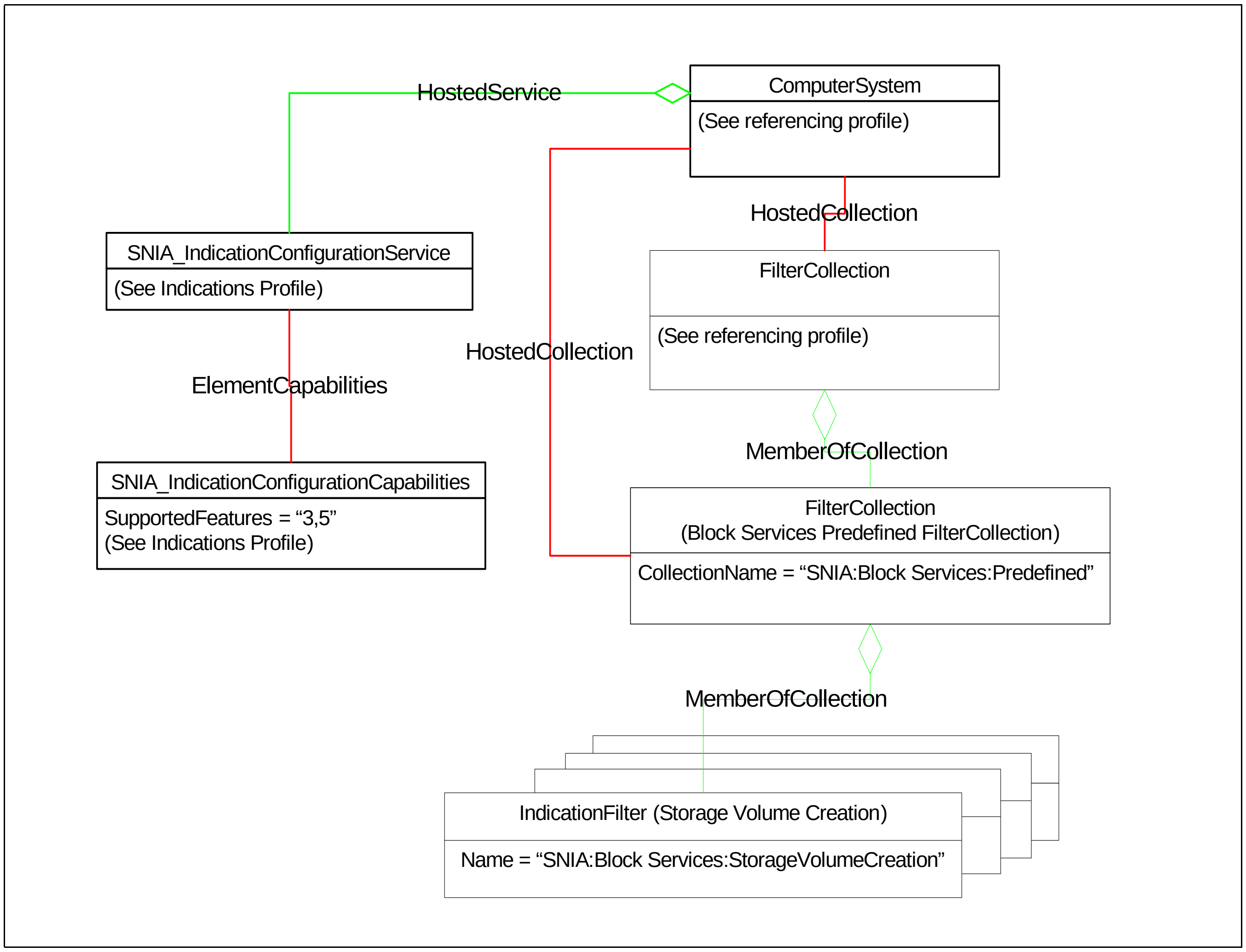
HostedService
ComputerSystem
(See referencing profile)
HostedCollection
SNIA_IndicationConfigurationService
(See Indications Profile)
FilterCollection
(See referencing profile)
HostedCollection
ElementCapabilities
MemberOfCollection
SNIA_IndicationConfigurationCapabilities
SupportedFeatures = “3,5”
(See Indications Profile)
FilterCollection
(Block Services Predefined FilterCollection)
CollectionName = “SNIA:Block Services:Predefined”
MemberOfCollection
IndicationFilter (Storage Volume Creation)
Name = “SNIA:Block Services:StorageVolumeCreation”

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:Predefined”. 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 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

5.2 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 *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* Clause 25: Health Package for details.

EXPERIMENTAL

The standard messages specific to this profile are listed in Table 27. See *Storage Management Technical Specification, Part 1 Common Architecture, 1.6.0 Rev 4* Clause 8: Standard Messages for a description of standard messages and the list of all standard messages.

Table 27 - 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

EXPERIMENTAL

5.2.1 StoragePool OperationalStatus

The StoragePool.OperationalStatus is optional, but required if the Storage Relocation profile is implemented. The StoragePool.OperationalStatus contains the overall status of the storage pool, summarized in Table 28.

Table 28 - OperationalStatus for StoragePool

Primary OperationalStatus	Subsidiary OperationalStatus	Description
2 “OK”		The storage pool is operational
2 “OK”	19 "Relocating”	The storage pool is operational, but is undergoing relocation
3 "Degraded"		The storage pool is operational, but at a lower quality of service than requested
3 "Degraded"	19 "Relocating”	The storage pool is operational, but at a lower quality of service due to a relocation operation
6 "Error"		The storage pool is in error
15 "Dormant"		The storage pool is not operational
15 "Dormant"	19 "Relocating”	The storage pool is not operational due to a relocation operation

EXPERIMENTAL

5.2.2 StorageVolume OperationalStatus

The StorageVolume.OperationalStatus contains the overall status of the volume, summarized in Table 29.

Table 29 - OperationalStatus for StorageVolume

Primary OperationalStatus	Subsidiary OperationalStatus	Description
2 “OK”		The storage volume is operational
2 “OK”	19 "Relocating	The storage volume is operational, but is undergoing relocation
3 "Degraded"		The storage volume is operational, but at a lower quality of service than requested
3 "Degraded"	19 "Relocating	The storage volume is operational, but at a lower quality of service due to a relocation operation
6 "Error"		The storage volume is in error
8 "Starting"		The storage volume is starting
15 "Dormant"		The storage volume is not operational

Table 29 - OperationalStatus for StorageVolume

Primary OperationalStatus	Subsidiary OperationalStatus	Description
15 "Dormant"	19 "Relocating	The storage volume is not operational due to a relocation operation

5.2.3 LogicalDisk OperationalStatus

The LogicalDisk.OperationalStatus contains the overall status of the logical disk, summarized in Table 30.

Table 30 - OperationalStatus for LogicalDisk

Primary OperationalStatus	Subsidiary OperationalStatus	Description
2 “OK”		The logical disk is operational
2 “OK”	19 "Relocating	The logical disk is operational, but is undergoing relocation
3 "Degraded"		The logical disk is operational, but at a lower quality of service than requested
3 "Degraded"	19 "Relocating	The logical disk is operational, but at a lower quality of service due to a relocation operation
6 "Error"		The logical disk is in error
8 "Starting"		The logical disk is starting
15 "Dormant"		The logical disk is not operational
15 "Dormant"	19 "Relocating	The logical disk is not operational due to a relocation operation

5.3 Cascading Considerations

Not defined in this standard.

5.4 Supported Profile, Subprofiles and Packages

Table 31 describes the supported profiles for Block Services.

Table 31 - Supported Profiles for Block Services

Profile Name	Organization	Version	Requirement	Description
Job Control	SNIA	1.5.0	Optional	
Extent Composition	SNIA	1.6.0	Optional	

5.5 Methods of this Profile

5.5.1 Extrinsic Methods on StorageCapabilities

5.5.1.1 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 (5.5.1.2, 5.5.1.3, and 5.5.1.4) 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 specification 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.

5.5.1.2 Getting Stripe Length

```
uint32 GetSupportedStripeLengths(
                            [Out] unint16 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.

5.5.1.3 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.

5.5.1.4 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.

5.5.2 Intrinsic Methods on StorageSetting

The following Intrinsic write methods are supported on StorageSetting:

- DeleteInstance
- ModifyInstance

5.5.3 Extrinsic Methods on StorageConfiguration

5.5.3.1 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 14: 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.

5.5.3.2 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

5.5.3.3 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 DSP00200 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 5.1.15.
- TheElement: If the method completes without creating a Job, then the TheElement is the storage element that is created. Otherwise, TheElement may or may not be Null. When the TheElement is NULL, then the storage element created can be determined by using the Job model.

5.5.3.4 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.

5.5.3.5 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 Dependent.

5.5.3.6 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 *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* Clause 26: 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.

EXPERIMENTAL

5.5.3.7 CreateElementsFromStoragePool

```
uint32 CreateElementsFromStoragePool (
       [In]  string ElementNames[]
       [In]  uint16 ElementType;
       [In]  uint64 ElementCount,
       [Out] CIM_ConcreteJob ref Job,
       [In]  CIM_StorageSetting ref Goal,
       [In, Out] Uint64 Size,
       [In]  CIM_StoragePool ref InPool,
       [Out] CIM_LogicalElement ref TheElements[] );
```

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

- ElementNames: One or more user relevant names. The first entry is assigned to the first created element, the second entry is assigned to the second element, and so on. If the number of entries in the ElementNames array is not equal to the ElementCount, the method shall return an error. If ElementNames is null, system assigns the element names.
- ElementType: This enumeration specifies what type of object to create. For example, “2” to create StorageVolumes.
- ElementCount: Count of elements 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 *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* Clause 26: 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.
- InPool: This contains the reference to the source StoragePool. If not specified, system locates an appropriate StoragePool.
- TheElements: If the method completes without creating a Job, then the TheElements are the storage elements that are created successfully. Otherwise, TheElements may be NULL. When the TheElements are NULL, the storage elements that are created can be determined by using the Job model.

Notes: If a job was created, there will be AffectedJobElement associations between the Job and each created elements. The number of the AffectedJobElement associations is the number of the elements created successfully.

If the method completes without creating a Job, references to the created elements are returned in the TheElements parameter. If the number of elements created is less than the number of elements requested, the return value of the method shall be 4098.

Generally, there will be one instance creation indication for each element created. However, some implementations may treat the entire request as one transaction and only generate one instance creation indication for all the elements created. See section 5.8.35

EXPERIMENTAL

5.5.3.8 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 *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* Clause 26: 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 is the desired size of the element. If it is not possible to create a StorageVolume of the desired size, a return code of “Size not supported” is 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.

5.5.3.9 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

5.5.3.10 ReturnElementsToStoragePool

```
uint32 ReturnElementsToStoragePool (
   [In]  uint16 Options,
   [Out] CIM_ConcreteJob ref Job,
   [In]  CIM_LogicalElement ref TheElements[] );
```

This method allows a client to delete a previously created elements such as StorageVolumes. The parameters are:

- Options: This enumeration specifies what should happen if non-existent element is supplied. A value of “2” requests the method to continue to delete the remaining elements that exist. A value of “3” requests the method to return an error. If null, the method deletes the elements that do exist (same as “2”).
- 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 *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* Clause 26: Job Control Subprofile.
- TheElements: An array containing the elements to be deleted.

EXPERIMENTAL

EXPERIMENTAL

5.5.3.11 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 specification 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 this specification. 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 *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* Clause 26: 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

5.5.3.12 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 then 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

5.5.3.13 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 then returning this code.

EXPERIMENTAL

5.5.4 Extrinsic Methods on StoragePool

5.5.4.1 General

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

- GetSupportedSizes
- GetSupportedSizeRange
- GetAvailableExtents

5.5.4.2 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 (aka 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.

5.5.4.3 GetSupportedSizeRange

```
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 of 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 1024-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 discontiguous 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 (aka an "empty" storage pool) or StorageVolumes, the returned MinimumVolumeSize parameter will have the value of 0.

5.5.4.3.1 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.

5.5.4.3.2 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.

5.5.4.4 Return Values

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.

5.5.4.4.1 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 specification 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 then 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 14: 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 12: 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.

5.6 Client Considerations and Recipes

5.6.1 Representative Instance Diagram

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

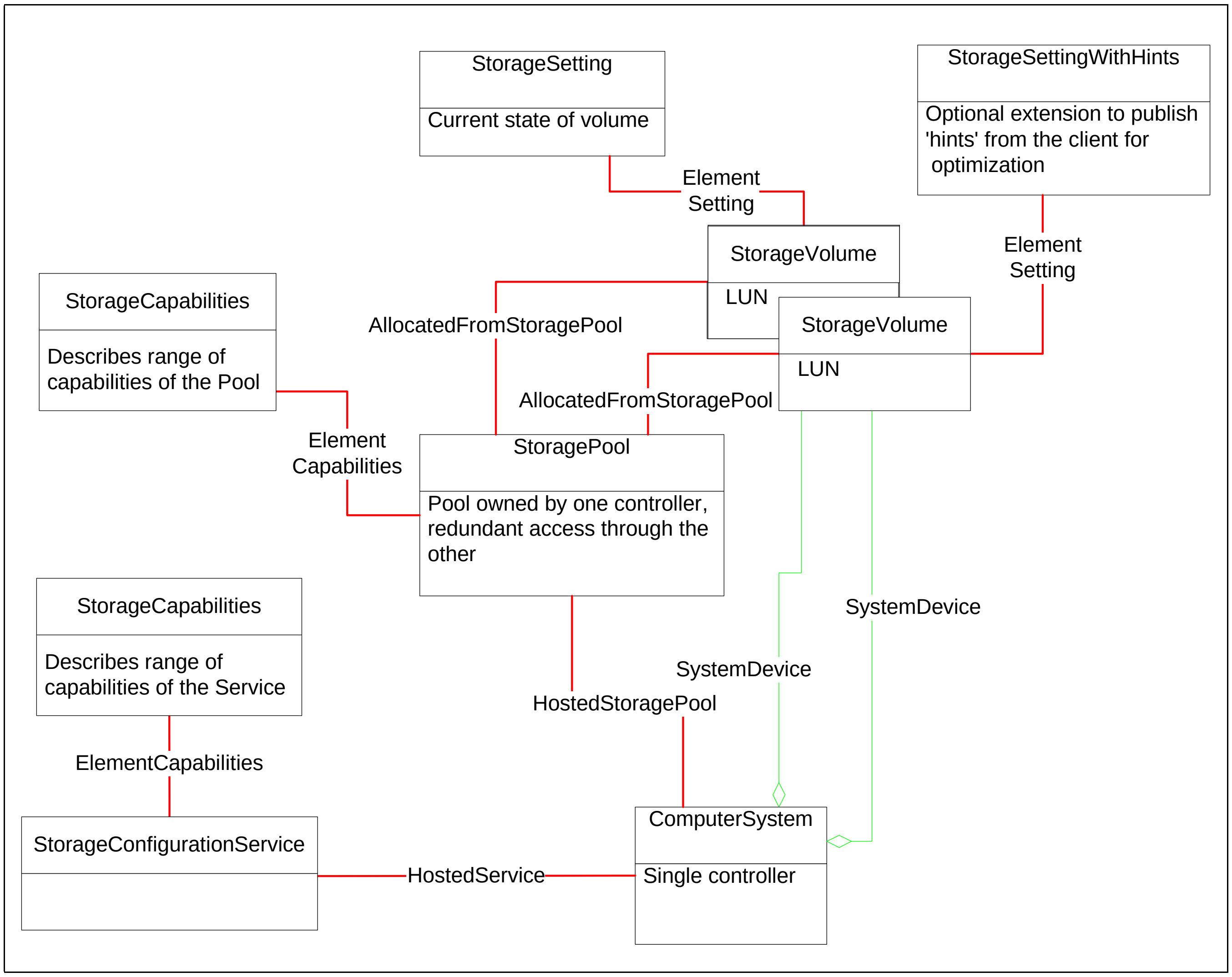

Figure 16 - Representative Block Service Instance Diagram

5.6.2 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 12: 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.

5.6.3 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.

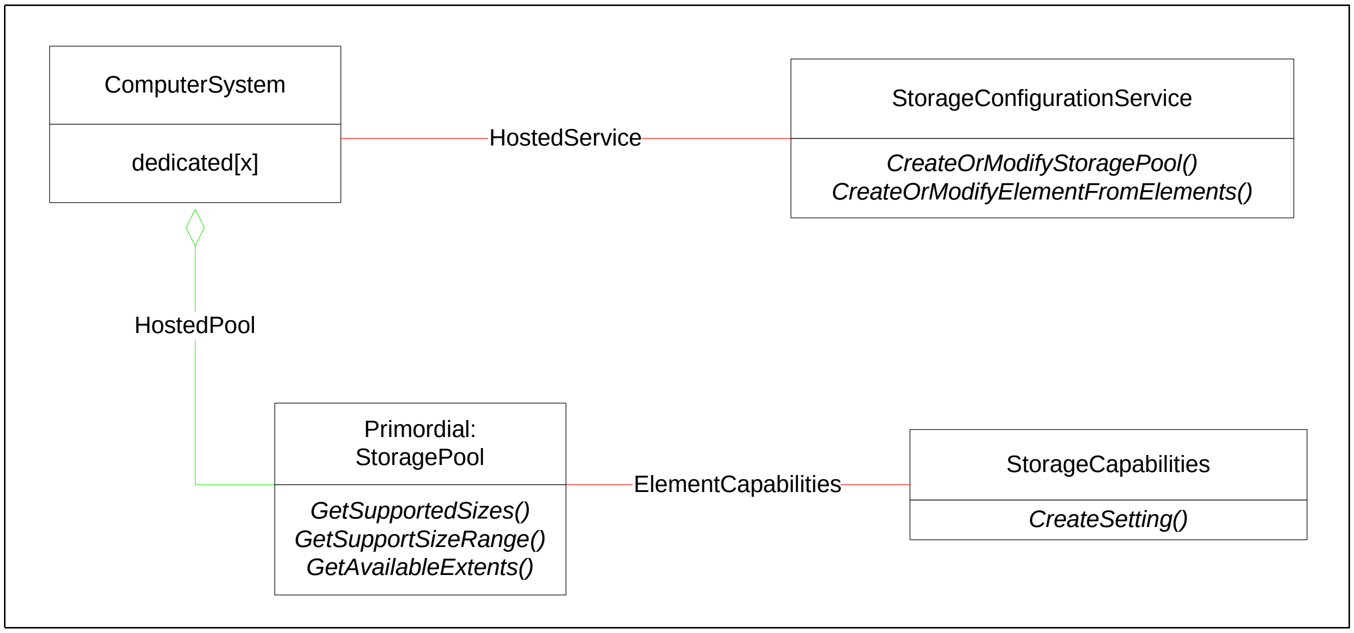

Figure 17 - StoragePool Creation - Initial State

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.

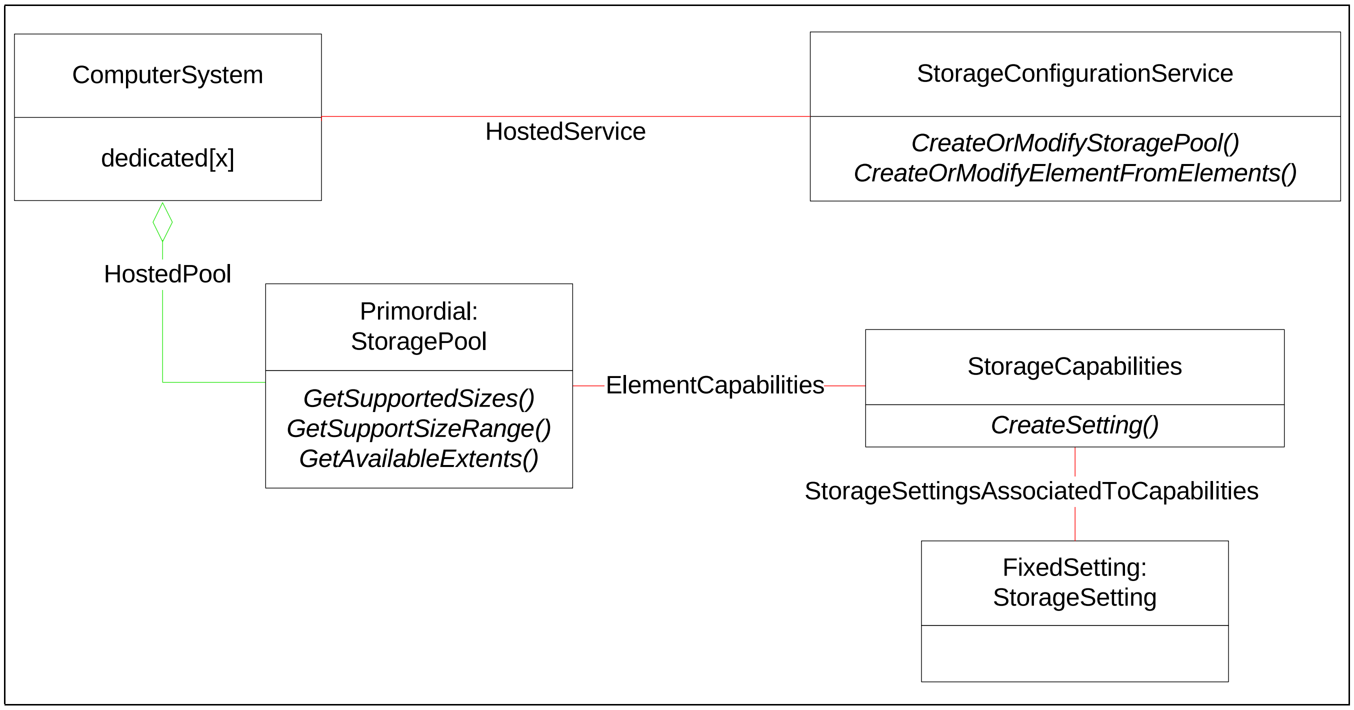

Figure 18 - StoragePool Creation - Step 1

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").

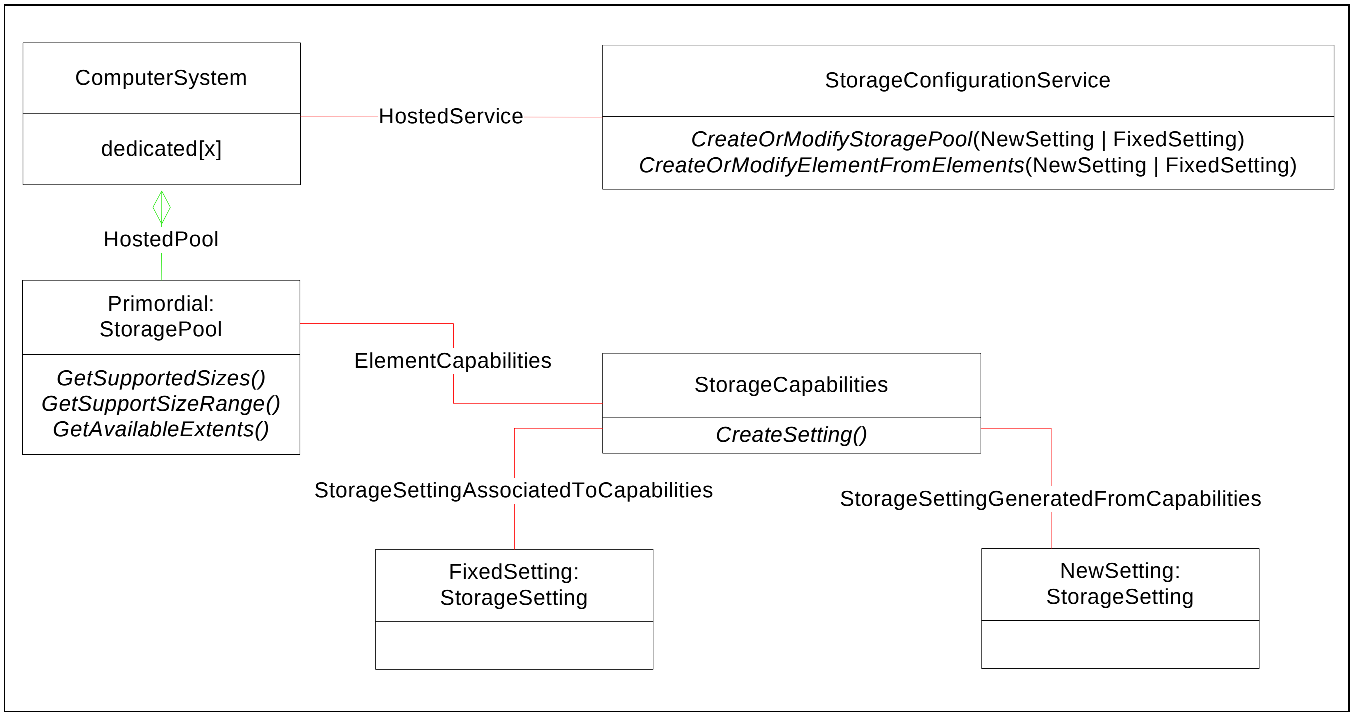
ComputerSystem
dedicated[x]
HostedService
StorageConfigurationService
CreateOrModifyStoragePool(NewSetting | FixedSetting)
CreateOrModifyElementFromElements(NewSetting | FixedSetting)
HostedPool
Primordial: StoragePool
GetSupportedSizes()
GetSupportSizeRange()
GetAvailableExtents()
ElementCapabilities
StorageCapabilities
CreateSetting()
StorageSettingAssociatedToCapabilities
StorageSettingGeneratedFromCapabilities
FixedSetting: StorageSetting
NewSetting: StorageSetting

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 5.1.15.

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. .

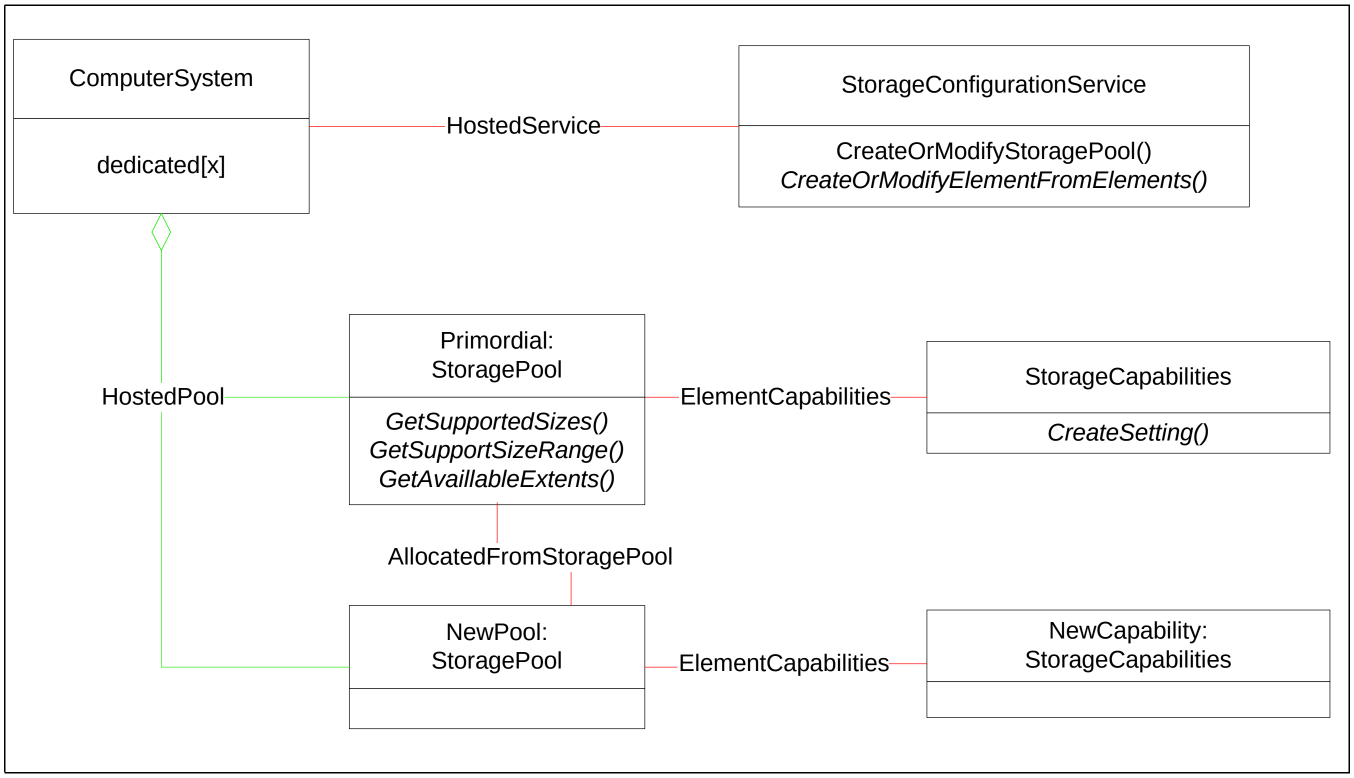

Figure 20 - StoragePool Creation - Step 3

5.6.4 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".

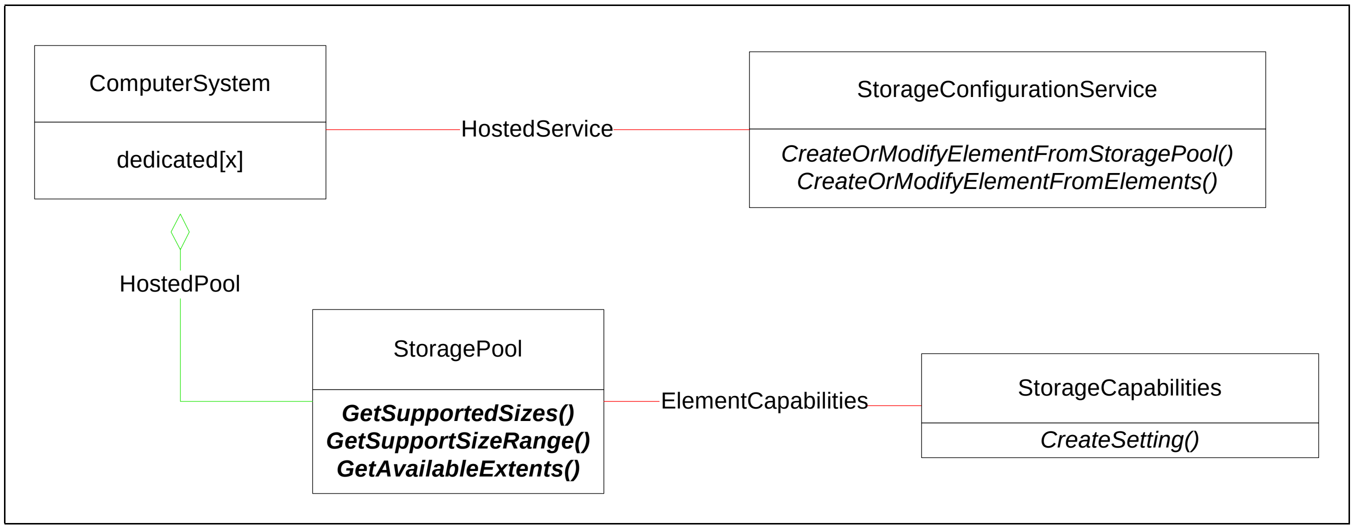

Figure 21 - StorageVolume Creation - Initial State

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.

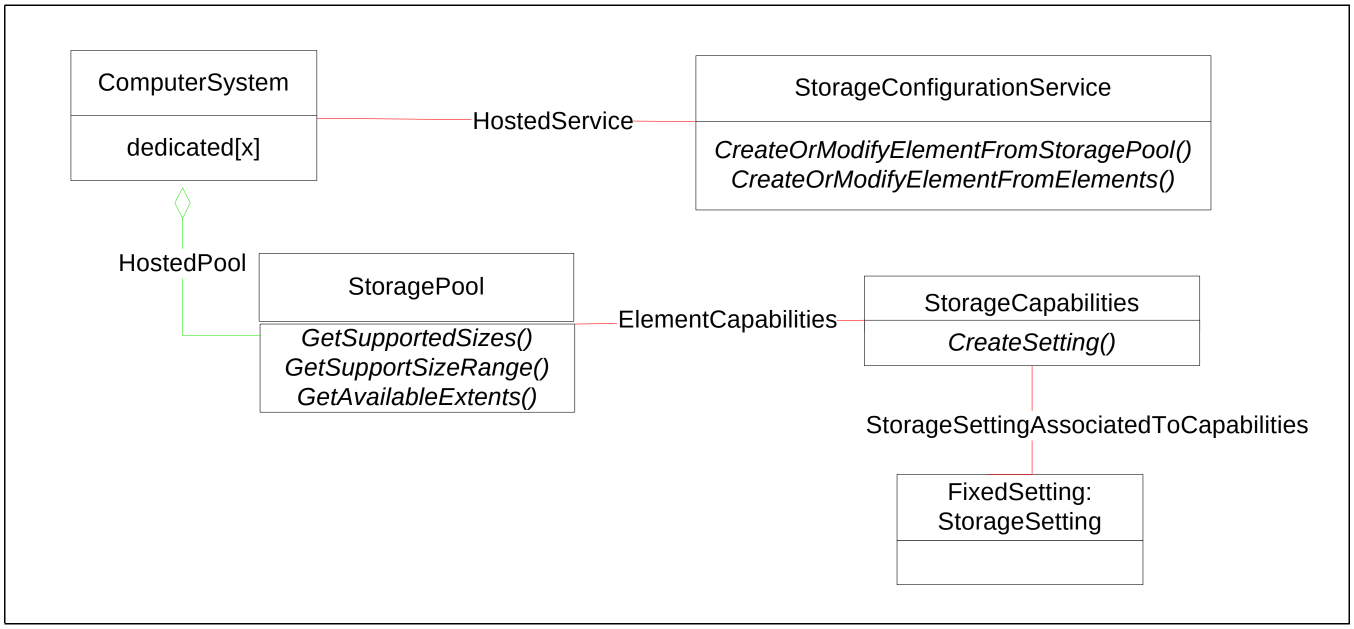

Figure 22 - StorageVolume Creation - Step 1

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 5.1.15.

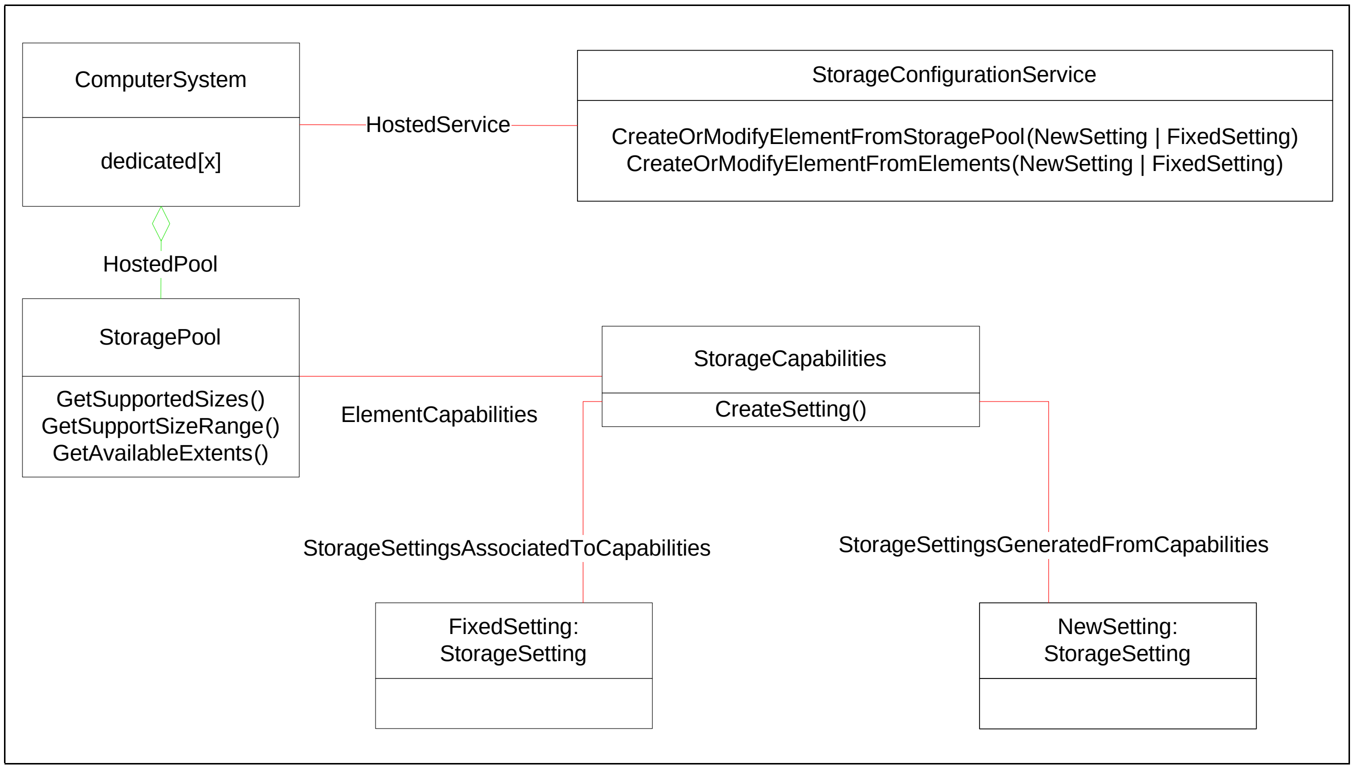
ComputerSystem
dedicated[x]
HostedService
StorageConfigurationService
CreateOrModifyElementFromStoragePool(NewSetting | FixedSetting)
CreateOrModifyElementFromElements(NewSetting | FixedSetting)
HostedPool
StoragePool
GetSupportedSizes()
GetSupportSizeRange()
GetAvailableExtents()
ElementCapabilities
StorageCapabilities
CreateSetting()
StorageSettingsAssociatedToCapabilities
StorageSettingsGeneratedFromCapabilities
FixedSetting: StorageSetting
NewSetting: StorageSetting

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".

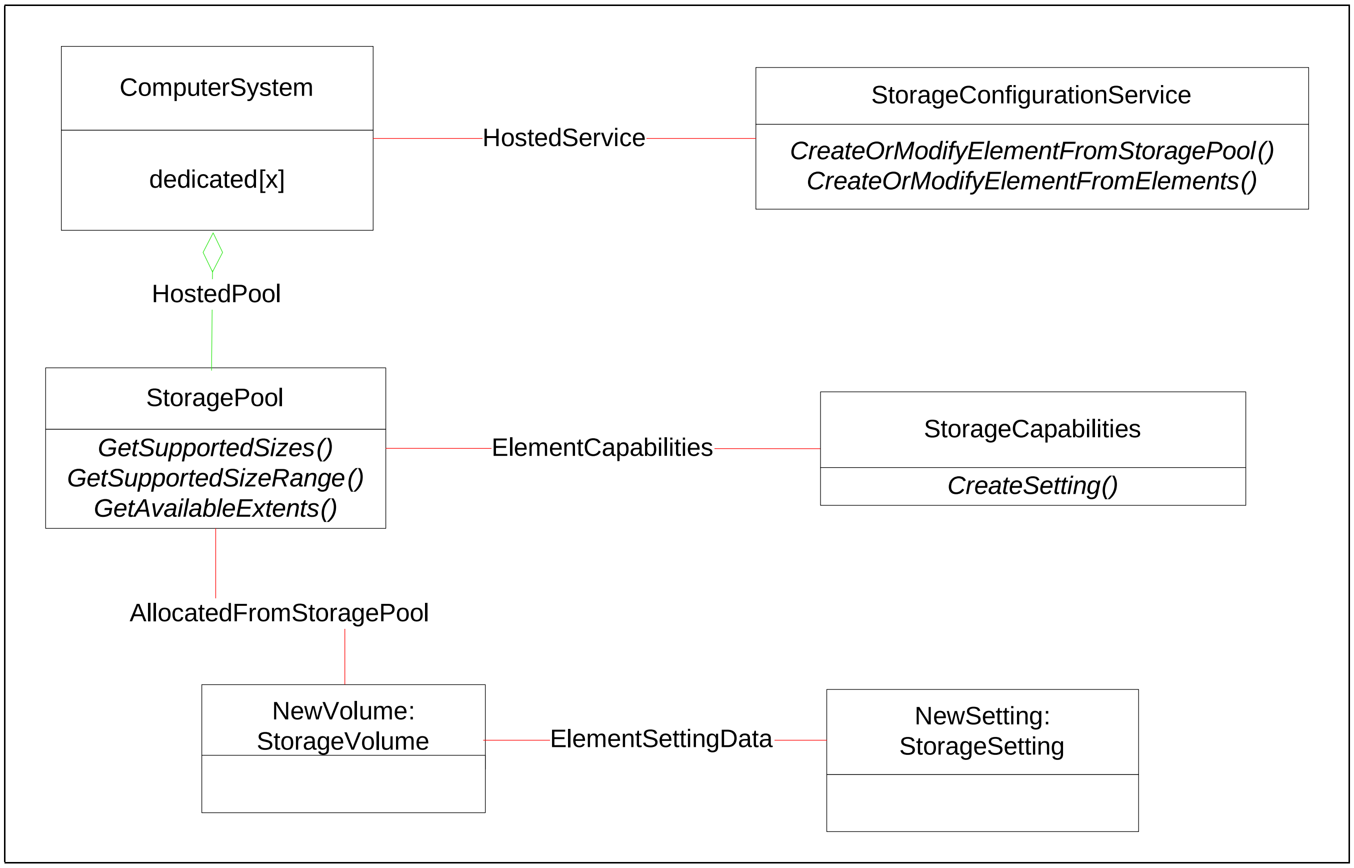

Figure 24 - StorageVolume Creation - Step 3

5.6.5 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>
    }
}
```

5.6.6 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 DataRedundanctMax, 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].SupportedAsyncronousActions[])
#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
//   asociation 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]->,
        ÒCIM_ElementCapabilitiesÓ,
         ÒCIM_StorageConfigurationCapabilitiesÓ,
        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].SupportedAsyncronousActions[])
              #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].Primodial == #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 coersed from %OutArguments[“Job”]
        “CIM_AffectedJobElement”,
        #StorageElementClass,
        null,
        null,
        false,
        false,
        null)
     // Only one storage element will be created,
    $CreatedElement-> = $CreatedElement[0].getObjectPath()
}
```

5.6.7 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 coersed 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 >
     }
}
```

5.6.8 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
// createdlike 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].SupportedAsyncronousActions[])
#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
//   asociation 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].Primodial == #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()
}
```

5.6.9 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 StorageConfiguratonServivce-> 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 >
    }
```

5.7 Registered Name and Version

Block Services version 1.6.0 (Component Profile)

CIM Schema Version: 2.25

5.8 CIM Elements

Table 32 describes the CIM elements for Block Services.

Table 32 - CIM Elements for Block Services

Element Name	Requirement	Description
5.8.1 CIM_AllocatedFromStoragePool (Pool from Pool)	Mandatory	AllocatedFromStoragePool.
5.8.2 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.
5.8.3 CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StorageVolume or LogicalDisk)	Optional	Expressed the ability for the element to be named or have its state changed.
5.8.4 CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StoragePool)	Optional	Expressed the ability for the element to be named or have its state changed.

Table 32 - CIM Elements for Block Services

Element Name	Requirement	Description
5.8.5 CIM_ElementCapabilities (StorageCapabilities to StorageConfigurationService)	Optional	Associates StorageCapabilities with StorageConfigurationService. This StorageCapabilities shall represent the capabilities of the entire implementation.
5.8.6 CIM_ElementCapabilities (StorageCapabilities to StoragePool)	Mandatory	Associates StorageCapabilities with StoragePool. This StorageCapabilities shall represent the capabilities of the StoragePool to which it is associated.
5.8.7 CIM_ElementCapabilities (StorageConfigurationCapabilities to StorageConfigurationService)	Mandatory	Associates StorageConfigurationCapabilities with StorageConfigurationService.
5.8.8 CIM_ElementCapabilities (StorageConfigurationCapabilities to concrete StoragePool)	Optional	Associates StorageConfigurationCapabilities with StoragePool.
5.8.9 CIM_ElementCapabilities (StorageConfigurationCapabilities to primordial StoragePool)	Optional	Associates StorageConfigurationCapabilities with StoragePool.
5.8.10 CIM_ElementCapabilities (Used to declare the naming capabilities of the StoragePool)	Optional	Associates EnabledLogicalElementCapabilities with StorageConfigurationService.
5.8.11 CIM_ElementCapabilities (Used to declare the naming capabilities of the StorageVolume or LogicalDisk)	Optional	Associates EnabledLogicalElementCapabilities with StorageConfigurationService.
5.8.12 CIM_ElementSettingData	Mandatory	
5.8.13 CIM_EnabledLogicalElementCapabilities (For StorageConfigurationService)	Optional	This class is used to express the naming and possible requested state change possibilities for storage elements.
5.8.14 CIM_EnabledLogicalElementCapabilities (For StoragePool)	Optional	This class is used to express the naming and possible requested state change possibilities for storage pools.
5.8.15 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.
5.8.16 CIM_FilterCollection (Block Services ProfileSpecificLifecycleIndicationFilterCollection)	Optional	Experimental. This is a collection of Lifecycle IndicationFilters defined by the Block Services Profile.

Table 32 - CIM Elements for Block Services

Element Name	Requirement	Description
5.8.17 CIM_HostedCollection (Block Services to ProfileSpecificLifecycleIndicationFilterCollection)	Optional	Experimental.
5.8.18 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).
5.8.19 CIM_HostedService	Conditional	Conditional requirement: Support for StorageConfigurationService.
5.8.20 CIM_HostedStoragePool	Mandatory	
5.8.21 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.
5.8.22 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.
5.8.23 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.
5.8.24 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 32 - CIM Elements for Block Services

Element Name	Requirement	Description
5.8.25 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.
5.8.26 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.
5.8.27 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.
5.8.28 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.
5.8.29 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 32 - CIM Elements for Block Services

Element Name	Requirement	Description
5.8.30 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.
5.8.31 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.
5.8.32 CIM_LogicalDisk	Conditional	Conditional requirement: Referenced from Volume Management - LogicalDisk is mandatory. A LogicalDisk is allocated from a concrete StoragePool.
5.8.33 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).
5.8.34 CIM_MemberOfCollection (Block Services ProfileSpecificLifecycleIndicationFilterCollection to Block Services Filters)	Optional	Experimental. This associates the Block Services ProfileSpecificLifecycleIndicationFilterCollection to the Filters defined by the Block Services Profile.
5.8.35 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.
5.8.36 CIM_OwningJobElement	Conditional	Conditional requirement: Support for Job Control profile.

Table 32 - CIM Elements for Block Services

Element Name	Requirement	Description
5.8.37 CIM_StorageCapabilities	Mandatory	
5.8.38 CIM_StorageConfigurationCapabilities (Concrete)	Conditional	Conditional requirement: Support for the Storage Relocation profile.
5.8.39 CIM_StorageConfigurationCapabilities (Global)	Conditional	Conditional requirement: Support for StorageConfigurationService.
5.8.40 CIM_StorageConfigurationCapabilities (Primordial)	Conditional	Conditional requirement: Support for the Storage Relocation profile.
5.8.41 CIM_StorageConfigurationService	Optional	
5.8.42 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.
5.8.43 CIM_StoragePool (Empty)	Optional	An empty StoragePool is a special case of a StoragePool (Concrete or Primordial) where the StoragePool contains no capacity.
5.8.44 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.
5.8.45 CIM_StorageSetting	Mandatory	
5.8.46 CIM_StorageSettingWithHints	Optional	
5.8.47 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").

Table 32 - CIM Elements for Block Services

Element Name	Requirement	Description
5.8.48 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.
5.8.49 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 the Storage Management Technical Specification, Part 1 Common Architecture for details on how to set Name, NameFormat, and NameNamespace properties.
5.8.50 CIM_SystemDevice (System to StorageVolume or LogicalDisk)	Mandatory	Associates top level system from Array, Virtualizer, ... to StorageVolume or LogicalDisk.
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_StoragePool	Mandatory	Creation/Deletion of StoragePool. See section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.24 CIM_IndicationFilter (Storage Pool Creation).
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_StoragePool	Mandatory	Deletion of StoragePool. See section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.25 CIM_IndicationFilter (Storage Pool Deletion).

Table 32 - CIM Elements for Block Services

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 section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.27 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 section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.28 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 section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.31 CIM_IndicationFilter (WQL Storage Volume OperationalStatus).
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 section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.29 CIM_IndicationFilter (Storage Volume OperationalStatus).

Table 32 - CIM Elements for Block Services

Element Name	Requirement	Description
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 section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.21 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 section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.22 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 section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.30 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 section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.23 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 section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.26 CIM_IndicationFilter (Storage Pool TotalManagedSpace).

5.8.1 CIM_AllocatedFromStoragePool (Pool from Pool)

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

Requirement: Mandatory

Table 33 describes class CIM_AllocatedFromStoragePool (Pool from Pool).

Table 33 - SMI Referenced Properties/Methods for CIM_AllocatedFromStoragePool (Pool from Pool)

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

5.8.2 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 34 describes class CIM_AllocatedFromStoragePool (Volume or LogicalDisk from Pool).

Table 34 - SMI Referenced Properties/Methods for CIM_AllocatedFromStoragePool (Volume or LogicalDisk from Pool)

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

5.8.3 CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StorageVolume or LogicalDisk)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 35 describes class CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StorageVolume or LogicalDisk).

Table 35 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StorageVolume or LogicalDisk)

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

5.8.4 CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StoragePool)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 36 describes class CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StoragePool).

Table 36 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StoragePool)

Properties	Flags	Requirement	Description & 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.

5.8.5 CIM_ElementCapabilities (StorageCapabilities to StorageConfigurationService)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 37 describes class CIM_ElementCapabilities (StorageCapabilities to StorageConfigurationService).

Table 37 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (StorageCapabilities to StorageConfigurationService)

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

5.8.6 CIM_ElementCapabilities (StorageCapabilities to StoragePool)

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

Table 38 describes class CIM_ElementCapabilities (StorageCapabilities to StoragePool).

Table 38 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (StorageCapabilities to StoragePool)

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

5.8.7 CIM_ElementCapabilities (StorageConfigurationCapabilities to StorageConfigurationService)

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

Table 39 describes class CIM_ElementCapabilities (StorageConfigurationCapabilities to StorageConfigurationService).

Table 39 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (StorageConfigurationCapabilities to StorageConfigurationService)

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

5.8.8 CIM_ElementCapabilities (StorageConfigurationCapabilities to concrete StoragePool)

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

Table 40 describes class CIM_ElementCapabilities (StorageConfigurationCapabilities to concrete StoragePool).

Table 40 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (StorageConfigurationCapabilities to concrete StoragePool)

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

5.8.9 CIM_ElementCapabilities (StorageConfigurationCapabilities to primordial StoragePool)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 41 describes class CIM_ElementCapabilities (StorageConfigurationCapabilities to primordial StoragePool).

Table 41 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (StorageConfigurationCapabilities to primordial StoragePool)

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

5.8.10 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 42 describes class CIM_ElementCapabilities (Used to declare the naming capabilities of the StoragePool).

Table 42 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (Used to declare the naming capabilities of the StoragePool)

Properties	Flags	Requirement	Description & 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.

5.8.11 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 43 describes class CIM_ElementCapabilities (Used to declare the naming capabilities of the StorageVolume or LogicalDisk).

Table 43 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (Used to declare the naming capabilities of the StorageVolume or LogicalDisk)

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

5.8.12 CIM_ElementSettingData

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 44 describes class CIM_ElementSettingData.

Table 44 - SMI Referenced Properties/Methods for CIM_ElementSettingData

Properties	Flags	Requirement	Description & 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.

5.8.13 CIM_EnabledLogicalElementCapabilities (For StorageConfigurationService)

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

Table 45 describes class CIM_EnabledLogicalElementCapabilities (For StorageConfigurationService).

Table 45 - SMI Referenced Properties/Methods for CIM_EnabledLogicalElementCapabilities (For StorageConfigurationService)

Properties	Flags	Requirement	Description & 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 Capacilities.
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.

Table 45 - SMI Referenced Properties/Methods for CIM_EnabledLogicalElementCapabilities (For StorageConfigurationService)

Properties	Flags	Requirement	Description & Notes
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.

5.8.14 CIM_EnabledLogicalElementCapabilities (For StoragePool)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 46 describes class CIM_EnabledLogicalElementCapabilities (For StoragePool).

Table 46 - SMI Referenced Properties/Methods for CIM_EnabledLogicalElementCapabilities (For StoragePool)

Properties	Flags	Requirement	Description & 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, it may be assumed that the state may not be changed.

5.8.15 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 47 describes class CIM_FilterCollection (Block Services Predefined FilterCollection).

Table 47 - SMI Referenced Properties/Methods for CIM_FilterCollection (Block Services Predefined FilterCollection)

Properties	Flags	Requirement	Description & 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:Predefined'.

5.8.16 CIM_FilterCollection (Block Services ProfileSpecificLifecycleIndicationFilterCollection)

Experimental. This is a collection of Lifecycle IndicationFilters defined by the Block Services Profile.

Requirement: Optional

Table 48 describes class CIM_FilterCollection (Block Services ProfileSpecificLifecycleIndicationFilterCollection).

Table 48 - SMI Referenced Properties/Methods for CIM_FilterCollection (Block Services ProfileSpecificLifecycleIndicationFilterCollection)

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	See the InstanceID definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.13 CIM_FilterCollection (StaticFilterCollection).
CollectionName		Mandatory	The value of CollectionName shall be 'SNIA:Block Services:ProfileSpecificLifecycleIndicationFilterCollection'.

5.8.17 CIM_HostedCollection (Block Services to ProfileSpecificLifecycleIndicationFilterCollection)

Experimental.

Requirement: Optional

Table 49 describes class CIM_HostedCollection (Block Services to ProfileSpecificLifecycleIndicationFilterCollection).

Table 49 - SMI Referenced Properties/Methods for CIM_HostedCollection (Block Services to ProfileSpecificLifecycleIndicationFilterCollection)

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	Reference to the ProfileSpecificLifecycleIndicationFilterCollection for the Block Services Profile.
Antecedent		Mandatory	Reference to the 'Top level' System.

5.8.18 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 50 describes class CIM_HostedCollection (System to predefined IndicationFilters).

Table 50 - SMI Referenced Properties/Methods for CIM_HostedCollection (System to predefined IndicationFilters)

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

5.8.19 CIM_HostedService

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Support for StorageConfigurationService.

Table 51 describes class CIM_HostedService.

Table 51 - SMI Referenced Properties/Methods for CIM_HostedService

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

5.8.20 CIM_HostedStoragePool

Requirement: Mandatory

Table 52 describes class CIM_HostedStoragePool.

Table 52 - SMI Referenced Properties/Methods for CIM_HostedStoragePool

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

5.8.21 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 53 describes class CIM_IndicationFilter (Logical Disk Creation).

Table 53 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Logical Disk Creation)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Name		Mandatory	This shall be 'SNIA:Block Services:LogicalDiskCreation'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
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 section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).

5.8.22 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 54 describes class CIM_IndicationFilter (Logical Disk Deletion).

Table 54 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Logical Disk Deletion)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Name		Mandatory	This shall be 'SNIA:Block Services:LogicalDiskDeletion'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
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 section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).

5.8.23 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 55 describes class CIM_IndicationFilter (Logical Disk OperationalStatus).

Table 55 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Logical Disk Operational-Status)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Name		Mandatory	This shall be 'SNIA:Block Services:LogicalDiskOperationalStatus'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
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 section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).

5.8.24 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 56 describes class CIM_IndicationFilter (Storage Pool Creation).

Table 56 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Storage Pool Creation)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Name		Mandatory	This shall be 'SNIA:Block Services:StoragePoolCreation'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
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 section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).

5.8.25 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 57 describes class CIM_IndicationFilter (Storage Pool Deletion).

Table 57 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Storage Pool Deletion)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Name		Mandatory	This shall be 'SNIA:Block Services:StoragePoolDeletion'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
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 section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).

5.8.26 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 58 describes class CIM_IndicationFilter (Storage Pool TotalManagedSpace).

Table 58 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Storage Pool TotalManagedSpace)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Name		Mandatory	This shall be 'SNIA:Block Services:StoragePoolTotalManagedSpace'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
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 section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).

5.8.27 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 59 describes class CIM_IndicationFilter (Storage Volume Creation).

Table 59 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Storage Volume Creation)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Name		Mandatory	This shall be 'SNIA:Block Services:StorageVolumeCreation'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
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 section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).

5.8.28 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 60 describes class CIM_IndicationFilter (Storage Volume Deletion).

Table 60 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Storage Volume Deletion)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Name		Mandatory	This shall be 'SNIA:Block Services:StorageVolumeDeletion'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
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 section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).

5.8.29 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 61 describes class CIM_IndicationFilter (Storage Volume OperationalStatus).

Table 61 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Storage Volume OperationalStatus)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Name		Mandatory	This shall be 'SNIA:Block Services:StorageVolumeOperationalStatus'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
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 section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).

5.8.30 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 62 describes class CIM_IndicationFilter (WQL Logical Disk OperationalStatus).

Table 62 - SMI Referenced Properties/Methods for CIM_IndicationFilter (WQL Logical Disk OperationalStatus)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Name		Mandatory	This shall be 'SNIA:Block Services:LogicalDiskOperationalStatusWQL'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
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 section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).

5.8.31 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 63 describes class CIM_IndicationFilter (WQL Storage Volume OperationalStatus).

Table 63 - SMI Referenced Properties/Methods for CIM_IndicationFilter (WQL Storage Volume OperationalStatus)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Name		Mandatory	This shall be 'SNIA:Block Services:StorageVolumeOperationalStatusWQL'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
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 section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).

5.8.32 CIM_LogicalDisk

Created By: Static
Modified By: Static
Deleted By: Extrinsic: StorageConfigurationService.ReturnToStoragePool
Requirement: Referenced from Volume Management - LogicalDisk is mandatory.

Table 64 describes class CIM_LogicalDisk.

Table 64 - SMI Referenced Properties/Methods for CIM_LogicalDisk

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	Opaque identifier.
ElementName		Optional	User-friendly name.
Name		Mandatory	OS Device Name.
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). In addition, the secondary OperationalStatus may be 19 (Relocating) with 2|3|15 (OK or Degraded 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".

Table 64 - SMI Referenced Properties/Methods for CIM_LogicalDisk

Properties	Flags	Requirement	Description & Notes
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'.
NumExtentsMigrating		Optional	Experimental. The number of Extents in the process of migrating for this logical disk when the logical disk relocation is on going.

5.8.33 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 65 describes class CIM_MemberOfCollection (Block Services Filter Collection to FilterCollection).

Table 65 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Block Services Filter Collection to FilterCollection)

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

5.8.34 CIM_MemberOfCollection (Block Services ProfileSpecificLifecycleIndicationFilterCollection to Block Services Filters)

Experimental. This associates the Block Services ProfileSpecificLifecycleIndicationFilterCollection to the Filters defined by the Block Services Profile.

Requirement: Optional

Table 66 describes class CIM_MemberOfCollection (Block Services ProfileSpecificLifecycleIndicationFilterCollection to Block Services Filters).

Table 66 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Block Services ProfileSpecificLifecycleIndicationFilterCollection to Block Services Filters)

Properties	Flags	Requirement	Description & Notes
Collection		Mandatory	Reference to the Block Services ProfileSpecificLifecycleIndicationFilterCollection.
Member		Mandatory	Reference to the IndicationFilters defined by the Block Services Profile.

5.8.35 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 67 describes class CIM_MemberOfCollection (Predefined Filter Collection to Block Services Filters).

Table 67 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Predefined Filter Collection to Block Services Filters)

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

5.8.36 CIM_OwningJobElement

Conditional on support for Job Control profile.

Requirement: Support for Job Control profile.

Table 68 describes class CIM_OwningJobElement.

Table 68 - SMI Referenced Properties/Methods for CIM_OwningJobElement

Properties	Flags	Requirement	Description & Notes
OwnedElement		Mandatory	
OwningElement		Mandatory	

5.8.37 CIM_StorageCapabilities

Created By: Static
Modified By: Static

Deleted By: Static
Requirement: Mandatory

Table 69 describes class CIM_StorageCapabilities.

Table 69 - SMI Referenced Properties/Methods for CIM_StorageCapabilities

Properties	Flags	Requirement	Description & 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).
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.

Table 69 - SMI Referenced Properties/Methods for CIM_StorageCapabilities

Properties	Flags	Requirement	Description & Notes
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.
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.

5.8.38 CIM_StorageConfigurationCapabilities (Concrete)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Support for the Storage Relocation profile.

Table 70 describes class CIM_StorageConfigurationCapabilities (Concrete).

Table 70 - SMI Referenced Properties/Methods for CIM_StorageConfigurationCapabilities (Concrete)

Properties	Flags	Requirement	Description & 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		Conditional	Conditional requirement: Support for the Storage Relocation profile. Lists what actions, invoked through StorageConfigurationService methods, shall not produce Concrete jobs. This version of the 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) or "17" (StorageVolume To StoragePool Relocation) or "18" (StoragePool To StoragePool Relocation) or "19" (StorageVolume To StorageExtent Relocation) or "20" (StoragePool To StorageExtent Relocation) or "21" (LogicalDisk To StorageExtent Relocation) or "22" (Multiple Storage Element Creation) or "23" (Multiple Storage Element Return).
SupportedStorageElementTypes		Mandatory	Lists the type of storage elements that are supported by this implementation. This version of the standard recognizes '2' (StorageVolume) or '4' (LogicalDisk).

Table 70 - SMI Referenced Properties/Methods for CIM_StorageConfigurationCapabilities (Concrete)

Properties	Flags	Requirement	Description & Notes
SupportedAsynchronousActions		Conditional	Conditional requirement: Support for the Storage Relocation profile. Lists what actions, invoked through StorageConfigurationService methods, may produce Concrete jobs. This version of the 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) or "17" (StorageVolume To StoragePool Relocation) or "18" (StoragePool To StoragePool Relocation) or "19" (StorageVolume To StorageExtent Relocation) or "20" (StoragePool To StorageExtent Relocation) or "21" (LogicalDisk To StorageExtent Relocation) or "22" (Multiple Storage Element Creation) or "23" (Multiple Storage Element Return)..
SupportedStorageElementFeatures		Conditional	Conditional requirement: Support for the Storage Relocation profile. Lists actions supported through the invocation of StorageServiceService.CreateOrModifyElementFromStoragePool(). Matches 3|8|14|15|16|17|18 (StorageVolume Creation or LogicalDisk Creation or StorageVolume To StoragePool Relocation or StoragePool To StoragePool Relocation or StorageVolume To StorageExtent Relocation or StoragePool To StorageExtent Relocation LogicalDisk To StorageExtent Relocation).
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		Conditional	Conditional requirement: Support for the Storage Relocation profile. Indicates the intended usage or any restrictions that may have been imposed on storage pools.
ClientSettablePoolUsage		Conditional	Conditional requirement: Support for the Storage Relocation profile. Indicates the intended usage or any restrictions that may have been imposed on the usage of a client-settable storage pool.
MaximumElementCreateCount		Optional	Indicates the maximum number of elements that can be specified to be created in a single method call. If 0 or null, there is no limit.
MaximumElementDeleteCount		Conditional	Conditional requirement: Support for the Storage Relocation profile. Indicates the maximum number of elements that can be deleted in a single method call. If 0 or null, there is no limit.

Table 70 - SMI Referenced Properties/Methods for CIM_StorageConfigurationCapabilities (Concrete)

Properties	Flags	Requirement	Description & Notes
MultipleElementCreateFeatures		Optional	Enumeration indicating features offered by the multiple element create method. "2" (Single instance creation indication).
MultipleElementDeleteFeatures		Optional	Enumeration indicating features offered by the multiple element delete method. "2" (Continue on nonexistent element) or "3" (Return error on nonexistent element).

5.8.39 CIM_StorageConfigurationCapabilities (Global)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Support for StorageConfigurationService.

Table 71 describes class CIM_StorageConfigurationCapabilities (Global).

Table 71 - SMI Referenced Properties/Methods for CIM_StorageConfigurationCapabilities (Global)

Properties	Flags	Requirement	Description & 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		Conditional	Conditional requirement: Support for the Storage Relocation profile. 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		Conditional	Conditional requirement: Support for the Storage Relocation profile. Lists what actions, invoked through StorageConfigurationService methods, may produce Concrete jobs.

Table 71 - SMI Referenced Properties/Methods for CIM_StorageConfigurationCapabilities (Global)

Properties	Flags	Requirement	Description & Notes
SupportedStorageElementFeatures		Conditional	Conditional requirement: Support for the Storage Relocation profile. Lists actions supported through the invocation of StorageServiceService.CreateOrModifyElementFromStoragePool(). Matches 3|5|8|9|11|12|13|14|15|16|17|18 (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 or StorageVolume To StoragePool Relocation or StoragePool To StoragePool Relocation or StorageVolume To StorageExtent Relocation or 'StoragePool To StorageExtent Relocation or LogicalDisk To StorageExtent Relocation).
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		Conditional	Conditional requirement: Support for the Storage Relocation profile. Indicates the intended usage or any restrictions that may have been imposed on storage pools.
ClientSettablePoolUsage		Conditional	Conditional requirement: Support for the Storage Relocation profile. Indicates the intended usage or any restrictions that may have been imposed on the usage of a client-settable storage pool.
MaximumElementCreateCount		Optional	Indicates the maximum number of elements that can be specified to be created in a single method call. If 0 or null, there is no limit.
MaximumElementDeleteCount		Conditional	Conditional requirement: Support for the Storage Relocation profile. Indicates the maximum number of elements that can be deleted in a single method call. If 0 or null, there is no limit.
MultipleElementCreateFeatures		Optional	Enumeration indicating features offered by the multiple element create method. "2" (Single instance creation indication).
MultipleElementDeleteFeatures		Optional	Enumeration indicating features offered by the multiple element delete method. "2" (Continue on nonexistent element) or "3" (Return error on nonexistent element).

5.8.40 CIM_StorageConfigurationCapabilities (Primordial)

Created By: Static
Modified By: Static

Deleted By: Static

Requirement: Support for the Storage Relocation profile.

Table 72 describes class CIM_StorageConfigurationCapabilities (Primordial).

Table 72 - SMI Referenced Properties/Methods for CIM_StorageConfigurationCapabilities (Primordial)

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
ElementName		Mandatory	
SupportedStoragePoolFeatures		Optional	Lists what StorageConfigurationService functionalities are implemented. Matches 2|3 (InExtents or Single InPool).
SupportedSynchronousActions		Conditional	Conditional requirement: Support for the Storage Relocation profile. Lists what actions, invoked through StorageConfigurationService methods, shall not produce Concrete jobs. This version of the standard recognizes "2" (Storage Pool Creation), "12" (Storage Element from Element Creation) or "15" (StoragePool Usage Modification) or "17" (StorageVolume To StoragePool Relocation) or "18" (StoragePool To StoragePool Relocation) or "19" (StorageVolume To StorageExtent Relocation) or "20" (StoragePool To StorageExtent Relocation) or "21" (LogicalDisk To StorageExtent Relocation).
SupportedStorageElementTypes		Optional	Lists the type of storage elements that are supported by this implementation. This version of the standard does not recognize any values for this property.
SupportedAsynchronousActions		Conditional	Conditional requirement: Support for the Storage Relocation profile. Lists what actions, invoked through StorageConfigurationService methods, may produce Concrete jobs. This version of the standard recognizes "2" (Storage Pool Creation), "12" (Storage Element from Element Creation) or "15" (StoragePool Usage Modification) or "17" (StorageVolume To StoragePool Relocation) or "18" (StoragePool To StoragePool Relocation) or "19" (StorageVolume To StorageExtent Relocation) or "20" (StoragePool To StorageExtent Relocation) or "21" (LogicalDisk To StorageExtent Relocation) or "22" (Multiple Storage Element Creation) or "23" (Multiple Storage Element Return).

Table 72 - SMI Referenced Properties/Methods for CIM_StorageConfigurationCapabilities (Primordial)

Properties	Flags	Requirement	Description & Notes
SupportedStorageElementFeatures		Conditional	Conditional requirement: Support for the Storage Relocation profile. Lists actions supported through the invocation of StorageServiceService.CreateOrModifyElementFromStoragePool(). This version of the 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) or 14 (StorageVolume To StoragePool Relocation) or 15 (StoragePool To StoragePool Relocation) or 16 (StorageVolume To StorageExtent Relocation) or 17 (StoragePool To StorageExtent Relocation) or 18 (LogicalDisk To StorageExtent Relocation).
SupportedStorageElementUsage		Optional	For Primordial StorageConfigurationCapabilities, this shall be NULL.
ClientSettableElementUsage		Optional	For Primordial StorageConfigurationCapabilities, this shall be NULL.
SupportedStoragePoolUsage		Conditional	Conditional requirement: Support for the Storage Relocation profile. Indicates the intended usage or any restrictions that may have been imposed on storage pools.
ClientSettablePoolUsage		Conditional	Conditional requirement: Support for the Storage Relocation profile. Indicates the intended usage or any restrictions that may have been imposed on the usage of a client-settable storage pool.

5.8.41 CIM_StorageConfigurationService

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

Table 73 describes class CIM_StorageConfigurationService.

Table 73 - SMI Referenced Properties/Methods for CIM_StorageConfigurationService

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

Table 73 - SMI Referenced Properties/Methods for CIM_StorageConfigurationService

Properties	Flags	Requirement	Description & Notes
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.
CreateElementsFromStoragePool()		Optional	Experimental. Create one or more storage elements. 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.
ReturnElementsToStoragePool()		Optional	Experimental. Release the capacity represented by one or more storage elements 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.

5.8.42 CIM_StoragePool (Concrete)

Created By: Extrinsic: StorageConfigurationService.CreateOrModifyStoragePool
Modified By: Extrinsic: StorageConfigurationService.CreateOrModifyStoragePool
Deleted By: Extrinsic: StorageConfigurationService.DeleteStoragePool
Requirement: Mandatory

Table 74 describes class CIM_StoragePool (Concrete).

Table 74 - SMI Referenced Properties/Methods for CIM_StoragePool (Concrete)

Properties	Flags	Requirement	Description & 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.

Table 74 - SMI Referenced Properties/Methods for CIM_StoragePool (Concrete)

Properties	Flags	Requirement	Description & Notes
OperationalStatus		Conditional	Experimental. Conditional requirement: Support for the Storage Relocation profile. Value shall be 2|3|6|15 (OK or Degraded or Error or Dormant). In addition, the secondary OperationalStatus may be 19 (Relocating) with 2|3|15 (OK or Degraded or Dormant).
TotalManagedSpace		Mandatory	
RemainingManaged Space		Mandatory	
Usage		Optional	The specialized usage intended for this element.
OtherUsageDescripti on		Optional	Set when Usage value is "Other".
ClientSettableUsage		Optional	Lists Usage values that can be set by a client for this element.
CapacityInMigratingS ource		Optional	Experimental. The total capacity of extents migrating out from this storage pool.
CapacityInMigratingT arget		Optional	Experimental. The total capacity of extents migrating into this storage pool.
GetSupportedSizes()		Conditional	Conditional requirement: Support for StorageConfigurationService. List the discrete storage element sizes that can be created or expanded from this Pool.
GetSupportedSizeRa nge()		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.

5.8.43 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 75 describes class CIM_StoragePool (Empty).

Table 75 - SMI Referenced Properties/Methods for CIM_StoragePool (Empty)

Properties	Flags	Requirement	Description & 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	
OperationalStatus		Conditional	Experimental. Conditional requirement: Support for the Storage Relocation profile. Value shall be 2|3|6|15 (OK or Degraded or Error or Dormant). In addition, the secondary OperationalStatus may be 19 (Relocating) with 2|3|15 (OK or Degraded or Dormant).
TotalManagedSpace		Mandatory	This shall be 0 for an empty StoragePool.
RemainingManagedSpace		Mandatory	
Usage		Optional	
OtherUsageDescription		Optional	
ClientSettableUsage		Optional	
CapacityInMigratingSource		Optional	Experimental. The total capacity of extents migrating out from this storage pool.
CapacityInMigratingTarget		Optional	Experimental. The total capacity of extents migrating into this storage pool.
GetSupportedSizes()		Conditional	Conditional requirement: Support for StorageConfigurationService.
GetSupportedSizeRange()		Conditional	Conditional requirement: Support for StorageConfigurationService.
GetAvailableExtents()		Optional	

5.8.44 CIM_StoragePool (Primordial)

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

Table 76 describes class CIM_StoragePool (Primordial).

Table 76 - SMI Referenced Properties/Methods for CIM_StoragePool (Primordial)

Properties	Flags	Requirement	Description & 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.
OperationalStatus		Conditional	Experimental. Conditional requirement: Support for the Storage Relocation profile. Value shall be 2|3|6|15 (OK or Degraded or Error or Dormant). In addition, the secondary OperationalStatus may be 19 (Relocating) with 2|3|15 (OK or Degraded or Dormant).
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.
CapacityInMigratingSource		Optional	Experimental. The total capacity of extents migrating out from this storage pool.
CapacityInMigratingTarget		Optional	Experimental. The total capacity of extents migrating into this storage pool.
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.

5.8.45 CIM_StorageSetting

Created By: Extrinsic: StorageCapabilities.CreateSetting

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 77 describes class CIM_StorageSetting.

Table 77 - SMI Referenced Properties/Methods for CIM_StorageSetting

Properties	Flags	Requirement	Description & 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.

Table 77 - SMI Referenced Properties/Methods for CIM_StorageSetting

Properties	Flags	Requirement	Description & Notes
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.
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.

5.8.46 CIM_StorageSettingWithHints

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

Table 78 describes class CIM_StorageSettingWithHints.

Table 78 - SMI Referenced Properties/Methods for CIM_StorageSettingWithHints

Properties	Flags	Requirement	Description & 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	

Table 78 - SMI Referenced Properties/Methods for CIM_StorageSettingWithHints

Properties	Flags	Requirement	Description & Notes
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	
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.

Table 78 - SMI Referenced Properties/Methods for CIM_StorageSettingWithHints

Properties	Flags	Requirement	Description & Notes
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	

5.8.47 CIM_StorageSettingsAssociatedToCapabilities

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

Table 79 describes class CIM_StorageSettingsAssociatedToCapabilities.

Table 79 - SMI Referenced Properties/Methods for CIM_StorageSettingsAssociatedToCapabilities

Properties	Flags	Requirement	Description & 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.

5.8.48 CIM_StorageSettingsGeneratedFromCapabilities

Created By: Extrinsic: CreateSetting
Modified By: Static
Deleted By: Static
Requirement: Support for StorageConfigurationService.

Table 80 describes class CIM_StorageSettingsGeneratedFromCapabilities.

Table 80 - SMI Referenced Properties/Methods for CIM_StorageSettingsGeneratedFromCapabilities

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

5.8.49 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 81 describes class CIM_StorageVolume.

Table 81 - SMI Referenced Properties/Methods for CIM_StorageVolume

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

Table 81 - SMI Referenced Properties/Methods for CIM_StorageVolume

Properties	Flags	Requirement	Description & 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). In addition, the secondary OperationalStatus may be 19 (Relocating) with 2|3|15 (OK or Degraded 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".

Table 81 - SMI Referenced Properties/Methods for CIM_StorageVolume

Properties	Flags	Requirement	Description & Notes
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'.
CanDelete		Optional	Experimental. Indicates if the volume is able to be deleted by a client application.
NumExtentsMigrating		Optional	Experimental. The number of Extents in the process of migrating for this storage volume when the volume relocation is on going.

5.8.50 CIM_SystemDevice (System to StorageVolume or LogicalDisk)

Created By: Static
Modified By: Static
Deleted By: Extrinsic: StorageConfigurationService.ReturnToStoragePool
Requirement: Mandatory

Table 82 describes class CIM_SystemDevice (System to StorageVolume or LogicalDisk).

Table 82 - SMI Referenced Properties/Methods for CIM_SystemDevice (System to StorageVolume or LogicalDisk)

Properties	Flags	Requirement	Description & Notes
PartComponent		Mandatory	
GroupComponent		Mandatory	

STABLE

EXPERIMENTAL

Clause 6: Block Storage Views Profile

6.1 Description

6.1.1 Synopsis

Profile Name: Block Storage Views (Component Profile)

Version: 1.6.0

Organization: SNIA

CIM Schema Version: 2.29

Table 83 describes the related profiles for Block Storage Views.

Table 83 - Related Profiles for Block Storage Views

Profile Name	Organization	Version	Requirement	Description
Block Services	SNIA	1.6.0	Conditional	Conditional requirement: Required if the array property CIM_ViewCapabilities.SupportedViews contains the string "SNIA:VolumeView" (and the Block Service Package is implemented).
Block Server Performance	SNIA	1.5.0	Conditional	
Disk Drive Lite	SNIA	1.6.0	Conditional	Conditional requirement: Required if the array property CIM_ViewCapabilities.SupportedViews contains the string "SNIA:DiskDriveView" (and the Disk Drive Lite Profile is implemented).
Masking and Mapping	SNIA	1.6.0	Conditional	

Table 83 - Related Profiles for Block Storage Views

Profile Name	Organization	Version	Requirement	Description
Extent Composition	SNIA	1.6.0	Conditional	Conditional requirement: Required if the array property CIM_ViewCapabilities.SupportedViews contains the string "SNIA:VolumeView" and Extent Composition is implemented.
Copy Services	SNIA	1.5.0	Conditional	Conditional requirement: Required if the array property CIM_ViewCapabilities.SupportedViews contains the string "SNIA:ReplicaPairlView" (and the Copy Services Profile is implemented).

Central Class: CIM_ViewCapabilities

Scoping Class: CIM_ComputerSystem

6.1.2 Overview

This Profile specifies View Classes for the Array, Storage Virtualizer, Host Hardware RAID Controller and Volume Management Profiles.

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

6.1.2.1 Goals of View Classes

6.1.2.1.1 Goals that 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 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).

6.1.2.1.2 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 View Classes are NOT intended to help in the latter case. However, View Classes should facilitate access to underlying classes in support of configuration operations.

It is important to note that the View Classes were 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).

6.1.2.2 Specific Requirements and Objectives of View Classes

6.1.2.2.1 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.

6.1.2.2.2 Specific Views requested

- Getting asset information
- Mix of StorageVolume with LUN Mapping & Masking
- Getting port information (with endpoints) or ports & volumes
- Hardware ID & StorageVolumes
- Disk drive view
- Volumes & Settings
- Extent Composition
- Privilege Hierarchy
- Hardware ID <-> StorageVolume

Most of these requests are addressed by this Profile.

- Allow View Classes to be used where real classes would

  This certainly includes "read" intrinsics and as parameters of Extrinsics

  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.

6.1.2.2.3 Support Life Cycle Indications on View Classes

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

6.1.3 Class Diagram for Block Storage Views View Classes

Figure 25: "Class diagram for managed element Block Storage View Classes" illustrates the class diagram for view classes that cover managed elements.

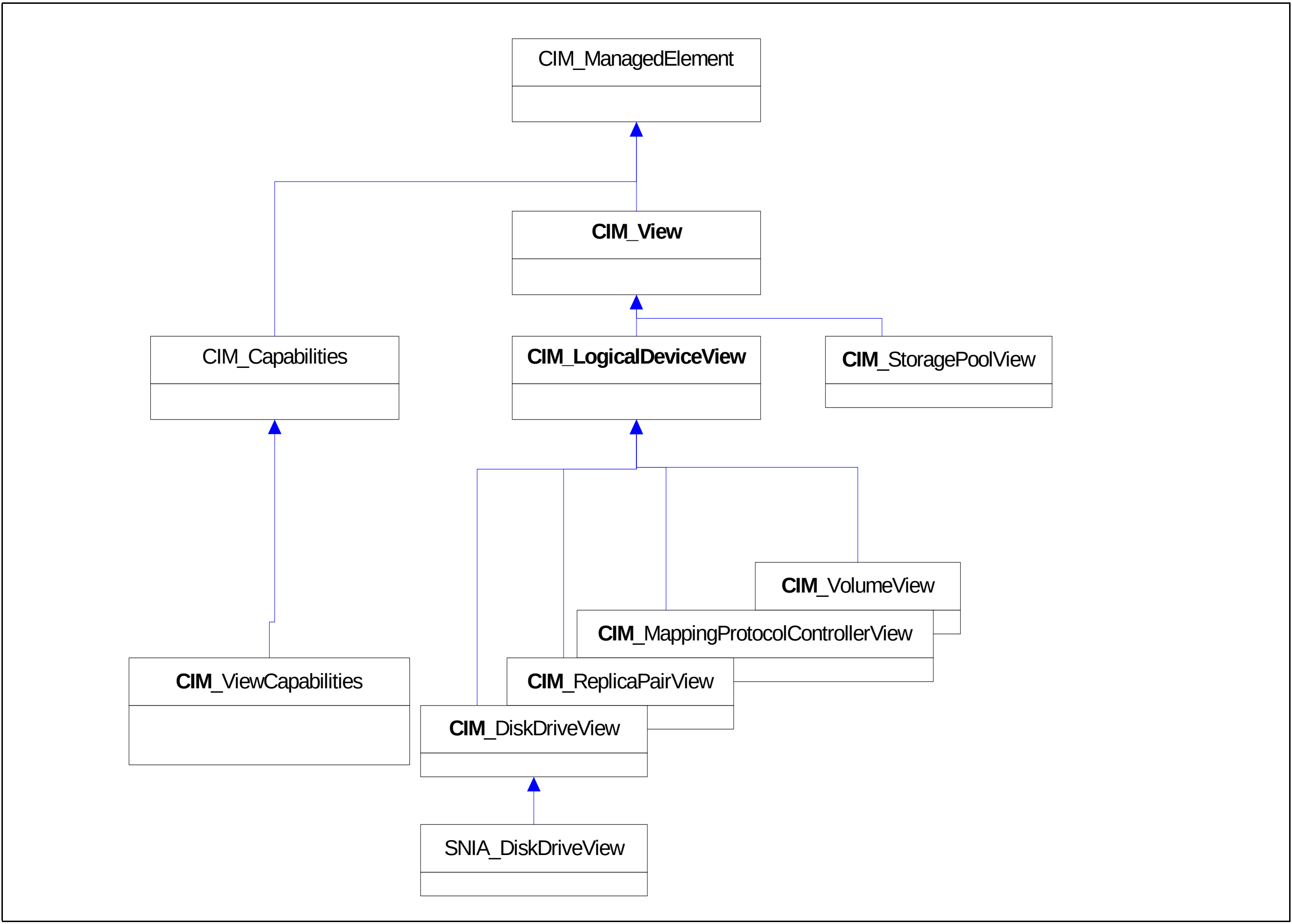

Figure 25 - Class diagram for managed element Block Storage View Classes

The view classes that represent managed elements (e.g., VolumeView and DiskDriveView) inherit from CIM_View. CIM_StoragePoolView inherits directly from CIM_View, while the other managed element view classes (e.g., VolumeView and DiskDriveView) are subclassed from CIM_LogicalDeviceView. The ViewCapabilities inherits from CIM_Capabilities.

Figure 26: "Class diagram for view associations" shows the class diagram for view associations.

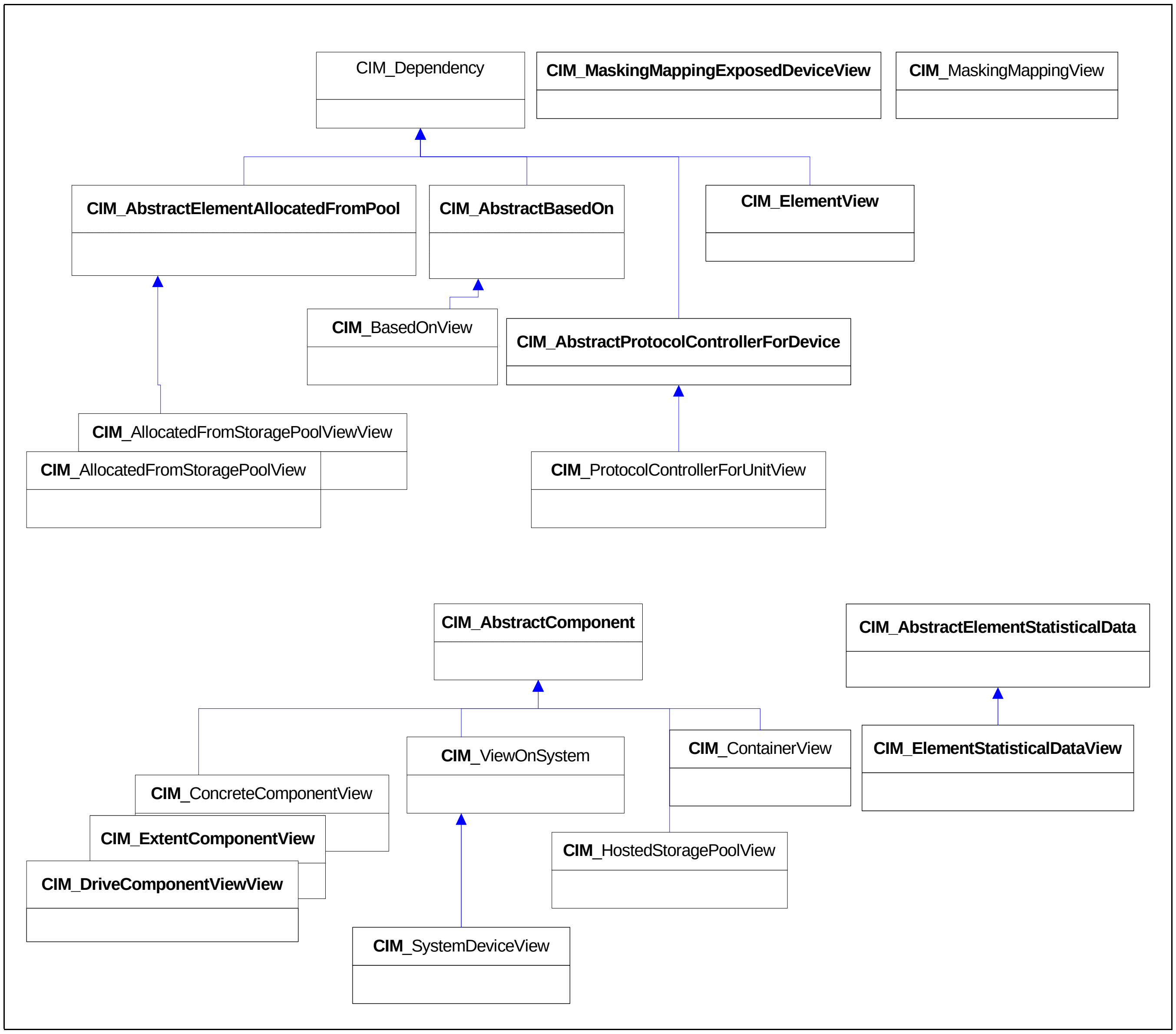
CIM_Dependency
CIM_MaskingMappingExposedDeviceView
CIM_MaskingMappingView
CIM_AbstractElementAllocatedFromPool
CIM_AbstractBasedOn
CIM_ElementView
CIM_BasedOnView
CIM_AbstractProtocolControllerForDevice
CIM_AllocatedFromStoragePoolViewView
CIM_AllocatedFromStoragePoolView
CIM_ProtocolControllerForUnitView
CIM_AbstractComponent
CIM_AbstractElementStatisticalData
CIM_ViewOnSystem
CIM_ContainerView
CIM_ElementStatisticalDataView
CIM_ConcreteComponentView
CIM_ExtentComponentView
CIM_DriveComponentViewView
CIM_HostedStoragePoolView
CIM_SystemDeviceView

Figure 26 - Class diagram for view associations

There are five types of associations defined in Figure 26: "Class diagram for view associations":

- CIM_ElementView - This associates a managed element view to the base instances from which it is derived
- Bridge Associations - These are associations that bridge between the view of the model and related instances in the normalized model. CIM_AllocatedFromStoragePoolView, CIM_BasedOnView, CIM_ProtocolControllerForUnitView, CIM_ConcreteComponentView, CIM_ExtentComponentView, CIM_ContainerView and CIM_ElementStatisticalDataView are examples of bridge associations
- Scoping Associations - These are special cases of bridge associations. They association a view instance with the scoping system in the normalized model. CIM_SystemDeviceView and CIM_HostedStoragePoolView are examples of scoping associations.

- Short cut Associations - These are go between two (or more) classes in the normalized model. The normalized model is a more complicated network of associations. The short cut associations reduce the association traversals needed to get between the normalized class instances. The CIM_MaskingMappingExposedDeviceView and the MaskingMappingView are examples of short cut associations.
- View Associations - These are associations between two views. These allow discovery by traversing view classes. CIM_AllocatedFromStoragePoolViewView and CIM_DriveComponentViewView are examples of view associations.

6.1.4 Implementation

6.1.4.1 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 27: "Block Storage View Class Capabilities".

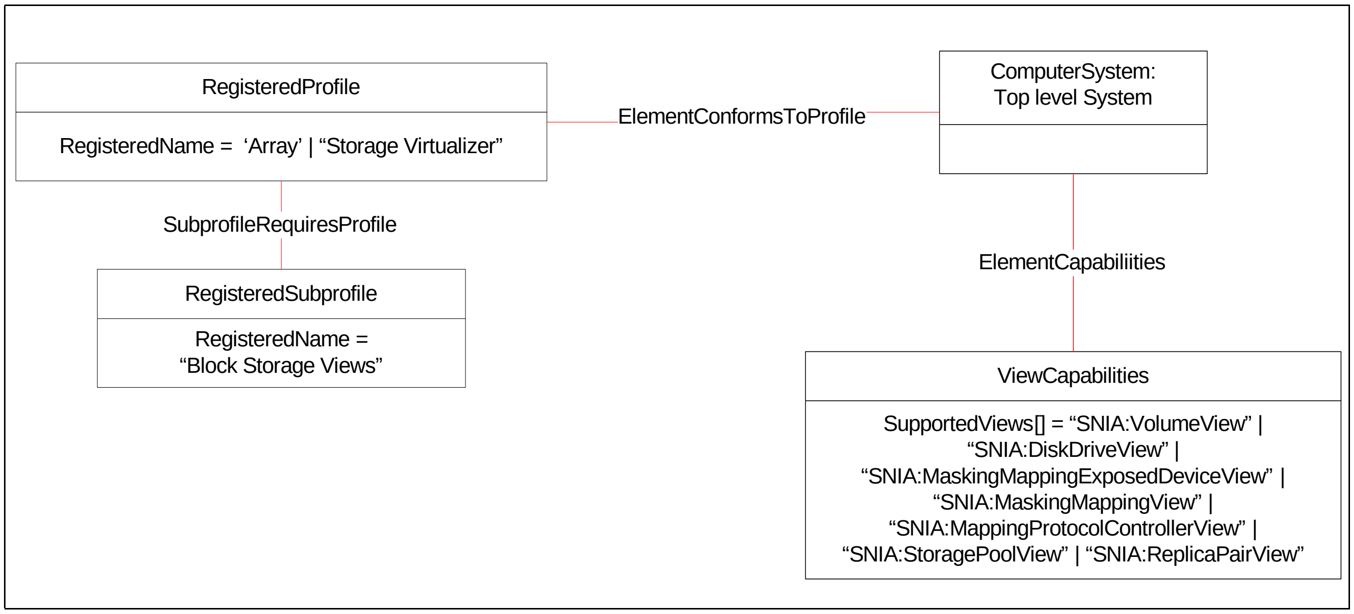
RegisteredProfile
RegisteredName = 'Array' | "Storage Virtualizer"
ElementConformsToProfile
ComputerSystem:
Top level System
SubprofileRequiresProfile
RegisteredSubprofile
RegisteredName =
"Block Storage Views"
ElementCapabiliities
ViewCapabilities
SupportedViews[] = "SNIA:VolumeView" |
"SNIA:DiskDriveView" |
"SNIA:MaskingMappingExposedDeviceView" |
"SNIA:MaskingMappingView" |
"SNIA:MappingProtocolControllerView" |
"SNIA:StoragePoolView" | "SNIA:ReplicaPairView"

Figure 27 - 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 CIM_ViewCapabilities instance. There shall be one instance of the CIM_ViewCapabilities class if the profile supports the Block Storage Views Subprofile. The CIM_ViewCapabilities instance shall have an array of strings that identify the view classes that are supported. For example, if the SupportedViews array includes the "SNIA:VolumeView" string, then the VolumeView class shall be supported.

6.1.4.2 Storage Volume Views

6.1.4.2.1 CIM_VolumeView and related associations

Figure 28: "VolumeView and related associations" illustrates the CIM_VolumeView and related associations.

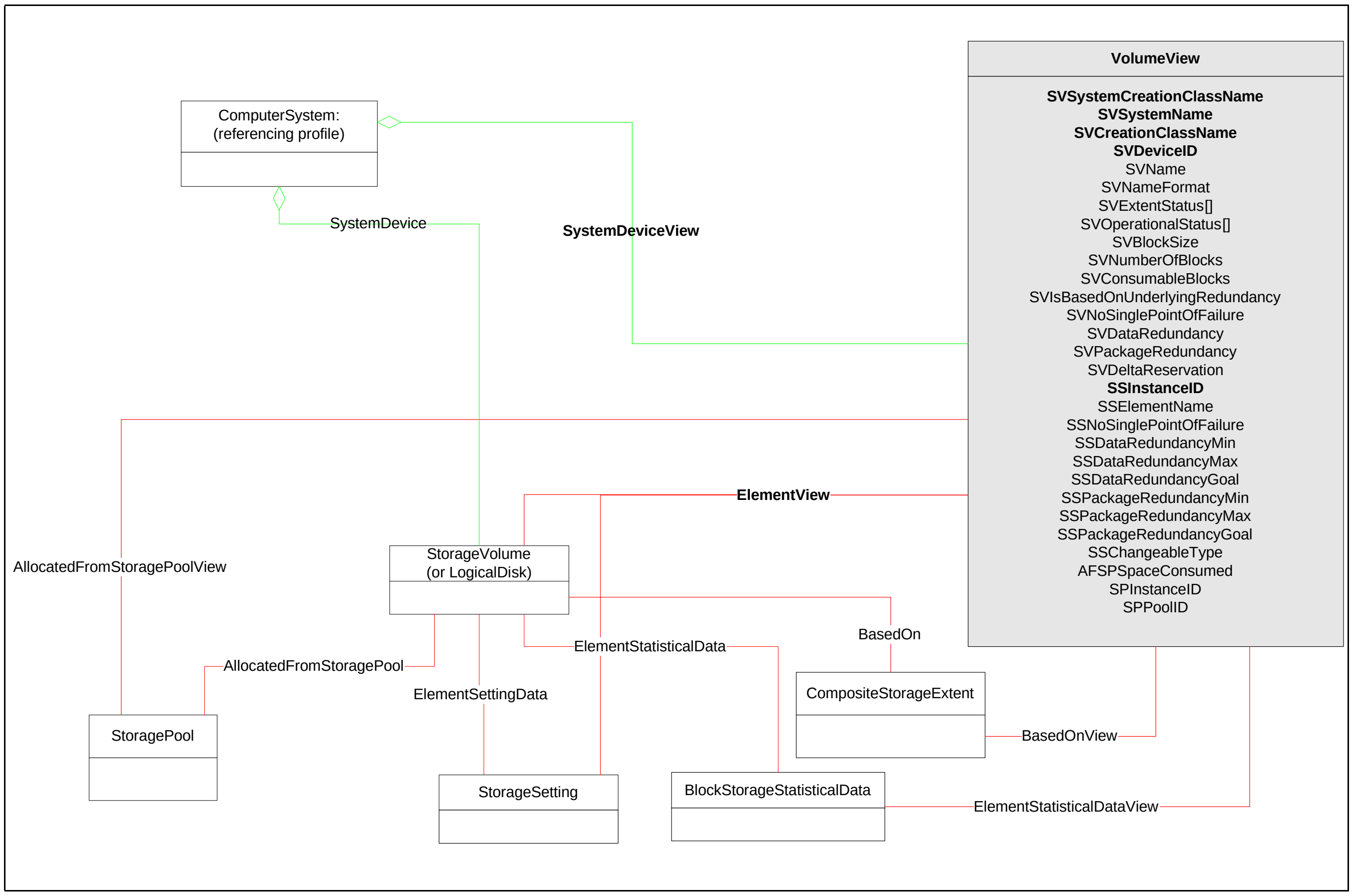

Figure 28 - VolumeView and related associations

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

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

The keys for the VolumeView are the StorageVolume and StoragePool keys from the base StorageVolume and StoragePool instances. There will be one instance of 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 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.

6.1.4.2.2 Mandatory, Conditional and Optional Properties of VolumeView

Properties that are mandatory in the mandatory base classes are mandatory in the VolumeView class. Properties that are Conditional in the base classes are conditional in the VolumeView class. Properties that are mandatory in optional (base) classes (CompositeExtent) are "conditional" in the VolumeView. If an optional base class is not supported by the referencing profile (e.g., 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 VolumeView.

EXPERIMENTAL

The CIM_VolumeView includes the experimental SVCanDelete and SVIsComposite properties which are picked up from the CanDelete and SVIsComposite properties of CIM_StorageVolume.

EXPERIMENTAL

6.1.4.2.3 Associations on VolumeView

The VolumeView is "read only." Access to base class instances on which the view is based can be updated (e.g., StorageVolume and StorageSetting) from the CIM_VolumeView instance by accessing the base instances via the ElementView association.

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

6.1.4.2.3.1 SystemDeviceView

From the scoping CIM_ComputerSystem instance a client will be able to find the VolumeView instances associated to the ComputerSystem via the SystemDeviceView association. 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 the SystemDevice association.

6.1.4.2.3.2 AllocatedFromStoragePoolView

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

6.1.4.2.3.3 BasedOnView

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

Similarly, from a “top level” StorageExtent instance, a client can find the VolumeView instance(s) that are based on that StorageExtent.

6.1.4.2.3.4 ElementStatisticalDataView

From the VolumeView instance, the client can find the BlockStorageStatisticalData instance for the StorageVolume or LogicalDisk of the VolumeView by following the ElementStatisticalDataView association.

6.1.4.3 Disk Drive Views

Figure 29: "DiskDriveView and related associations" illustrates the DiskDriveView class and related associations.

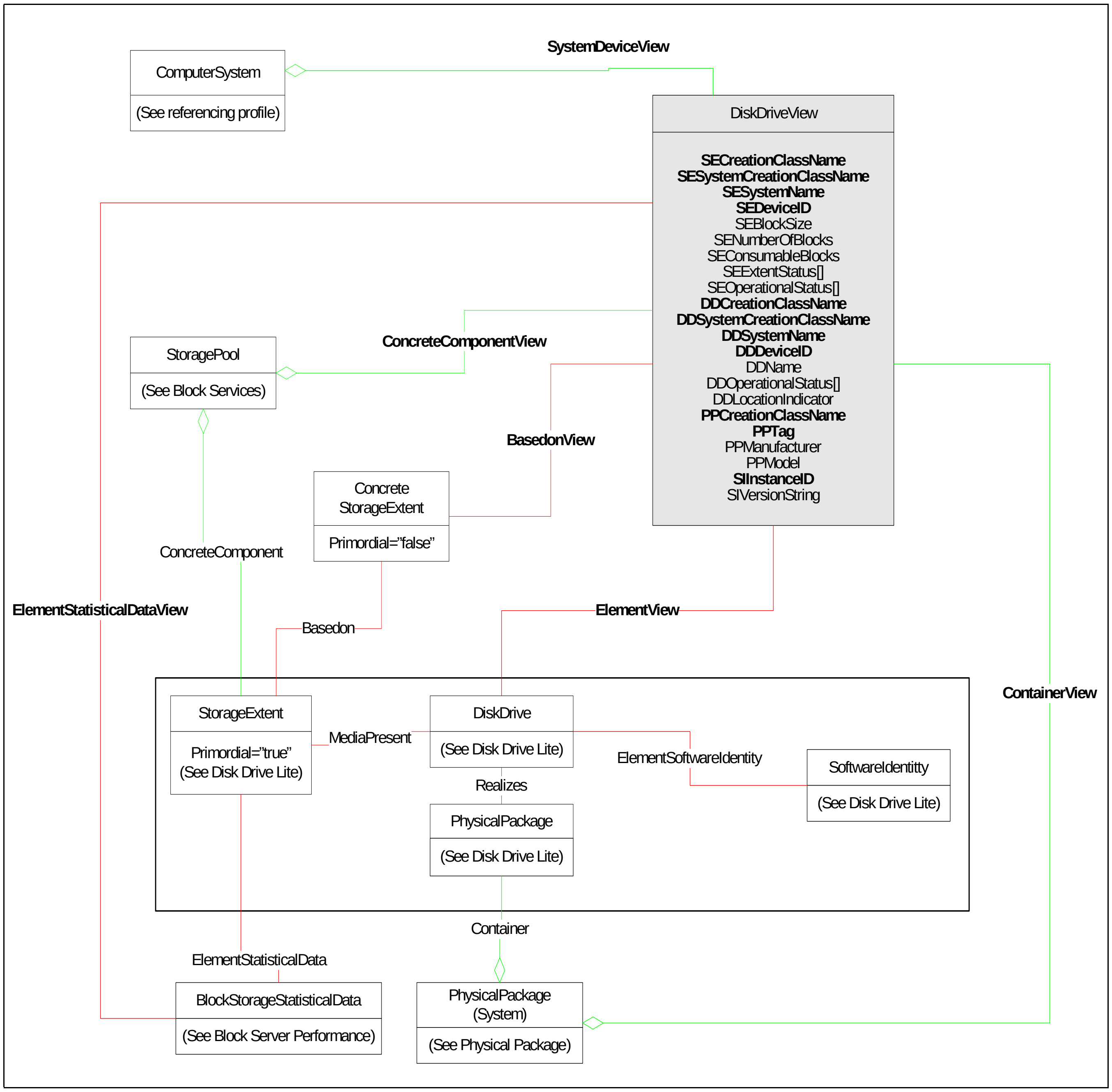

Figure 29 - DiskDriveView and related associations

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

- StorageExtent
- DiskDrive
- PhysicalPackage
- SoftwareIdentity (conditional)

- LogicalPort (Optional)

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

6.1.4.3.1 Mandatory, Conditional and Optional Properties of 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 DiskDriveView class. Properties that are conditional in a base class are conditional in the DiskDriveView class. Properties that are mandatory in optional (base) classes (BlockStorageStatisticalData and SoftwareIdentity) are also "conditional" in the DiskDriveView. If an optional base class is not supported by the referencing profile (e.g., 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 DiskDriveView.

EXPERIMENTAL

The DiskDriveView includes an optional array property (LPPortType) to hold the PortTypes for the target ports for the DiskDrive.

In addition to the CIM_DiskDriveView, this profile also defines a SNIA_DiskDriveView subclass of CIM_DiskDriveView. The SNIA_DiskDriveView is derived from SNIA_DiskDrive, rather than CIM_DiskDrive. The SNIA_DiskDriveView includes a DDDiskType, DDFormFactor and DDEncryption properties which are picked up from the corresponding properties of SNIA_DiskDrive.

EXPERIMENTAL

6.1.4.3.2 Associations on DiskDriveView

The DiskDriveView is "read only." In order to support update of information in the DiskDriveView instance, it would be necessary to update the class instances on which it is based. An association ElementView is provided to the DiskDrive instance.

> **Note:** The ElementView association is only provided to base instances that can be modified.

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

6.1.4.3.2.1 ConcreteComponentView (mandatory if the DiskDriveView is implemented)

From a primordial StoragePool instance a client will be able to find the DiskDriveViews associated to the StoragePool via the 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 ConcreteComponent association.

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

6.1.4.3.2.2 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 DiskDriveViews associated to the PhysicalPackage instance via the ContainerView. This will return the DiskDriveView instances that correspond to the Disk Drive PhysicalPackage that would be found via association traversal from the system PhysicalPackage to the Disk Drive PhysicalPackage via Container association.

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

6.1.4.3.2.3 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 DiskDriveViews associated to the StorageExtent instance via the 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 StorageExtent to the Disk Drive StorageExtent via BasedOn association.

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

6.1.4.3.2.4 SystemDeviceView (mandatory if the DiskDriveView is implemented)

From the ComputerSystem of the referencing profile a client will be able to find the DiskDriveViews associated to the ComputerSystem via the SystemDeviceView. This will return the DiskDriveViews that correspond to the DiskDrive instances that would be found via association traversal from the ComputerSystem to the DiskDrive instances via SystemDevice.

Similarly, if the client has a DiskDriveView instance, the client can find the owning ComputerSystem by following the SystemDeviceView association from the DiskDriveView instance to the ComputerSystem instance for the ComputerSystem that scopes the DiskDrive instances.

6.1.4.3.2.5 ElementStatisticalDataView

From the DiskDriveView instance, the client can find the BlockStorageStatisticalData instance for the Disk Drive StorageExtent of the DiskDriveView by following the ElementStatisticalDataView association.

6.1.4.4 Masking and Mapping Views

6.1.4.4.1 The MaskingMappingExposedDeviceView Association

Figure 30: "MaskingMappingExposedDeviceView Association" illustrates the MaskingMappingExposedDeviceView Association.

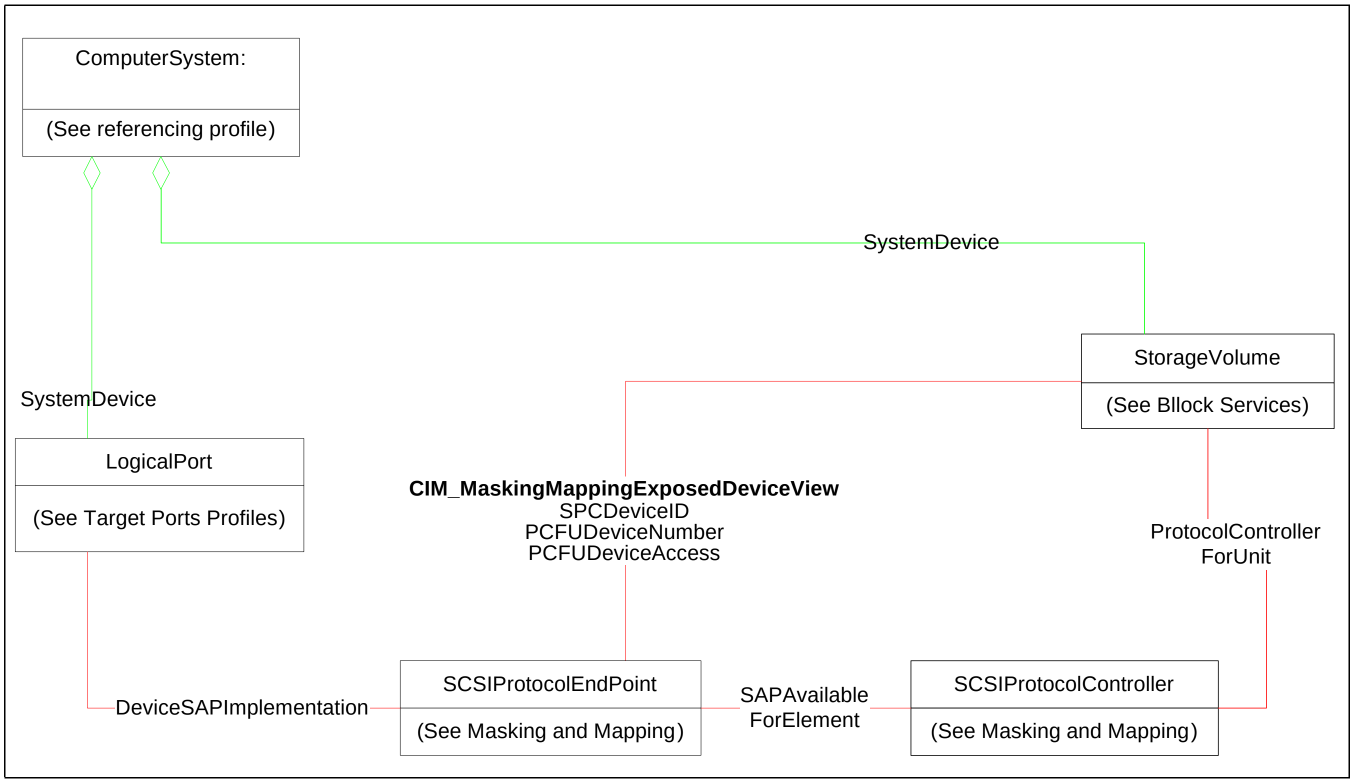

Figure 30 - MaskingMappingExposedDeviceView Association

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

- SCSIProtocolController
- SAPAvailableForElement
- ProtocolControllerForUnit

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

6.1.4.4.1.1 Mandatory, Conditional and Optional Properties of CIM_MaskingMappingExposedDeviceView Association

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

The MaskingMappingExposedDeviceView is "read only." It would be used to do association traversal from StorageVolumes to SCSIProtocolEndpoints that expose the Volumes.

6.1.4.4.2 MaskingMappingView Association

Figure 31: "MaskingMappingView Association" illustrates the MaskingMappingView Association.

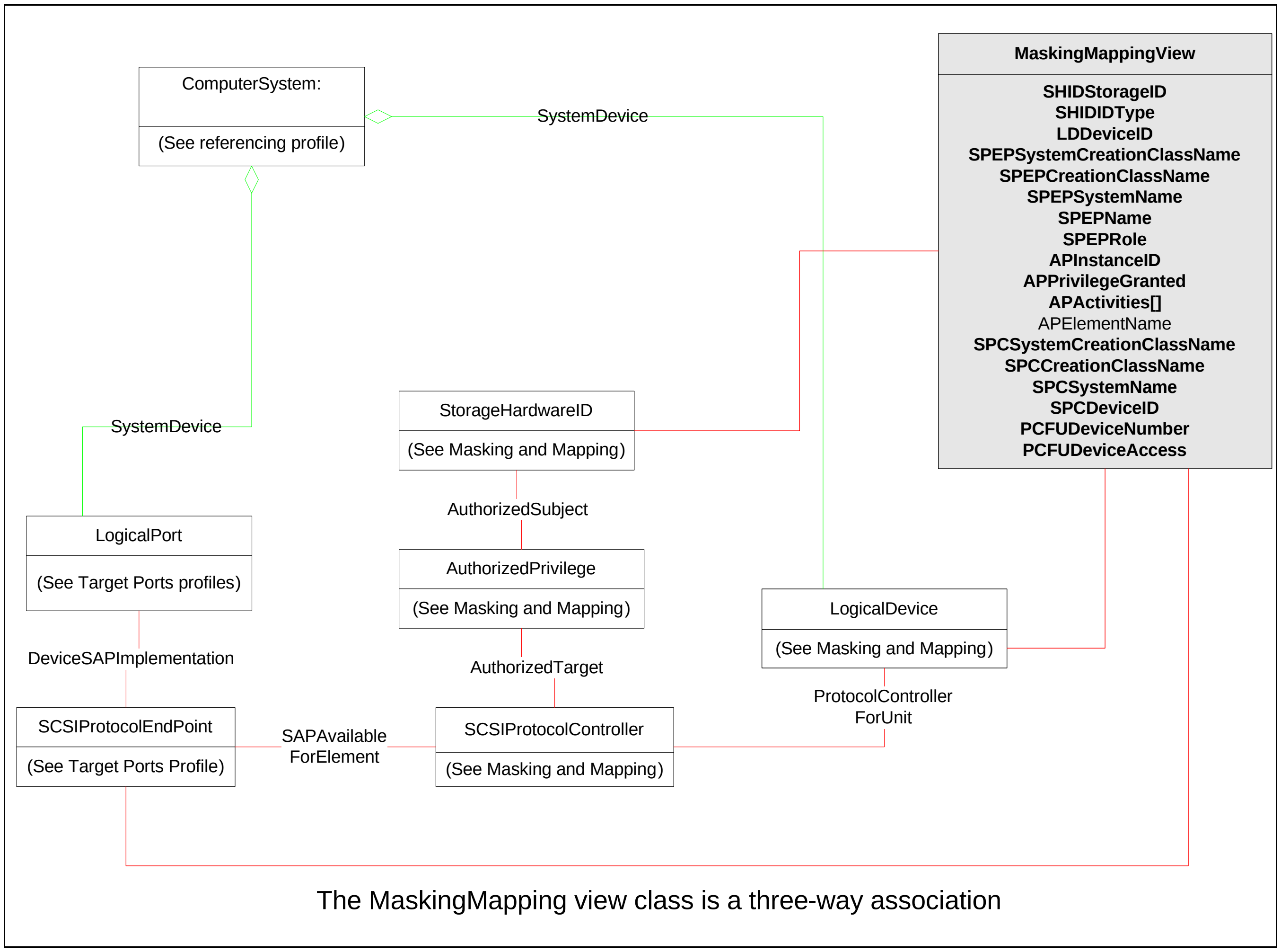

Figure 31 - MaskingMappingView Association

The MaskingMappingView 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 MaskingMappingView are the SHID reference, the SCSIProtocolEndpoint reference and the LogicalDevice reference. There will be one instance of MaskingMappingView for each unique combination of Storage Hardware ID (e.g., host), LogicalDeivce (e.g., StorageVolume) and SCSIProtocolEndpoint (e.g., LogicalPort).

6.1.4.4.2.1 Mandatory, Conditional and Optional Properties of MaskingMappingView Association

In addition to the references to StorageHardwareID, LogicalDevice and the SCSIProtocolEndpoint the MaskingMappingView 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 MaskingMappingView. This allows a client to enumerate the MaskingMappingView association and get the identifiers for the endpoints in the association.

The MaskingMappingView is "read only." It would be used to do associate the StorageHardwareIDs, StorageVolumes to SCSIProtocolEndpoints.

EXPERIMENTAL

6.1.4.4.3 MappingProtocolControllerView

Figure 32 illustrates the elements involved in supporting the MappingProtocolControllerView.

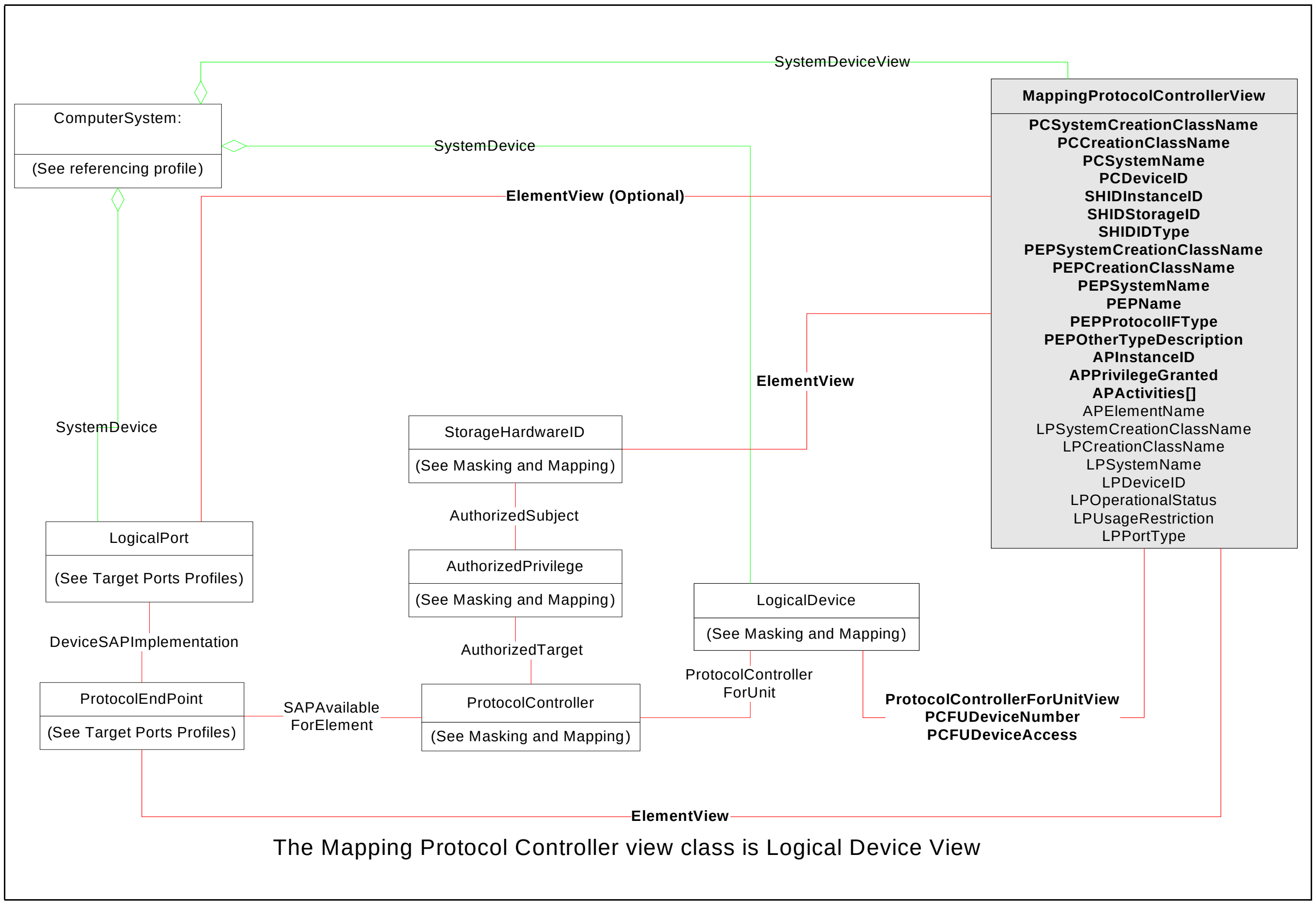

Figure 32 - The MappingProtocolControllerView

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

- LogicalPort
- ProtocolEndpoint

- ProtocolController
- AuthorizedPrivilege
- StorageHardwareID

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

6.1.4.4.4 Mandatory, Conditional and Optional Properties of 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 MappingProtocolControllerView class. Properties that are conditional in a base class are conditional in the MappingProtocolControllerView class.

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

6.1.4.4.5 Associations on MappingProtocolControllerView

The CIM_MappingProtocolControllerView is "read only." In order to support update of information in the MappingProtocolControllerView instance, it would be necessary to update the class instances on which it is based. An association ElementView is provided to the CIM_StorageHardwareID, CIM_LogicalPort and CIM_ProtocolEndpoint instances.

Note: The ElementView association is only provided to base instances that can be modified.

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

6.1.4.4.5.1 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 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.

6.1.4.4.5.2 SystemDeviceView (mandatory if the MappingProtocolControllerView is implemented)

From the CIM_ComputerSystem of the referencing profile a client will be able to find the MappingProtocolControllerViews associated to the ComputerSystem via the 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 MappingProtocolControllerView instance, the client can find the scoping ComputerSystem by following the SystemDeviceView association from the MappingProtocolControllerView instance to the CIM_ComputerSystem instance for the ComputerSystem that scopes the CIM_ProtocolController instances.

6.1.4.5 Storage Pool Views

6.1.4.5.1 StoragePoolView

Figure 33 illustrates the elements involved in supporting the StoragePoolView.

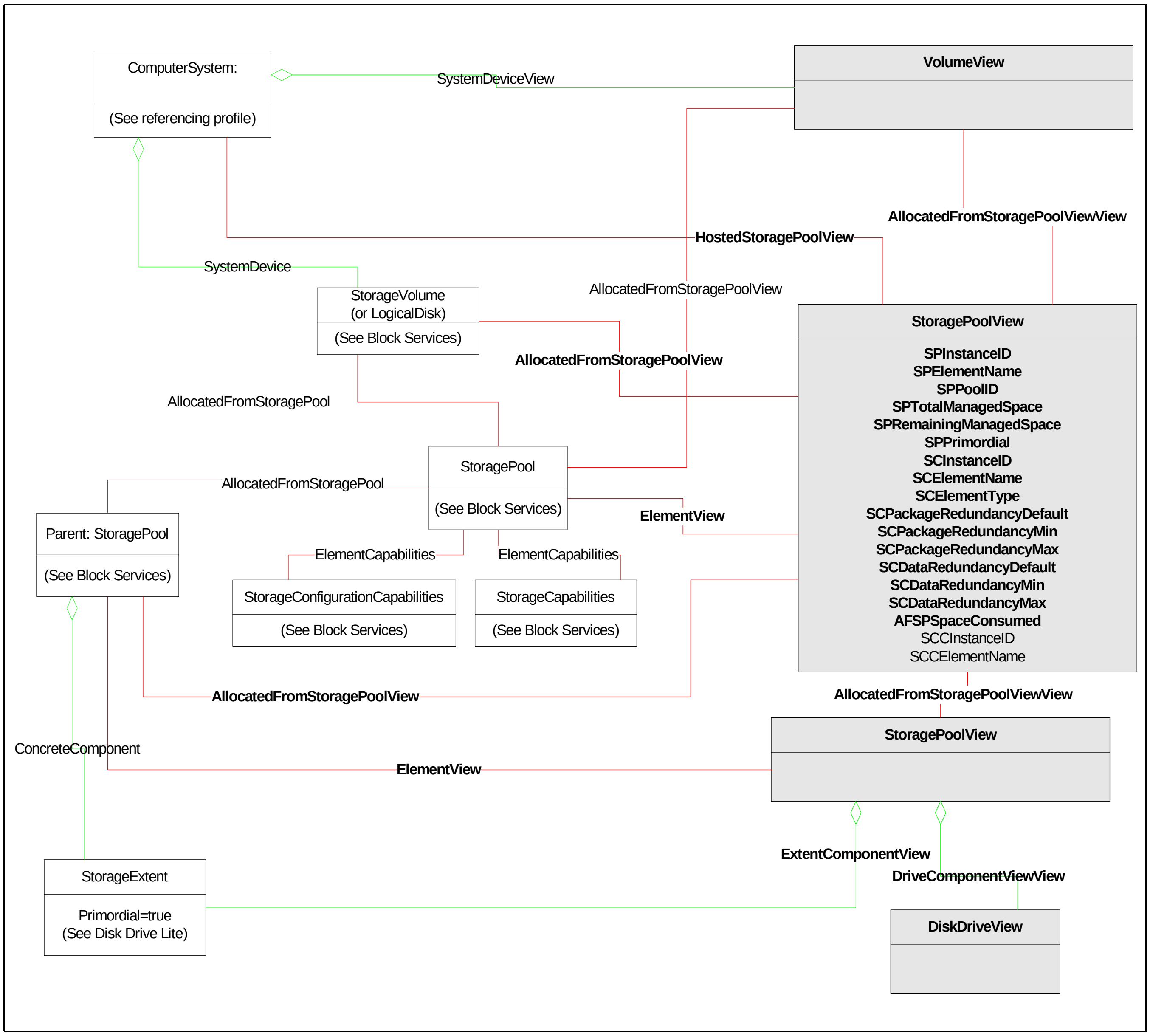

Figure 33 - The StoragePoolView

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

- StoragePool
- StorageCapabilities
- StorageConfigurationCapabilities (Optional)
- AllocatedFromStoragePool

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

6.1.4.5.2 Mandatory, Conditional and Optional Properties of 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 StoragePoolView class. Properties that are conditional in a base class are conditional in the StoragePoolView class. Properties that are mandatory in optional (base) classes (e.g., StorageConfigurationCapabilities) are "conditional" in the 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 StoragePoolView.

6.1.4.5.3 Associations on StoragePoolView

The StoragePoolView is "read only." In order to support update of information in the StoragePoolView instance, it would be necessary to update the class instances on which it is based. An association ElementView is provided to the CIM_StoragePool instance.

Note: The ElementView association is only provided to base instances that can be modified.

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

6.1.4.5.3.1 AllocatedFromStoragePoolView (StoragePoolView to StoragePool)

This association is mandatory if the StoragePoolView is implemented.

From a StoragePoolView instance, the client can find the parent StoragePool to which the pool is allocated from by following the AllocatedFromStoragePoolView association from the 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 StoragePoolViews that are allocated from the StoragePool via the 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.

6.1.4.5.3.2 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 AllocatedFromStoragePoolView association from the CIM class (StorageVolume or LogicalDisk) to the appropriate StoragePoolView instance that corresponds to the CIM_StoragePool instance the volume is allocated from.

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

6.1.4.5.3.3 AllocatedFromStoragePoolViewView (VolumeView to StoragePoolView)

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

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

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

6.1.4.5.3.4 AllocatedFromStoragePoolViewView (StoragePoolView to StoragePoolView)

This association is mandatory if the StoragePoolView is implemented.

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

Similarly, if the client has a StoragePoolView instance a client will be able to find the StoragePoolViews that are allocated from the StoragePool via the 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.

6.1.4.5.3.5 HostedStoragePoolView

This is mandatory if the StoragePoolView is implemented.

From the owning CIM_ComputerSystem a client will be able to find the StoragePoolViews associated to the ComputerSystem via the 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 StoragePoolView instance, the client can find the owning ComputerSystem by following the HostedStoragePoolView association from the StoragePoolView instance to the CIM_ComputerSystem instance for the ComputerSystem that scopes the CIM_StoragePool instances.

6.1.4.5.3.6 ExtentComponentView

This is mandatory if the StoragePoolView is implemented.

From a StoragePoolView instance, the client can find the pool component CIM_StorageExtent instances for the extents that form the pool via the ExtentComponentView. This will return the StorageExtents that correspond to the 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 StoragePoolView by following the ExtentComponentView association from the CIM_StorageExtent instance to the StoragePoolView instance for the storage pool that has the CIM_StorageExtent as a pool component.

6.1.4.5.3.7 DriveComponentViewView

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

From a StoragePoolView instance, the client will be able to find the DiskDriveViews for drives that are components of that StoragePoolView by following the 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 DriveComponentViewView association from the DiskDriveView instance to the appropriate StoragePoolView instance that corresponds to the CIM_StoragePool instance the drive is a component of.

6.1.4.6 Replication Views

6.1.4.6.1 ReplicaPairView

Figure 34 illustrates the elements involved in supporting the ReplicaPairView.

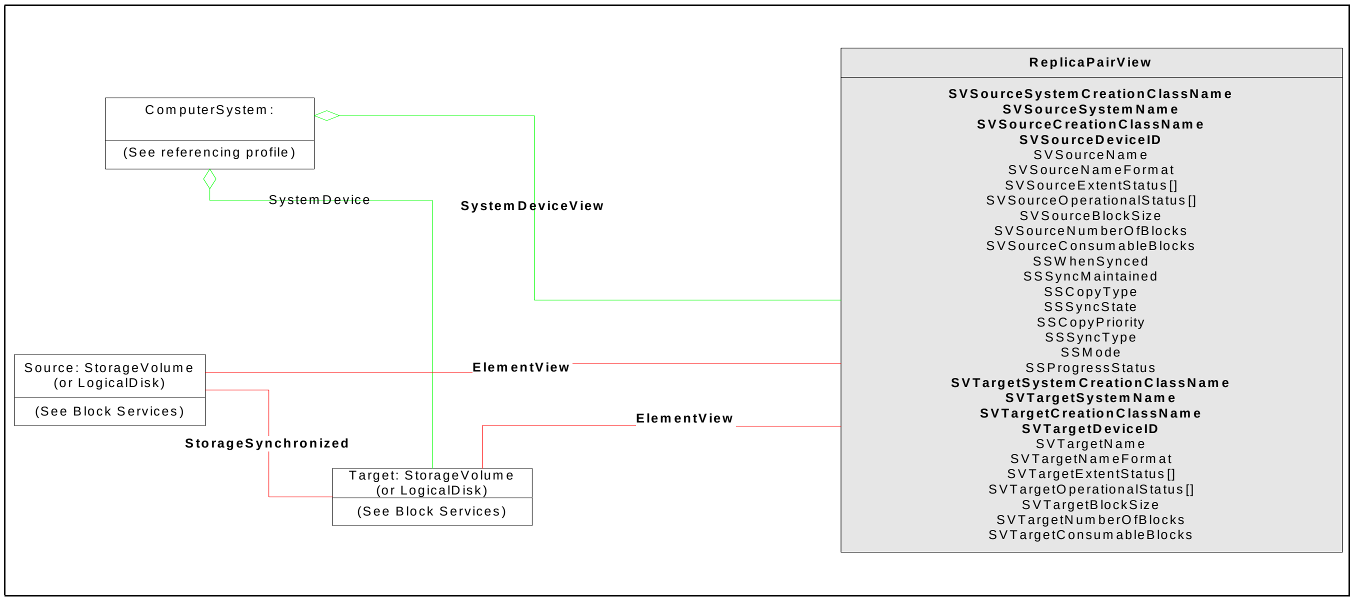

Figure 34 - The ReplicaPairView

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

6.1.4.6.2 Mandatory, Conditional and Optional Properties of 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 ReplicaPairView class. Properties that are conditional in a base class are conditional in the ReplicaPairView class.

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

6.1.4.6.3 Associations on ReplicaPairView

The ReplicaPairView is "read only." In order to support update of information in the ReplicaPairView instance, it would be necessary to update the class instances on which it is based. An association ElementView is provided to the CIM_StorageVolume instances (both source and target).

Note: The ElementView association is only provided to base instances that can be modified.

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

6.1.4.6.3.1 SystemDeviceView (ReplicaPairViews)

This is mandatory if the ReplicaPairView is implemented.

From the CIM_ComputerSystem of the referencing profile a client will be able to find the ReplicaPairViews associated to the ComputerSystem via the 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 ReplicaPairView instance, the client can find the owning ComputerSystem by following the SystemDeviceView association from the ReplicaPairView instance to the CIM_ComputerSystem instance for the ComputerSystem that scopes the CIM_StorageVolume (CIM_LogicalDisk) instances.

EXPERIMENTAL

6.2 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.

6.3 Cascading Considerations

Not defined in this standard.

6.4 Methods of the Profile

6.4.1 Extrinsic Methods of the Profile

None

6.4.2 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

View classes are modified by creating, deleting and modifying the base classes from which they are derived. The property values of 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 ElementView association.

6.5 Client Considerations and Recipes

6.5.1 Use Cases

6.5.1.1 Discovery of the Volumes on an Array

Table 84 identifies the elements of the use case to discover the volumes on an Array.

Table 84 - 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 ViewCapabilities.SupportedViews contains “SNIA:VolumeView”.
Postconditions	Administrator has all Volumes, their Settings and what Pools they are allocated from.

6.5.1.2 Discovery of the Disk Drives in a Primordial Pool

Table 85 identifies the elements of the use case to discover the Disk Drives in a Primordial Pool.

Table 85 - 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 ViewCapabilities.SupportedViews contains “SNIA:DiskDriveView”.

Table 85 - Discovery of the Disk Drives in a Primordial Pool

Use Case Element	Description
Postconditions	Administrator has all DiskDrives and related information (scoped by the Pool or System)

6.5.1.3 Discover Volumes exposed on a (Target) Port

Table 86 identifies the elements of the use case to Discover Volumes exposed on a (Target) Port.

Table 86 - 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 MaskingMappingExposedDeviceView)
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 ViewCapabilities.SupportedViews contains “SNIA:MaskingMappingExposedDeviceView”.
Postconditions	Administrator has all Volumes that depend on the port for access.

6.5.1.4 Discover (target port) redundancy for a Volume

Table 87 identifies the elements of the use case to discover (target port) redundancy for a Volume.

Table 87 - 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 MaskingMappingExposedDeviceView) 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

Table 87 - Discover (target port) redundancy for a Volume

Use Case Element	Description
Preconditions	The Array provider has implemented the Block Storage Views Profile and ViewCapabilities.SupportedViews contains “SNIA:MaskingMappingExposedDeviceView”.
Postconditions	Administrator has the ports through which the volume may be accessed.

6.5.1.5 Discover Volumes exposed to a Host Port

Table 88 identifies the elements of the use case to discover Volumes exposed to a Host Port.

Table 88 - 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 (MaskingMappingView)
Alternative Paths	None
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 ViewCapabilities.SupportedViews contains “SNIA:MappingMaskingView”.
Postconditions	Administrator has all Volumes that are mapped to the host port.

6.5.1.6 Discover the Mapping information for an array

Table 89 identifies the elements of the use case to discover the Mapping information for an array.

Table 89 - 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.

Table 89 - Discover Mapping information for an array

Use Case Element	Description
Assumptions	None
Preconditions	The Array provider has implemented the Block Storage Views Profile and ViewCapabilities.SupportedViews contains “SNIA:MappingProtocolControllerView”.
Postconditions	Administrator has all the Masking and Mapping information.

6.5.1.7 Discover the Pool topology for an array

Table 90 identifies the elements of the use case to discover the Pool topology for an array.

Table 90 - 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 ViewCapabilities.SupportedViews contains “SNIA:StoragePoolView”.
Postconditions	Administrator has all the Pools and their capabilities information.

6.5.1.8 Discover the Replica Pairs for an array

Table 91 identifies the elements of the use case to discover the Replica Pairs for an array.

Table 91 - 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

Table 91 - Discover the Replica Pairs for an array

Use Case Element	Description
Preconditions	The Array provider has implemented the Block Storage Views Profile and ViewCapabilities.SupportedViews contains “SNIA:ReplicaPairView”.
Postconditions	Administrator has all the Replicas that are defined in the Array.

6.5.2 Recipes

Not supported in this version of the standard.

6.6 CIM Elements

Table 92 describes the CIM elements for Block Storage Views.

Table 92 - CIM Elements for Block Storage Views

Element Name	Requirement	Description
6.6.1 CIM_AllocatedFromStoragePoolView (StoragePoolView to StoragePool)	Conditional	Conditional requirement: Required if the array property CIM_ViewCapabilities.SupportedViews contains the string "SNIA:StoragePoolView" (and the Block Service Package is implemented). This associates a CIM_StoragePoolView instance to a CIM_StoragePool instance. This is required if the CIM_StoragePoolView is implemented.
6.6.2 CIM_AllocatedFromStoragePoolView (Volume to StoragePoolView)	Conditional	Conditional requirement: Required if the array property CIM_ViewCapabilities.SupportedViews contains the string "SNIA:StoragePoolView" (and the Block Service Package is implemented). This associates a CIM_StorageVolume (or CIM_LogicalDisk) instance to a CIM_StoragePoolView. This is required if the CIM_StoragePoolView is implemented.
6.6.3 CIM_AllocatedFromStoragePoolView (VolumeView to StoragePool)	Conditional	Conditional requirement: Required if the array property CIM_ViewCapabilities.SupportedViews contains the string "SNIA:VolumeView" (and the Block Service Package is implemented). This associates a CIM_VolumeView instance to a CIM_StoragePool. This is required if the CIM_VolumeView is implemented.

Table 92 - CIM Elements for Block Storage Views

Element Name	Requirement	Description
6.6.4 CIM_AllocatedFromStoragePoolViewView (PoolView to PoolView)	Conditional	Experimental. Conditional requirement: Required if the array property CIM_ViewCapabilities.SupportedViews contains the string "SNIA:StoragePoolView" (and the Block Service Package is implemented). This associates a CIM_StoragePoolView instance to its parent CIM_StoragePoolView instance that it is allocated from.
6.6.5 CIM_AllocatedFromStoragePoolViewView (VolumeView to PoolView)	Conditional	Experimental. Conditional requirement: Required if the array property CIM_ViewCapabilities.SupportedViews contains the strings "SNIA:StoragePoolView" and "SNIA:VolumeView" (and the Block Service Package is implemented). This associates a CIM_VolumeView instance to a CIM_StoragePoolView instance that volume is allocated from.
6.6.6 CIM_BasedOnView (ExtentOnDriveExtent)	Conditional	Conditional requirement: Required if the array property CIM_ViewCapabilities.SupportedViews contains the string "SNIA:DiskDriveView" and Extent Composition is implemented. This associates a concrete CIM_StorageExtent instance to a CIM_DiskDriveView instance. This is required if the CIM_DiskDriveView and ExtentComposition are implemented.
6.6.7 CIM_BasedOnView (VolumeOnExtent)	Conditional	Conditional requirement: Required if the array property CIM_ViewCapabilities.SupportedViews contains the string "SNIA:VolumeView" and Extent Composition is implemented. This associates a CIM_VolumeView instance to a base CIM_StorageExtent instance on which the volume is based. This is required if the CIM_VolumeView and ExtentComposition are implemented.

Table 92 - CIM Elements for Block Storage Views

Element Name	Requirement	Description
6.6.8 CIM_ConcreteComponentView	Conditional	Conditional requirement: Required if the array property CIM_ViewCapabilities.SupportedViews contains the string "SNIA:DiskDriveView" (and the Disk Drive Lite Profile is implemented). The CIM_ConcreteComponentView associates the CIM_DiskDriveView instance to the primordial StoragePool to which the disk drive StorageExtent is assigned. This is required if the CIM_DiskDriveView is implemented.
6.6.9 CIM_ContainerView	Conditional	Conditional requirement: Required if the array property CIM_ViewCapabilities.SupportedViews contains the string "SNIA:DiskDriveView" (and the Disk Drive Lite Profile is implemented). The CIM_ContainerView associates the CIM_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 CIM_DiskDriveView is implemented.
6.6.10 CIM_DiskDriveView	Conditional	Conditional requirement: Required if the array property CIM_ViewCapabilities.SupportedViews contains the string "SNIA:DiskDriveView" (and the Disk Drive Lite Profile is implemented). The CIM_DiskDriveView instance represents a Disk Drive and its associated information. This is required if CIM_ViewCapabilities.SupportedViews includes "SNIA:DiskDriveView".
6.6.11 CIM_DriveComponentViewView	Conditional	Experimental. Conditional requirement: Required if the array property CIM_ViewCapabilities.SupportedViews contains the string "SNIA:StoragePoolView" and "SNIA:DiskDriveView" (and the Disk Drive Lite Profile is implemented). This associates a CIM_StoragePoolView instance to a CIM_DiskDriveView instance that is a component of the StoragePool.
6.6.12 CIM_ElementCapabilities (View Capabilities)	Mandatory	Associates the top level ComputerSystem to the CIM_ViewCapabilities supported by the implementation.

Table 92 - CIM Elements for Block Storage Views

Element Name	Requirement	Description
6.6.13 CIM_ElementStatisticalDataView (DiskDriveView)	Conditional	Conditional requirement: Required if the array property CIM_ViewCapabilities.SupportedViews contains the string "SNIA:DiskDriveView", CIM_BlockStatisticsCapabilities.ElementTypesSupported contains "10" and Block Server Performance is implemented. This associates a CIM_DiskDriveView instance to the CIM_BlockStorageStatisticalData instance for the Disk Drive. This is required if the CIM_DiskDriveView and the Block Server Performance Subprofile are implemented.
6.6.14 CIM_ElementStatisticalDataView (VolumeView)	Conditional	Conditional requirement: Required if the array property CIM_ViewCapabilities.SupportedViews contains the string "SNIA:VolumeView", CIM_BlockStatisticsCapabilities.ElementTypesSupported contains "8" and Block Server Performance is implemented. This associates a CIM_VolumeView instance to the CIM_BlockStorageStatisticalData instance for the StorageVolume (or LogicalDisk). This is required if the CIM_VolumeView and the Block Server Performance Subprofile are implemented.
6.6.15 CIM_ElementView (DiskDrive)	Conditional	Conditional requirement: Required if the array property CIM_ViewCapabilities.SupportedViews contains the string "SNIA:DiskDriveView" (and the Disk Drive Lite Profile is implemented). This associates a CIM_DiskDriveView instance to a base CIM_DiskDrive instance that can be modified. This is required if the CIM_DiskDriveView is implemented.
6.6.16 CIM_ElementView (StorageSetting)	Conditional	Conditional requirement: Required if the array property CIM_ViewCapabilities.SupportedViews contains the string "SNIA:VolumeView" (and the Block Service Package is implemented). This associates a CIM_VolumeView class instance to a base CIM_StorageSetting class instance that can be modified. This is required if the CIM_VolumeView is implemented.

Table 92 - CIM Elements for Block Storage Views

Element Name	Requirement	Description
6.6.17 CIM_ElementView (Volume)	Conditional	Conditional requirement: Required if the array property CIM_ViewCapabilities.SupportedViews contains the string "SNIA:VolumeView" (and the Block Service Package is implemented). This associates a CIM_VolumeView instance to a base CIM_StorageVolume (or CIM_LogicalDisk) instance that can be modified. This is required if the CIM_VolumeView is implemented.
6.6.18 CIM_ExtentComponentView	Conditional	Experimental. Conditional requirement: Required if the array property CIM_ViewCapabilities.SupportedViews contains the string "SNIA:StoragePoolView" (and the Block Service Package is implemented). This associates a CIM_StoragePoolView instance to a CIM_StorageExtent instance that is a component of the StoragePool.
6.6.19 CIM_HostedStoragePoolView	Conditional	Conditional requirement: Required if the array property CIM_ViewCapabilities.SupportedViews contains the string "SNIA:StoragePoolView" (and the Block Service Package is implemented). This associates a CIM_StoragePoolView instance to the CIM_ComputerSystem instance that hosts the underlying StoragePool.
6.6.20 CIM_MappingProtocolControllerView	Conditional	Experimental. Conditional requirement: Required if the array property CIM_ViewCapabilities.SupportedViews contains the string "SNIA:MappingProtocolControllerView" (and the Masking and Mapping Profile is implemented). The CIM_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 CIM_ViewCapabilities.SupportedViews includes "SNIA:MappingProtocolControllerView".

Table 92 - CIM Elements for Block Storage Views

Element Name	Requirement	Description
6.6.21 CIM_MaskingMappingExposedDeviceView	Conditional	Conditional requirement: Required if the array property CIM_ViewCapabilities.SupportedViews contains the string "SNIA:MaskingMappingExposedDeviceView" (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 CIM_ViewCapabilities.SupportedViews includes "SNIA:MaskingMappingExposedDeviceView".
6.6.22 CIM_MaskingMappingView	Conditional	Conditional requirement: Required if the array property CIM_ViewCapabilities.SupportedViews contains the string "SNIA:MaskingMappingView" (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 CIM_ViewCapabilities.SupportedViews contains "SNIA:MaskingMappingView".
6.6.23 CIM_ProtocolControllerForUnitView	Conditional	Experimental. Conditional requirement: Required if the array property CIM_ViewCapabilities.SupportedViews contains the string "SNIA:MappingProtocolControllerView" (and the Masking and Mapping Profile is implemented). Associates an instance of MappingProtocolControllerView to a LogicalDevice.
6.6.24 CIM_ReplicaPairView	Conditional	Experimental. Conditional requirement: Required if the array property CIM_ViewCapabilities.SupportedViews contains the string "SNIA:ReplicaPairlView" (and the Copy Services Profile is implemented). A view that combines a source and target volume and the StorageSynchronized between them.

Table 92 - CIM Elements for Block Storage Views

Element Name	Requirement	Description
6.6.25 CIM_StoragePoolView	Conditional	Experimental. Conditional requirement: Required if the array property CIM_ViewCapabilities.SupportedViews contains the string "SNIA: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.
6.6.26 CIM_SystemDeviceView (DiskDriveViews)	Conditional	Conditional requirement: Required if the array property CIM_ViewCapabilities.SupportedViews contains the string "SNIA:DiskDriveView" (and the Disk Drive Lite Profile is implemented). This association links CIM_DiskDriveView instances to the scoping system. This is required if the CIM_DiskDriveView is implemented.
6.6.27 CIM_SystemDeviceView (MappingProtocolControllerViews)	Conditional	Conditional requirement: Required if the array property CIM_ViewCapabilities.SupportedViews contains the string "SNIA:MappingProtocolControllerView" (and the Masking and Mapping Profile is implemented). This association links CIM_MappingProtocolControllerView instances to the scoping system. This is required if the CIM_MappingProtocolControllerView is implemented.
6.6.28 CIM_SystemDeviceView (ReplicaPairViews)	Conditional	Conditional requirement: Required if the array property CIM_ViewCapabilities.SupportedViews contains the string "SNIA:ReplicaPairlView" (and the Copy Services Profile is implemented). This association links CIM_ReplicaPairView instances to the scoping system. This is required if the CIM_ReplicaPairView is implemented.

Table 92 - CIM Elements for Block Storage Views

Element Name	Requirement	Description
6.6.29 CIM_SystemDeviceView (VolumeViews)	Conditional	Conditional requirement: Required if the array property CIM_ViewCapabilities.SupportedViews contains the string "SNIA:VolumeView" (and the Block Service Package is implemented). This association links CIM_VolumeView instances to the scoping system. This is required if the CIM_VolumeView is implemented.
6.6.30 CIM_ViewCapabilities	Mandatory	The CIM_ViewCapabilities identifies the capabilities of the implementation of view classes.
6.6.31 CIM_VolumeView	Conditional	Conditional requirement: Required if the array property CIM_ViewCapabilities.SupportedViews contains the string "SNIA:VolumeView" (and the Block Service Package is implemented). The CIM_VolumeView represents the storage (LogicalDisks or StorageVolumes) of a block storage profile. This is required if the CIM_ViewCapabilities.SupportedViews includes "SNIA:VolumeView".
6.6.32 SNIA_DiskDriveView	Optional	Experimental. The SNIA_DiskDriveView instance represents a Disk Drive and its associated information.

6.6.1 CIM_AllocatedFromStoragePoolView (StoragePoolView to StoragePool)

The CIM_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 CIM_AllocatedFromStoragePoolView is subclassed from CIM_AbstractElementAllocatedFromPool.

Created By: External

Modified By: External

Deleted By: External

Requirement: Required if the array property CIM_ViewCapabilities.SupportedViews contains the string "SNIA:StoragePoolView" (and the Block Service Package is implemented).

Table 93 describes class CIM_AllocatedFromStoragePoolView (StoragePoolView to StoragePool).

Table 93 - SMI Referenced Properties/Methods for CIM_AllocatedFromStoragePoolView (StoragePoolView to StoragePool)

Properties	Flags	Requirement	Description & 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.

6.6.2 CIM_AllocatedFromStoragePoolView (Volume to StoragePoolView)

The CIM_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 CIM_AllocatedFromStoragePoolView is subclassed from CIM_AbstractElementAllocatedFromPool.

Created By: External

Modified By: External

Deleted By: External

Requirement: Required if the array property CIM_ViewCapabilities.SupportedViews contains the string "SNIA:StoragePoolView" (and the Block Service Package is implemented).

Table 94 describes class CIM_AllocatedFromStoragePoolView (Volume to StoragePoolView).

Table 94 - SMI Referenced Properties/Methods for CIM_AllocatedFromStoragePoolView (Volume to StoragePoolView)

Properties	Flags	Requirement	Description & 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.

Table 94 - SMI Referenced Properties/Methods for CIM_AllocatedFromStoragePoolView (Volume to StoragePoolView)

Properties	Flags	Requirement	Description & Notes
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.

6.6.3 CIM_AllocatedFromStoragePoolView (VolumeView to StoragePool)

The CIM_AllocatedFromStoragePoolView instance is a view that is derived from the CIM_AllocatedFromStoragePool association between the StorageVolume or LogicalDisk (of the CIM_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 CIM_AllocatedFromStoragePoolView is subclassed from CIM_AbstractElementAllocatedFromPool.

Created By: External

Modified By: External

Deleted By: External

Requirement: Required if the array property CIM_ViewCapabilities.SupportedViews contains the string "SNIA:VolumeView" (and the Block Service Package is implemented).

Table 95 describes class CIM_AllocatedFromStoragePoolView (VolumeView to StoragePool).

Table 95 - SMI Referenced Properties/Methods for CIM_AllocatedFromStoragePoolView (VolumeView to StoragePool)

Properties	Flags	Requirement	Description & 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 CIM_VolumeView is allocated.
Dependent		Mandatory	The CIM_VolumeView instance that is allocated from the StoragePool. There is only one VolumeView instance for the combined StorageVolume (or LogicalDisk) - StoragePool pair.

6.6.4 CIM_AllocatedFromStoragePoolViewView (PoolView to PoolView)

Experimental. This CIM_AllocatedFromStoragePoolViewView is an association between a CIM_StoragePoolView instances and the CIM_StoragePoolView instance that they are allocated from. . The CIM_AllocatedFromStoragePoolViewView is subclassed from CIM_AbstractElementAllocatedFromPool.

Requirement: Required if the array property CIM_ViewCapabilities.SupportedViews contains the string "SNIA:StoragePoolView" (and the Block Service Package is implemented).

Table 96 describes class CIM_AllocatedFromStoragePoolViewView (PoolView to PoolView).

Table 96 - SMI Referenced Properties/Methods for CIM_AllocatedFromStoragePoolViewView (PoolView to PoolView)

Properties	Flags	Requirement	Description & 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.

6.6.5 CIM_AllocatedFromStoragePoolViewView (VolumeView to PoolView)

Experimental. This CIM_AllocatedFromStoragePoolViewView is an association between a CIM_VolumeView instances and the CIM_StoragePoolView instance that the Volume is allocated from. . The CIM_AllocatedFromStoragePoolViewView is subclassed from CIM_AbstractElementAllocatedFromPool.

Requirement: Required if the array property CIM_ViewCapabilities.SupportedViews contains the strings "SNIA:StoragePoolView" and "SNIA:VolumeView" (and the Block Service Package is implemented).

Table 97 describes class CIM_AllocatedFromStoragePoolViewView (VolumeView to PoolView).

Table 97 - SMI Referenced Properties/Methods for CIM_AllocatedFromStoragePoolViewView (VolumeView to PoolView)

Properties	Flags	Requirement	Description & 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.

6.6.6 CIM_BasedOnView (ExtentOnDriveExtent)

The CIM_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 CIM_BaseOnView is subclassed from CIM_AbstractBasedOn.

Created By: External

Modified By: External

Deleted By: External

Requirement: Required if the array property CIM_ViewCapabilities.SupportedViews contains the string "SNIA:DiskDriveView" and Extent Composition is implemented.

Table 98 describes class CIM_BasedOnView (ExtentOnDriveExtent).

Table 98 - SMI Referenced Properties/Methods for CIM_BasedOnView (ExtentOnDriveExtent)

Properties	Flags	Requirement	Description & Notes
StartingAddress		Optional	This is derived from the BasedOn.StartingAddress.
EndingAddress		Optional	This is derived from the BasedOn.EndingAddress.
OrderIndex		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 CIM_DiskDriveView on which a concrete StorageExtent is based.
Dependent		Mandatory	The CIM_StorageExtent instance that is dependent on the CIM_DiskDriveView.

6.6.7 CIM_BasedOnView (VolumeOnExtent)

The CIM_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 CIM_BaseOnView is subclassed from CIM_AbstractBasedOn.

Created By: External

Modified By: External

Deleted By: External

Requirement: Required if the array property CIM_ViewCapabilities.SupportedViews contains the string "SNIA:VolumeView" and Extent Composition is implemented.

Table 99 describes class CIM_BasedOnView (VolumeOnExtent).

Table 99 - SMI Referenced Properties/Methods for CIM_BasedOnView (VolumeOnExtent)

Properties	Flags	Requirement	Description & Notes
StartingAddress		Optional	This is derived from the BasedOn.StartingAddress.
EndingAddress		Optional	This is derived from the BasedOn.EndingAddress.
OrderIndex		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 CIM_VolumeView StorageVolume is based.
Dependent		Mandatory	The CIM_VolumeView instance.

6.6.8 CIM_ConcreteComponentView

The CIM_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 CIM_ConcreteComponentView is subclassed from CIM_AbstractComponent.

Created By: External

Modified By: Static

Deleted By: External

Requirement: Required if the array property CIM_ViewCapabilities.SupportedViews contains the string "SNIA:DiskDriveView" (and the Disk Drive Lite Profile is implemented).

Table 100 describes class CIM_ConcreteComponentView.

Table 100 - SMI Referenced Properties/Methods for CIM_ConcreteComponentView

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	The CIM_StoragePool to which the StorageExtent of the Disk Drive is assigned.
PartComponent		Mandatory	A CIM_DiskDriveView instance that is assigned to the StoragePool.

6.6.9 CIM_ContainerView

The CIM_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 CIM_ContainerView is subclassed from CIM_AbstractComponent.

Created By: External

Modified By: Static

Deleted By: External

Requirement: Required if the array property CIM_ViewCapabilities.SupportedViews contains the string "SNIA:DiskDriveView" (and the Disk Drive Lite Profile is implemented).

Table 101 describes class CIM_ContainerView.

Table 101 - SMI Referenced Properties/Methods for CIM_ContainerView

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	The CIM_PhysicalPackage for the ComputerSystem instance that groups the CIM_PhysicalPackage of the Disk Drive.
PartComponent		Mandatory	A CIM_DiskDriveView instance that includes CIM_PhysicalPackage information for the CIM_DiskDrive.

6.6.10 CIM_DiskDriveView

The CIM_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 CIM_DiskDriveView is subclassed from CIM_View.

Created By: External

Modified By: External

Deleted By: External

Requirement: Required if the array property CIM_ViewCapabilities.SupportedViews contains the string "SNIA:DiskDriveView" (and the Disk Drive Lite Profile is implemented).

Table 102 describes class CIM_DiskDriveView.

Table 102 - SMI Referenced Properties/Methods for CIM_DiskDriveView

Properties	Flags	Requirement	Description & 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.

Table 102 - SMI Referenced Properties/Methods for CIM_DiskDriveView

Properties	Flags	Requirement	Description & Notes
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.
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.

Table 102 - SMI Referenced Properties/Methods for CIM_DiskDriveView

Properties	Flags	Requirement	Description & Notes
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.
LPPortType		Optional	Experimental. This is an array property that contains the PortTypes for the target ports that may be used to access the disk drive.

6.6.11 CIM_DriveComponentViewView

Experimental. The CIM_DriveComponentViewView is an association between a CIM_StoragePoolView instances and the CIM_DiskDriveView instances for Disk Drives of the StoragePool. The CIM_DriveComponentViewView is subclassed from CIM_AbstractComponent.

Requirement: Required if the array property CIM_ViewCapabilities.SupportedViews contains the string "SNIA:StoragePoolView" and "SNIA:DiskDriveView" (and the Disk Drive Lite Profile is implemented).

Table 103 describes class CIM_DriveComponentViewView.

Table 103 - SMI Referenced Properties/Methods for CIM_DriveComponentViewView

Properties	Flags	Requirement	Description & Notes
PartComponent		Mandatory	The DiskDriveView instance.
GroupComponent		Mandatory	The StoragePoolView instance for a primordial StoragePool.

6.6.12 CIM_ElementCapabilities (View Capabilities)

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

Table 104 describes class CIM_ElementCapabilities (View Capabilities).

Table 104 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (View Capabilities)

Properties	Flags	Requirement	Description & Notes
Capabilities		Mandatory	The ViewCapabilities.
ManagedElement		Mandatory	The top level ComputerSystem that has the ViewCapabilities.

6.6.13 CIM_ElementStatisticalDataView (DiskDriveView)

The CIM_ElementStatisticalDataView is an association between a CIM_DiskDriveView instance and the CIM_BlockStorageStatisticalData instance for the DiskDrive. The CIM_ElementStatisticalDataView is subclassed from CIM_AbstractElementStatisticalData.

Created By: External

Modified By: External

Deleted By: External

Requirement: Required if the array property CIM_ViewCapabilities.SupportedViews contains the string "SNIA:DiskDriveView", CIM_BlockStatisticsCapabilities.ElementTypesSupported contains "10" and Block Server Performance is implemented.

Table 105 describes class CIM_ElementStatisticalDataView (DiskDriveView).

Table 105 - SMI Referenced Properties/Methods for CIM_ElementStatisticalDataView (Disk-DriveView)

Properties	Flags	Requirement	Description & Notes
Stats		Mandatory	The CIM_BlockStorageStatisticalData instance for the DiskDrive (StorageExtent) instance.
ManagedElement		Mandatory	The CIM_DiskDriveView instance that has the CIM_BlockStorageStatisticalData instance.

6.6.14 CIM_ElementStatisticalDataView (VolumeView)

The CIM_ElementStatisticalDataView is an association between a CIM_VolumeView instance and the CIM_BlockStorageStatisticalData instance for the StorageVolume (or LogicalDisk). The CIM_ElementStatisticalDataView is subclassed from CIM_AbstractElementStatisticalData.

Created By: External

Modified By: External

Deleted By: External

Requirement: Required if the array property CIM_ViewCapabilities.SupportedViews contains the string "SNIA:VolumeView", CIM_BlockStatisticsCapabilities.ElementTypesSupported contains "8" and Block Server Performance is implemented.

Table 106 describes class CIM_ElementStatisticalDataView (VolumeView).

Table 106 - SMI Referenced Properties/Methods for CIM_ElementStatisticalDataView (Vol-umeView)

Properties	Flags	Requirement	Description & Notes
Stats		Mandatory	The CIM_BlockStorageStatisticalData instance for the StorageVolume (or LogicalDisk) instance.
ManagedElement		Mandatory	The CIM_VolumeView instance that has the CIM_BlockStorageStatisticalDatainstance.

6.6.15 CIM_ElementView (DiskDrive)

The CIM_ElementView instance is an association between a CIM_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 CIM_ElementView is subclassed from CIM_Dependency.

Created By: External

Modified By: External

Deleted By: External

Requirement: Required if the array property CIM_ViewCapabilities.SupportedViews contains the string "SNIA:DiskDriveView" (and the Disk Drive Lite Profile is implemented).

Table 107 describes class CIM_ElementView (DiskDrive).

Table 107 - SMI Referenced Properties/Methods for CIM_ElementView (DiskDrive)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	The base CIM_DiskDrive instance on which the CIM_DiskDriveView instance is based.
Dependent		Mandatory	The CIM_DiskDriveView instance that is based on the CIM_DiskDrive instance.

6.6.16 CIM_ElementView (StorageSetting)

The CIM_ElementView instance is an association between the CIM_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 CIM_ElementView is subclassed from CIM_Dependency.

Created By: External

Modified By: External

Deleted By: External

Requirement: Required if the array property CIM_ViewCapabilities.SupportedViews contains the string "SNIA:VolumeView" (and the Block Service Package is implemented).

Table 108 describes class CIM_ElementView (StorageSetting).

Table 108 - SMI Referenced Properties/Methods for CIM_ElementView (StorageSetting)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	The base CIM_StorageSetting instance on which the CIM_VolumeView instance is based.
Dependent		Mandatory	The CIM_VolumeView instance that is based on the CIM_StorageSetting instance.

6.6.17 CIM_ElementView (Volume)

The CIM_ElementView instance is an association between a CIM_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 CIM_ElementView is subclassed from CIM_Dependency.

Created By: External

Modified By: External

Deleted By: External

Requirement: Required if the array property CIM_ViewCapabilities.SupportedViews contains the string "SNIA:VolumeView" (and the Block Service Package is implemented).

Table 109 describes class CIM_ElementView (Volume).

Table 109 - SMI Referenced Properties/Methods for CIM_ElementView (Volume)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	The base CIM_StorageVolume (or CIM_LogicalDisk) instance on which the CIM_VolumeView instance is based.
Dependent		Mandatory	The CIM_VolumeView instance that is based on the CIM_StorageVolume (or CIM_LogicalDisk) instance.

6.6.18 CIM_ExtentComponentView

Experimental. The CIM_ExtentComponentView is an association between a CIM_StoragePoolView instances and the CIM_StorageExtent instances for the StoragePool. The CIM_ExtentComponentView is subclassed from CIM_AbstractComponent.

Requirement: Required if the array property CIM_ViewCapabilities.SupportedViews contains the string "SNIA:StoragePoolView" (and the Block Service Package is implemented).

Table 110 describes class CIM_ExtentComponentView.

Table 110 - SMI Referenced Properties/Methods for CIM_ExtentComponentView

Properties	Flags	Requirement	Description & Notes
PartComponent		Mandatory	A reference to a StorageExtent.
GroupComponent		Mandatory	A reference to a StoragePoolView instance that contains the Extent.

6.6.19 CIM_HostedStoragePoolView

The CIM_HostedStoragePoolView is an association between a CIM_StoragePoolView instances and the CIM_ComputerSystem instance for the StoragePool. The CIM_HostedStoragePoolView is subclassed from CIM_ScopedView.

Requirement: Required if the array property CIM_ViewCapabilities.SupportedViews contains the string "SNIA:StoragePoolView" (and the Block Service Package is implemented).

Table 111 describes class CIM_HostedStoragePoolView.

Table 111 - SMI Referenced Properties/Methods for CIM_HostedStoragePoolView

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

6.6.20 CIM_MappingProtocolControllerView

Experimental. The CIM_MappingProtocolControllerView instance is a view that is derived from CIM_ProtocolController, CIM_StorageHardwareID, CIM_AuthorizedPrivilege, CIM_ProtocolEndPoint and CIM_LogicalPort, and their associations. The CIM_MappingProtocolControllerView is subclassed from CIM_View.

Created By: External

Modified By: External

Deleted By: External

Requirement: Required if the array property CIM_ViewCapabilities.SupportedViews contains the string "SNIA:MappingProtocolControllerView" (and the Masking and Mapping Profile is implemented).

Table 112 describes class CIM_MappingProtocolControllerView.

Table 112 - SMI Referenced Properties/Methods for CIM_MappingProtocolControllerView

Properties	Flags	Requirement	Description & 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	N	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.

Table 112 - SMI Referenced Properties/Methods for CIM_MappingProtocolControllerView

Properties	Flags	Requirement	Description & Notes
PEPName		Mandatory	The Name as reported in the underlying ProtocolEndpoint.
PEPProtocolIFType		Mandatory	The ProtocolIFType as reported in the underlying ProtocolEndpoint.
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.
APInstanceID		Mandatory	The InstanceID as reported in the underlying AuthorizedPrivilege.
APPrivilegeGranted		Mandatory	The PrivilegeGranted as reported in the underlying AuthorizedPrivilege.
APActivities[]		Mandatory	The Activities[] as reported in the underlying AuthorizedPrivilege.
APElementName		Optional	The ElementName as reported in the underlying AuthorizedPrivilege.
LPSystemCreationClassName	N	Mandatory	The SystemCreationClassName as reported in the underlying LogicalPort. This may be NULL if the underlying LogicalPort is an Ethernet Port.
LPCreationClassName	N	Mandatory	The CreationClassName as reported in the underlying LogicalPort. This may be NULL if the underlying LogicalPort is an Ethernet Port.
LPSystemName	N	Mandatory	The SystemName as reported in the underlying LogicalPort. This may be NULL if the underlying LogicalPort is an Ethernet Port.
LPDeviceID	N	Mandatory	The DeviceID as reported in the underlying LogicalPort. This may be NULL if the underlying LogicalPort is an Ethernet Port.
LPOperationalStatus	N	Mandatory	The OperationalStatus as reported in the underlying LogicalPort. This may be NULL if the underlying LogicalPort is an Ethernet Port.
LPUsageRestriction	N	Mandatory	The UsageRestriction as reported in the underlying LogicalPort. This may be NULL if the underlying LogicalPort is an Ethernet Port.
LPPortType	N	Mandatory	The PortType as reported in the underlying LogicalPort. This may be NULL if the underlying LogicalPort is an Ethernet Port.

6.6.21 CIM_MaskingMappingExposedDeviceView

The CIM_MaskingMappingExposedDeviceView instance is a view that is derived from CIM_SAPAvailableForElement, CIM_SCSIProtocolController and CIM_ProtocolControllerForUnit. The CIM_MaskingMappingExposedDeviceView is not subclassed from anything.

Created By: External

Modified By: External

Deleted By: External

Requirement: Required if the array property CIM_ViewCapabilities.SupportedViews contains the string "SNIA:MaskingMappingExposedDeviceView" (and the Masking and Mapping Profile is implemented).

Table 113 describes class CIM_MaskingMappingExposedDeviceView.

Table 113 - SMI Referenced Properties/Methods for CIM_MaskingMappingExposedDeviceView

Properties	Flags	Requirement	Description & Notes
SPCSystemCreation ClassName		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.
SPCCreationClassN ame		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.

6.6.22 CIM_MaskingMappingView

The CIM_MaskingMappingView 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 CIM_MaskingMappingView is not subclassed from anything.

Created By: External

Modified By: External

Deleted By: External

Requirement: Required if the array property CIM_ViewCapabilities.SupportedViews contains the string "SNIA:MaskingMappingView" (and the Masking and Mapping Profile is implemented).

Table 114 describes class CIM_MaskingMappingView.

Table 114 - SMI Referenced Properties/Methods for CIM_MaskingMappingView

Properties	Flags	Requirement	Description & 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.

Table 114 - SMI Referenced Properties/Methods for CIM_MaskingMappingView

Properties	Flags	Requirement	Description & Notes
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.

6.6.23 CIM_ProtocolControllerForUnitView

Experimental. The CIM_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 CIM_ViewCapabilities.SupportedViews contains the string "SNIA:MappingProtocolControllerView" (and the Masking and Mapping Profile is implemented).

Table 115 describes class CIM_ProtocolControllerForUnitView.

Table 115 - SMI Referenced Properties/Methods for CIM_ProtocolControllerForUnitView

Properties	Flags	Requirement	Description & Notes
DeviceNumber		Mandatory	The DeviceNumber as reported in the underlying ProtocolControllerForUnit.
PCFUDeviceAccess		Mandatory	The DeviceAccess as reported in the underlying ProtocolControllerForUnit.
Antecedent		Mandatory	The MappingProtocolControllerView Instance.
Dependent		Mandatory	The Storage Volume.

6.6.24 CIM_ReplicaPairView

Experimental. The CIM_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 CIM_ViewCapabilities.SupportedViews contains the string "SNIA:ReplicaPairlView" (and the Copy Services Profile is implemented).

Table 116 describes class CIM_ReplicaPairView.

Table 116 - SMI Referenced Properties/Methods for CIM_ReplicaPairView

Properties	Flags	Requirement	Description & 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).
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).

Table 116 - SMI Referenced Properties/Methods for CIM_ReplicaPairView

Properties	Flags	Requirement	Description & Notes
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).
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).

Table 116 - SMI Referenced Properties/Methods for CIM_ReplicaPairView

Properties	Flags	Requirement	Description & Notes
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).
SVTargetDeltaReservation		Mandatory	The DeltaReservation supported by the volume as reported in the underlying target StorageVolume (or LogicalDisk).

Table 116 - SMI Referenced Properties/Methods for CIM_ReplicaPairView

Properties	Flags	Requirement	Description & Notes
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).

6.6.25 CIM_StoragePoolView

Experimental. The CIM_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 CIM_ViewCapabilities.SupportedViews contains the string "SNIA:StoragePoolView" (and the Block Service Package is implemented).

Table 117 describes class CIM_StoragePoolView.

Table 117 - SMI Referenced Properties/Methods for CIM_StoragePoolView

Properties	Flags	Requirement	Description & 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.

Table 117 - SMI Referenced Properties/Methods for CIM_StoragePoolView

Properties	Flags	Requirement	Description & Notes
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.
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.

Table 117 - SMI Referenced Properties/Methods for CIM_StoragePoolView

Properties	Flags	Requirement	Description & Notes
AFSPSpaceConsumed		Mandatory	The SpaceConsumed as reported in the underlying AllocatedFromStoragePool to this pool's parent pool.
SCCInstanceID	N	Mandatory	The InstanceID as reported in the underlying StorageConfigurationCapabilities (if any) associated to the StoragePool.
SCCElementName	N	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.
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.

6.6.26 CIM_SystemDeviceView (DiskDriveViews)

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

Requirement: Required if the array property CIM_ViewCapabilities.SupportedViews contains the string "SNIA:DiskDriveView" (and the Disk Drive Lite Profile is implemented).

Table 118 describes class CIM_SystemDeviceView (DiskDriveViews).

Table 118 - SMI Referenced Properties/Methods for CIM_SystemDeviceView (DiskDriveViews)

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	The Computer System that contains this DiskDriveView instance.
PartComponent		Mandatory	The CIM_DiskDriveView instance that is a device on the computer system.

6.6.27 CIM_SystemDeviceView (MappingProtocolControllerViews)

Created By: External

Modified By: Static

Deleted By: External

Requirement: Required if the array property CIM_ViewCapabilities.SupportedViews contains the string "SNIA:MappingProtocolControllerView" (and the Masking and Mapping Profile is implemented).

Table 119 describes class CIM_SystemDeviceView (MappingProtocolControllerViews).

Table 119 - SMI Referenced Properties/Methods for CIM_SystemDeviceView (MappingProtocol-ControllerViews)

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	The Computer System that contains this MappingProtocolControllerView instance.
PartComponent		Mandatory	The CIM_MappingProtocolControllerView instance that is a device on the computer system.

6.6.28 CIM_SystemDeviceView (ReplicaPairViews)

Created By: External

Modified By: Static

Deleted By: External

Requirement: Required if the array property CIM_ViewCapabilities.SupportedViews contains the string "SNIA:ReplicaPairlView" (and the Copy Services Profile is implemented).

Table 120 describes class CIM_SystemDeviceView (ReplicaPairViews).

Table 120 - SMI Referenced Properties/Methods for CIM_SystemDeviceView (ReplicaPairViews)

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	The Computer System that contains this ReplicaPairView instance.
PartComponent		Mandatory	The CIM_ReplicaPairView instance that is a device on the computer system.

6.6.29 CIM_SystemDeviceView (VolumeViews)

Created By: External

Modified By: Static

Deleted By: External

Requirement: Required if the array property CIM_ViewCapabilities.SupportedViews contains the string "SNIA:VolumeView" (and the Block Service Package is implemented).

Table 121 describes class CIM_SystemDeviceView (VolumeViews).

Table 121 - SMI Referenced Properties/Methods for CIM_SystemDeviceView (VolumeViews)

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	The Computer System that contains this VolumeView instance.
PartComponent		Mandatory	The CIM_VolumeView instance that is a device on the computer system.

6.6.30 CIM_ViewCapabilities

The CIM_ViewCapabilities instance defines the capabilities of an implementation support for CIM_ view classes. The CIM_ViewCapabilities is subclassed from CIM_Capabilities.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 122 describes class CIM_ViewCapabilities.

Table 122 - SMI Referenced Properties/Methods for CIM_ViewCapabilities

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	An opaque, unique id for the view class capability of an implementation.

Table 122 - SMI Referenced Properties/Methods for CIM_ViewCapabilities

Properties	Flags	Requirement	Description & Notes
ElementName		Optional	A provider supplied user-Friendly Name for this CIM_ViewCapabilities element.
SupportedViews		Mandatory	This array of strings lists the view classes that are supported by the implementation. Valid string values are "SNIA:VolumeView" | "SNIA:DiskDriveView" | "SNIA:MaskingMappingExposedDeviceView" | "SNIA:MaskingMappingView"|"SNIA:MappingProtocolControllerView" | "SNIA:StoragePoolView" |"SNIA:ReplicaPairView" .

6.6.31 CIM_VolumeView

The CIM_VolumeView instance is a view that is derived from CIM_StorageVolume, CIM_ElementSettingData, CIM_StorageSetting, CIM_AllocatedFromStoragePool and CIM_StoragePool. The CIM_VolumeView is subclassed from CIM_View.

Created By: External

Modified By: External

Deleted By: External

Requirement: Required if the array property CIM_ViewCapabilities.SupportedViews contains the string "SNIA:VolumeView" (and the Block Service Package is implemented).

Table 123 describes class CIM_VolumeView.

Table 123 - SMI Referenced Properties/Methods for CIM_VolumeView

Properties	Flags	Requirement	Description & 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).

Table 123 - SMI Referenced Properties/Methods for CIM_VolumeView

Properties	Flags	Requirement	Description & Notes
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.
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.

Table 123 - SMI Referenced Properties/Methods for CIM_VolumeView

Properties	Flags	Requirement	Description & Notes
SSPackageRedundancyMax		Mandatory	The PackageRedundancyMax value supported by the StorageSetting for the volume as reported in the 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).
SVCanDelete		Optional	Experimental. The CanDelete supported by the volume as reported in the underlying StorageVolume.
SVIsComposite		Optional	Experimental. The IsComposite supported by the volume as reported in the underlying StorageVolume.
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.

Table 123 - SMI Referenced Properties/Methods for CIM_VolumeView

Properties	Flags	Requirement	Description & Notes
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.
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	

6.6.32 SNIA_DiskDriveView

Experimental. The SNIA_DiskDriveView instance is a view that is derived from CIM_StorageExtent, CIM_MediaPresent, SNIA_DiskDrive, CIM_Realizes, CIM_PhysicalPackage, CIM_ElementSoftwareIdentity and CIM_SoftwareIdentity. The SNIA_DiskDriveView is subclassed from CIM_DiskDriveView.

Created By: External

Modified By: External

Deleted By: External

Requirement: Optional

Table 124 describes class SNIA_DiskDriveView.

Table 124 - SMI Referenced Properties/Methods for SNIA_DiskDriveView

Properties	Flags	Requirement	Description & 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.

Table 124 - SMI Referenced Properties/Methods for SNIA_DiskDriveView

Properties	Flags	Requirement	Description & Notes
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 124 - SMI Referenced Properties/Methods for SNIA_DiskDriveView

Properties	Flags	Requirement	Description & 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.
DDDiskType		Optional	The DiskType for the Disk Drive as reported in the underlying DiskDrive instance.
DDFormFactor		Optional	The FormFactor for the Disk Drive as reported in the underlying DiskDrive instance.
DDEncryption		Optional	The Encryption 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.
LPPortType		Optional	This is an array property that contains the PortTypes for the target ports that may be used to access the disk drive.

EXPERIMENTAL

STABLE

Clause 7: Block Server Performance Subprofile

7.1 Description

7.1.1 Synopsis

Profile Name: Block Server Performance (Component Profile)

Version: 1.5.0

Organization: SNIA

CIM Schema Version: 2.11.0

Table 125 describes the related profiles for Block Server Performance.

Table 125 - Related Profiles for Block Server Performance

Profile Name	Organization	Version	Requirement	Description
Multiple Computer System	SNIA	1.2.0	Optional	
Extent Composition	SNIA	1.6.0	Optional	
SPI Target Ports	SNIA	1.4.0	Optional	
FC Target Ports	SNIA	1.4.0	Optional	
iSCSI Target Ports	SNIA	1.6.0	Optional	
DA Target Ports	SNIA	1.4.0	Optional	
SPI Initiator Ports	SNIA	1.4.0	Optional	
FC Initiator Ports	SNIA	1.6.0	Optional	
iSCSI Initiator Ports	SNIA	1.2.0	Optional	
Disk Drive Lite	SNIA	1.6.0	Optional	
Replication Services	SNIA	1.6.0	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

7.1.2 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:** 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 specification. Mapping functions are listed within the specification today. As new objects (like cache which is currently not defined) and new relationships between objects are defined, these parts of this specification 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 specification 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: 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 7.7.7.

- 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).

7.2 Implementation

7.2.1 Performance Additions Overview

Figure 36: "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.

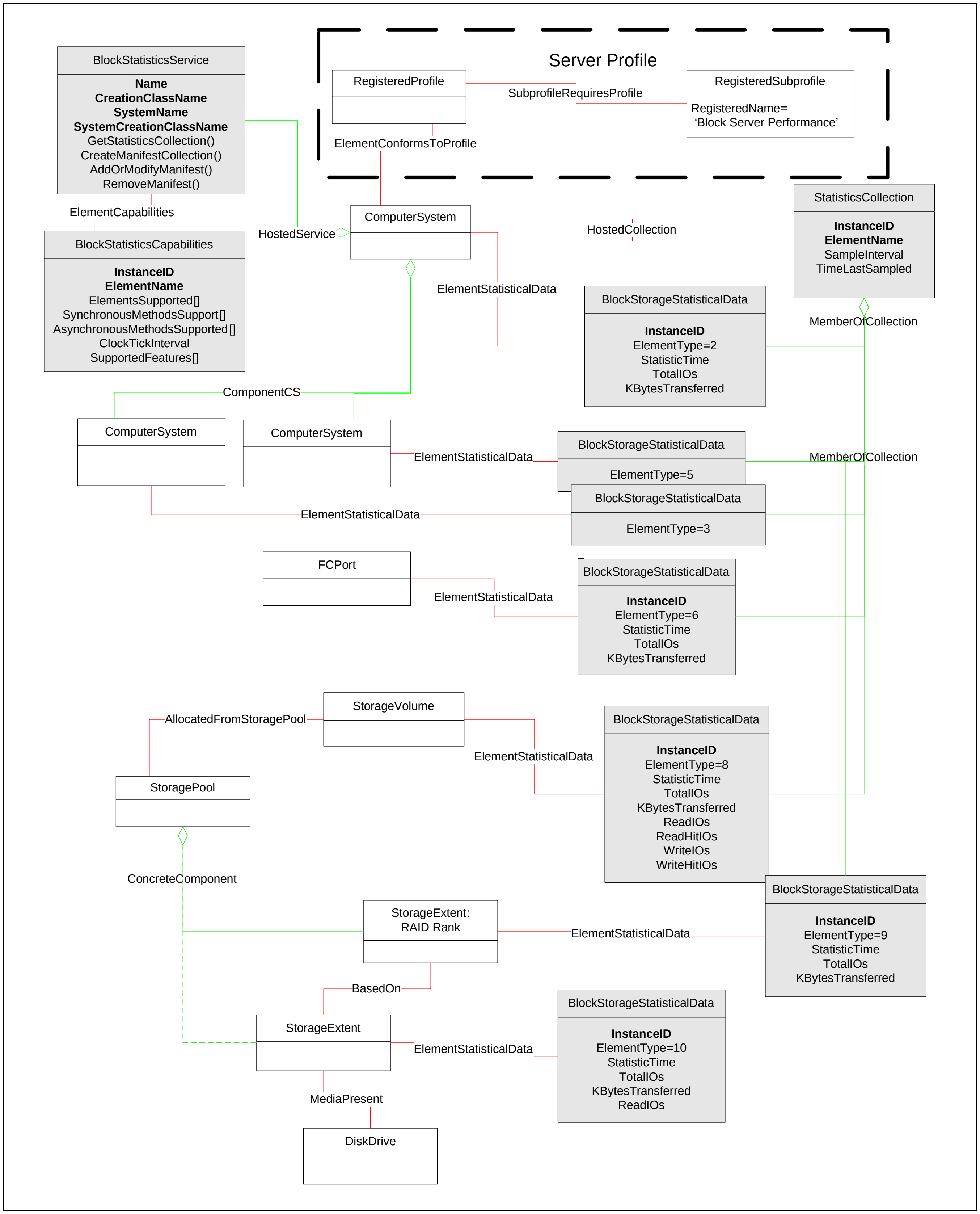

Figure 36 - Block Server Performance Subprofile Summary Instance Diagram

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 7.8 "CIM Elements".

What Figure 36 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: 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: 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 37: "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

7.2.2 Performance Additions to base Array Profile

Figure 37: "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.

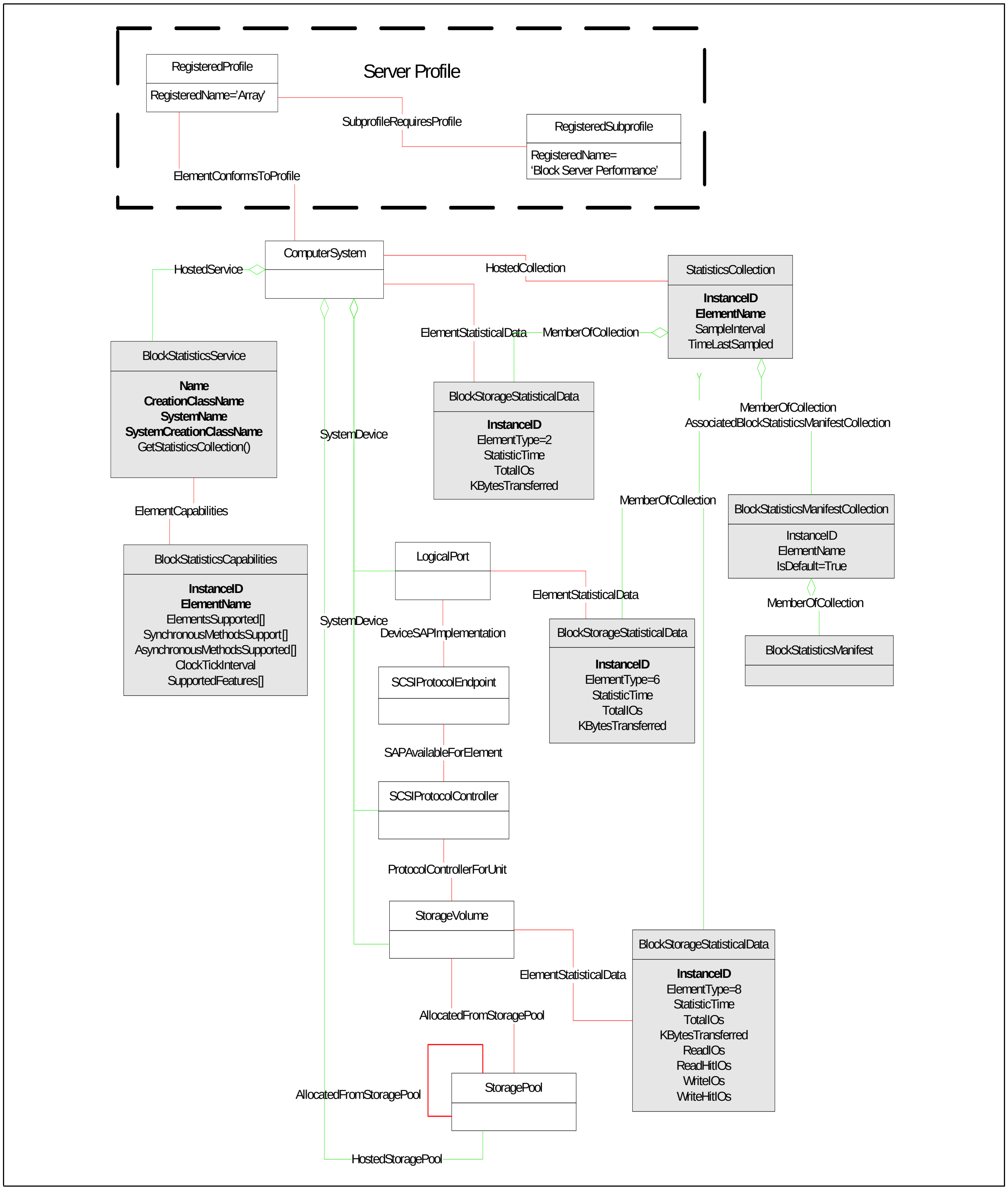

Figure 37 - Base Array Profile Block Server Performance Instance Diagram

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 7.8 "CIM Elements".

7.2.3 Performance Additions to base Storage Virtualizer Profile

Figure 38: "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.

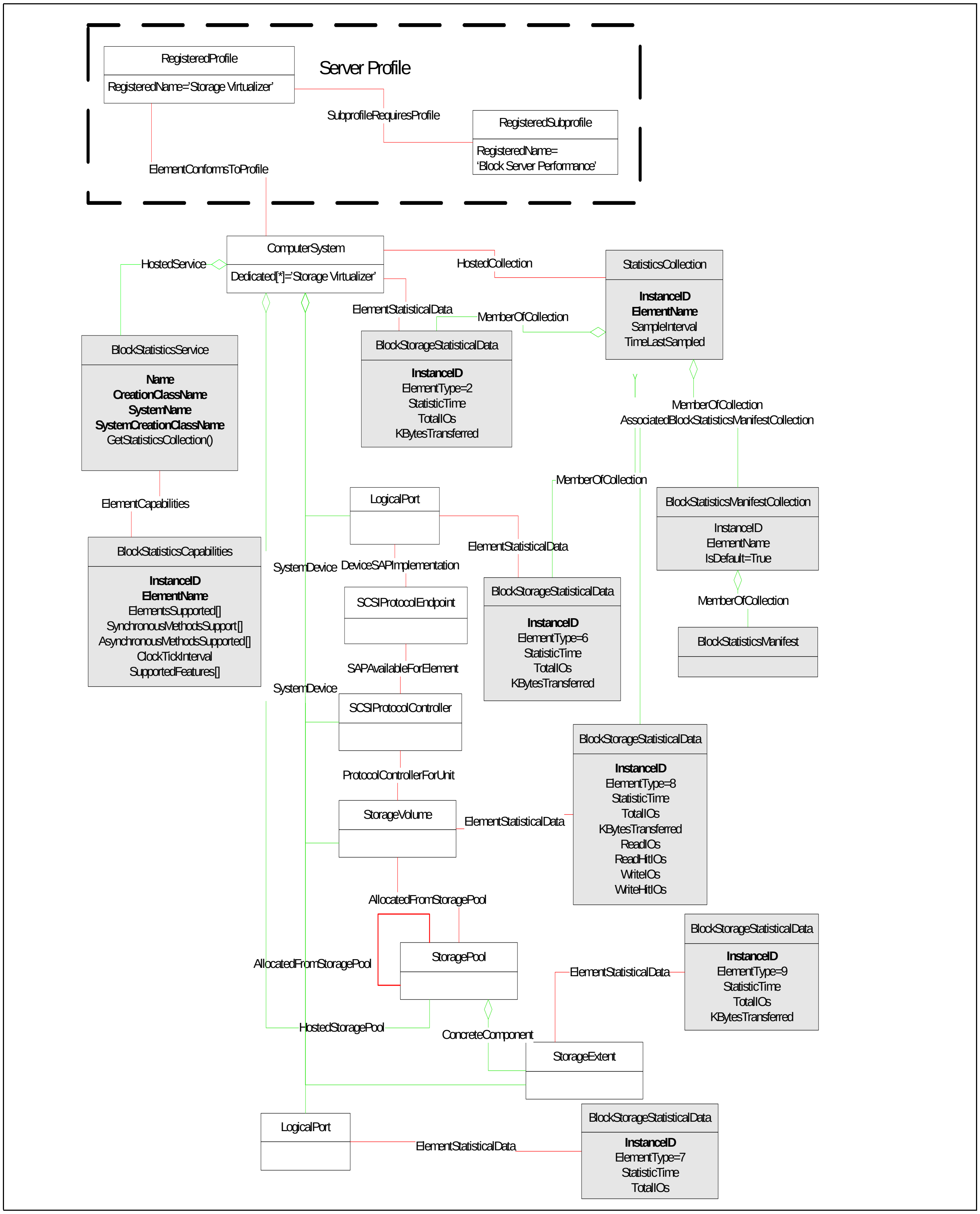

Figure 38 - Base Storage Virtualizer Profile Block Server Performance Instance Diagram

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 7.8 "CIM Elements".

7.2.4 Performance Additions to base Volume Management Profile

Figure 39: "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.

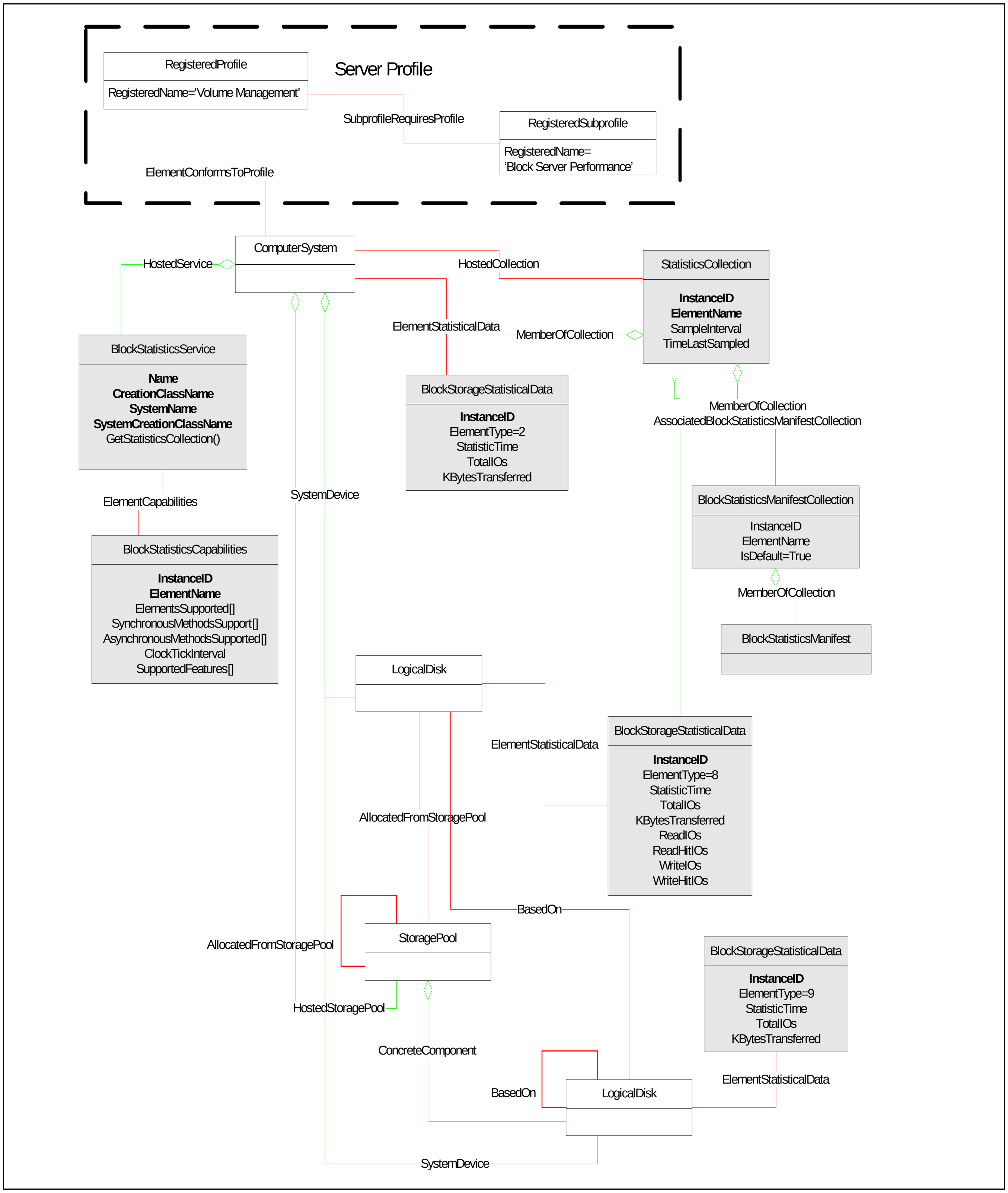

Figure 39 - Base Volume Management Profile Block Server Performance Instance Diagram

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 7.8 "CIM Elements".

7.2.5 Summary of BlockStorageStatisticsData support by Profile

Table 126 defines the Element Types (for BlockStorageStatisticalData instances) that may be supported by profile.

Table 126 - 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 specification defines the element type for the profile. Actual support by any given implementation would be implementation dependent. But the specification covers defining the element type for the profile. NO means that this specification does not specify this element type for the profile.

7.2.6 Server Profile Support for the Block Server Performance Subprofile

At the top of Figure 37: "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.

7.2.7 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 BlockStrorageStatisticalData 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

7.2.8 Performance Additions applied to Multiple Computer Systems

Figure 40: "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: 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.

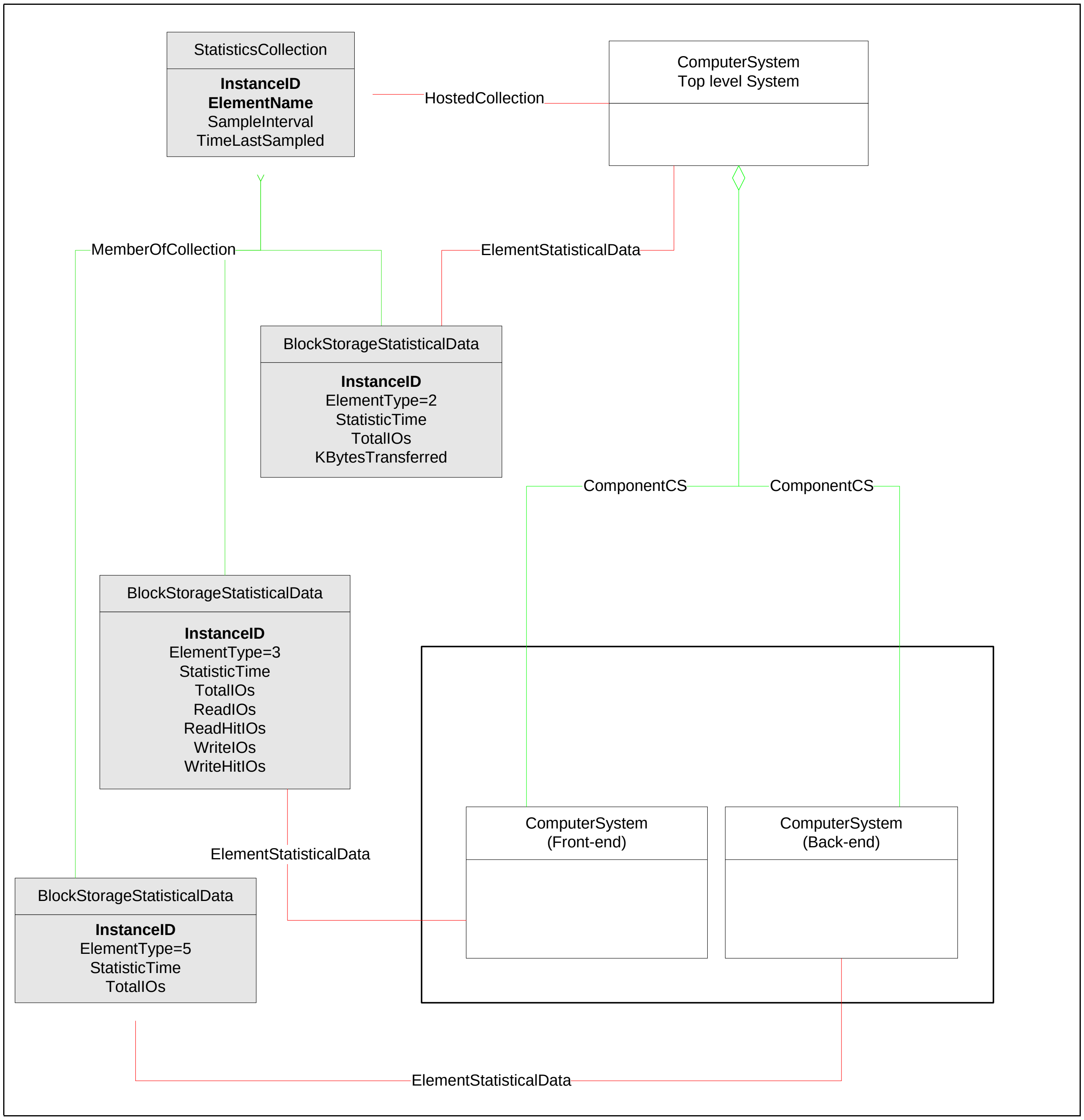

Figure 40 - Multiple Computer System Subprofile Block Server Performance Instance Diagram

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 7.8 "CIM Elements".

7.2.9 Performance Additions to Backend Ports

Figure 41: "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: 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.

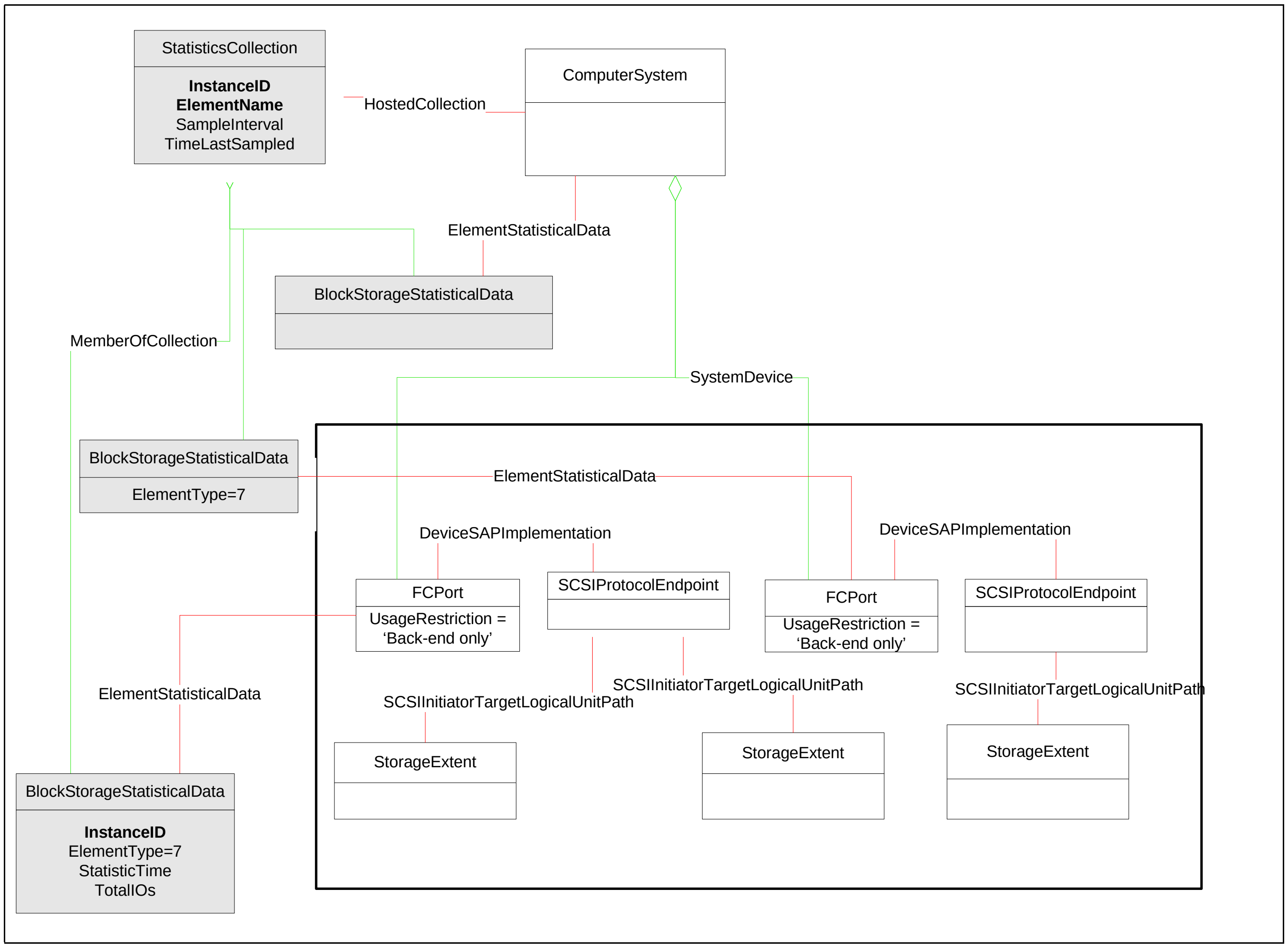

Figure 41 - Fibre Channel Initiator Port Subprofile Block Server Performance Instance Diagram

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 7.8 "CIM Elements".

EXPERIMENTAL

In some systems a port may be either a front-end or backend port. In this 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

7.2.10 Performance Additions to Extent Composition

Figure 42: "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: 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.

Note: 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.

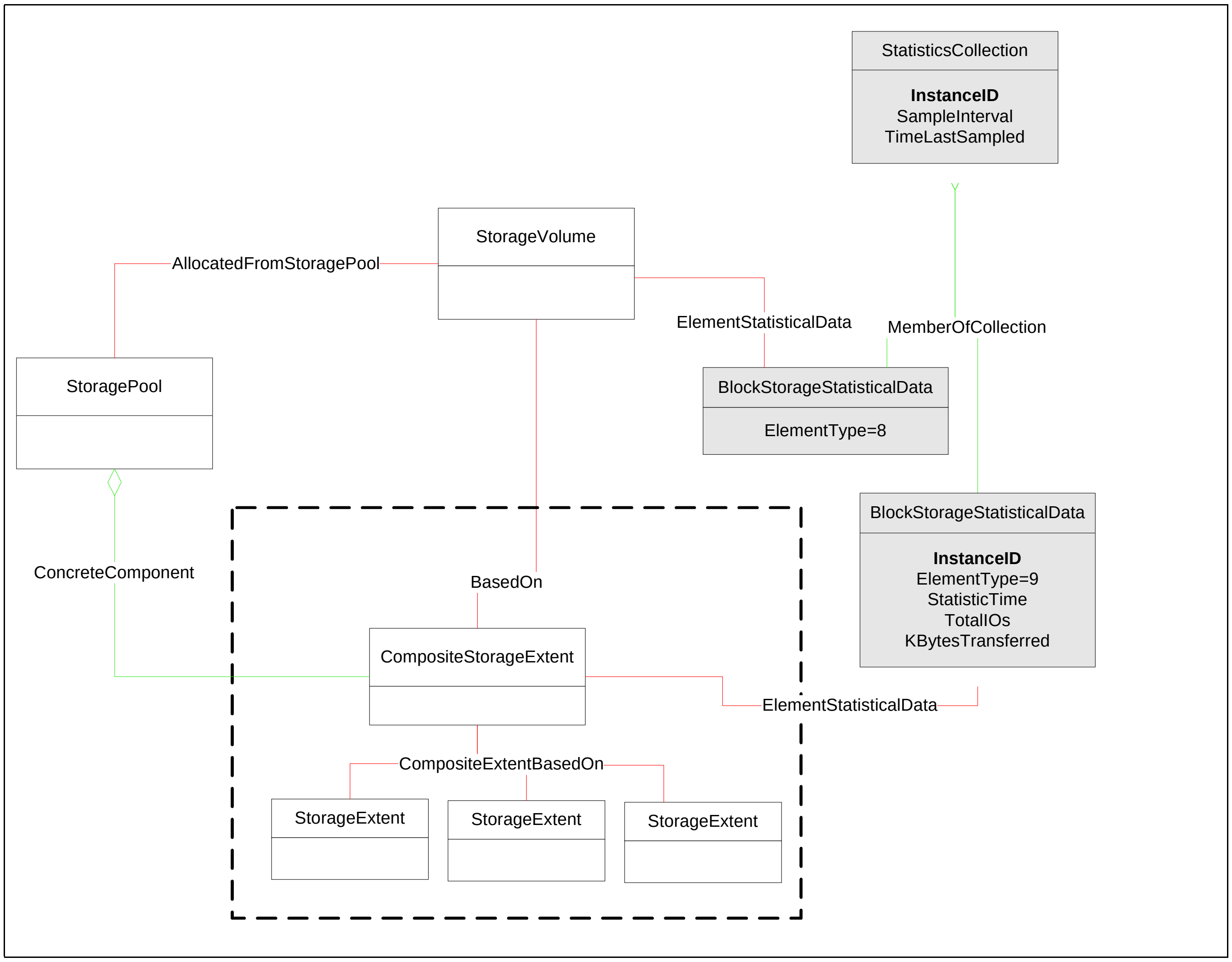

Figure 42 - Extent Composition Subprofile Block Server Performance Instance Diagram

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 7.8 "CIM Elements".

Note: The low level extents represent Disk Drive Extents and they would not be part of the Storage Virtualizer or Volume Management Profiles.

7.2.11 Performance Additions to Disk Drives

Figure 43: "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: The Volume Management Profiles would NEVER show the “Disks” statistics. Also note that an Array or Storage Virtualizer may model both “Extents” and “Disks”. Note: 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.

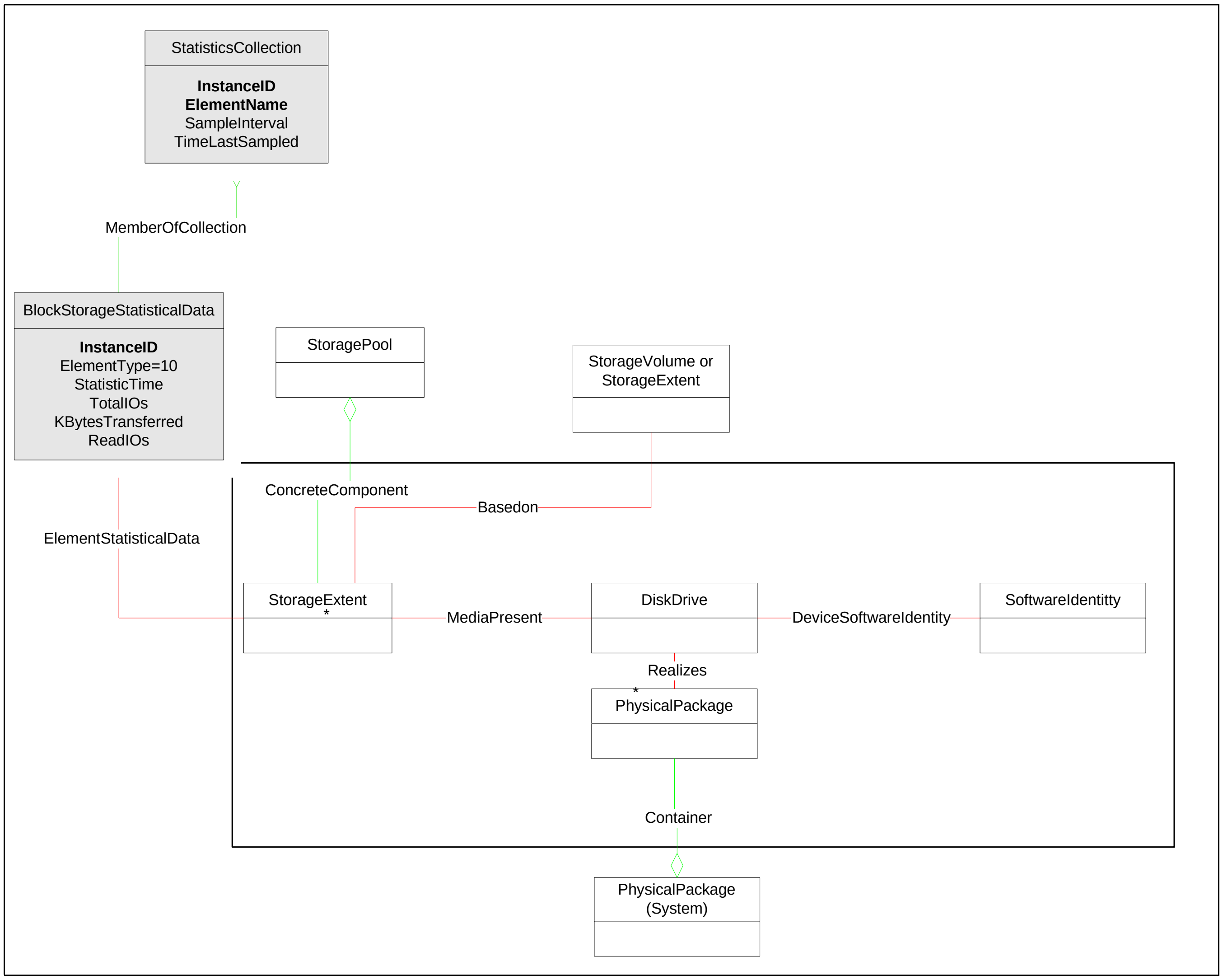

Figure 43 - Disk Drive Lite Subprofile Block Server Performance Instance Diagram

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 7.8 "CIM Elements".

7.2.12 Performance Additions to SCSIArbitraryLogicalUnits (Controller LUNs)

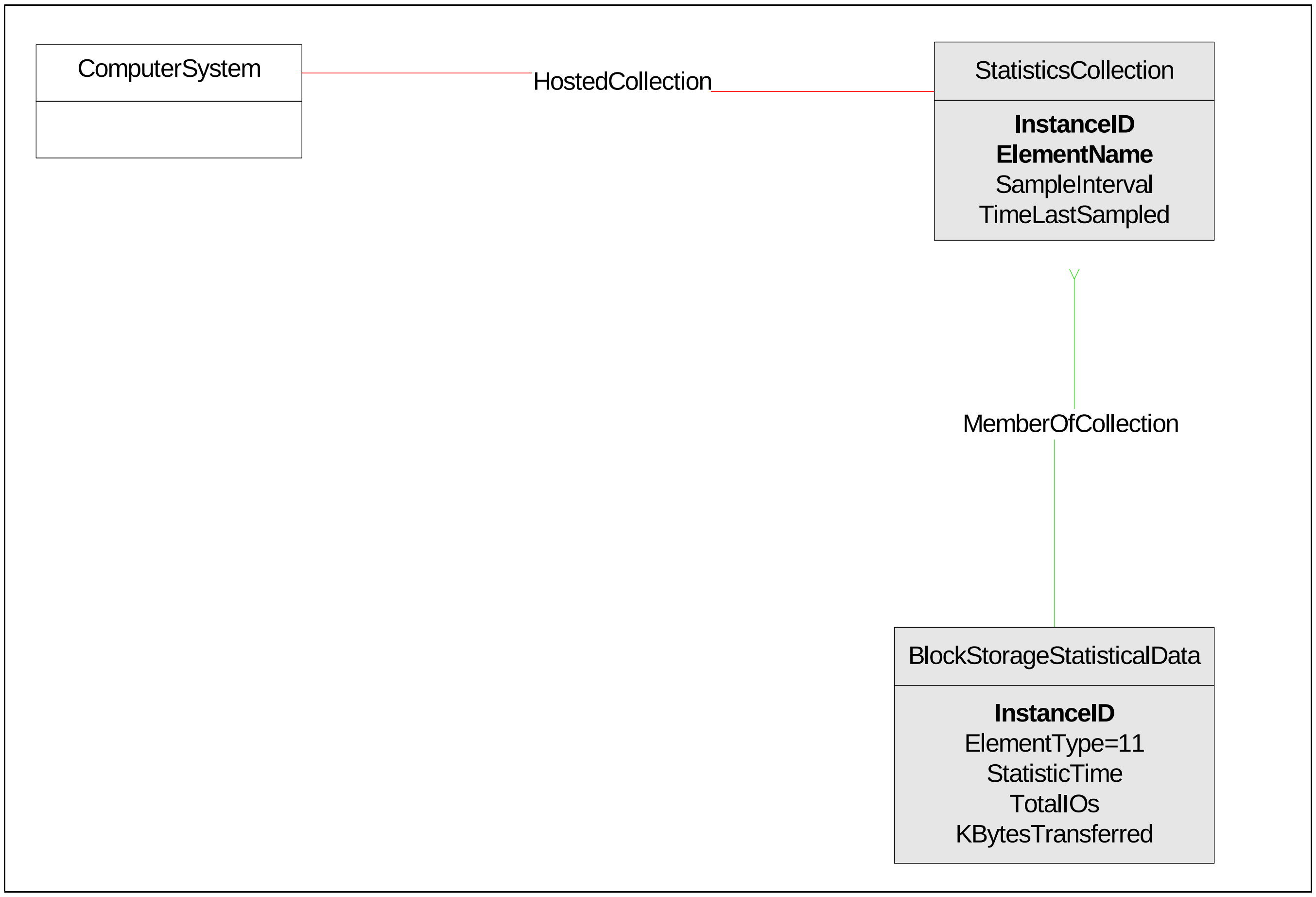

Figure 44 - SCSIArbitraryLogicalUnit Block Server Performance Instance Diagram

Figure 44: "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: 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: 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 7.8 "CIM Elements".

EXPERIMENTAL

7.2.13 Performance Additions for Remote Mirrors

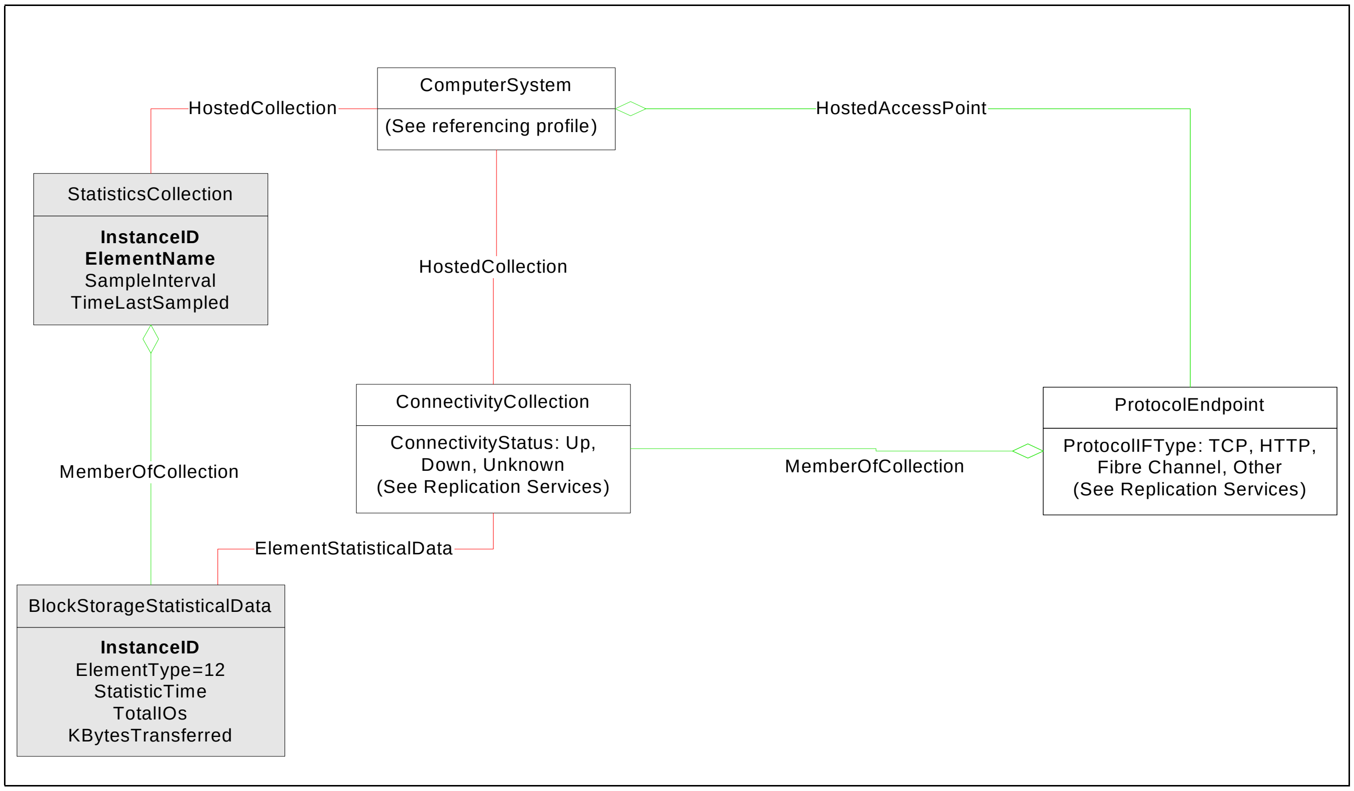

Figure 45 - Remote Mirrors Block Server Performance Instance Diagram

Figure 45: "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., meta data) IO requests. In this case, the BlockStorageStatisticalData instance is associated with the ConnectivityCollection instance that represents the connection to the remote system. **Note:** 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: 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 7.8 "CIM Elements".

EXPERIMENTAL

7.2.14 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 7.2.7.

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 46: "Block Server Performance Manifest Collections".

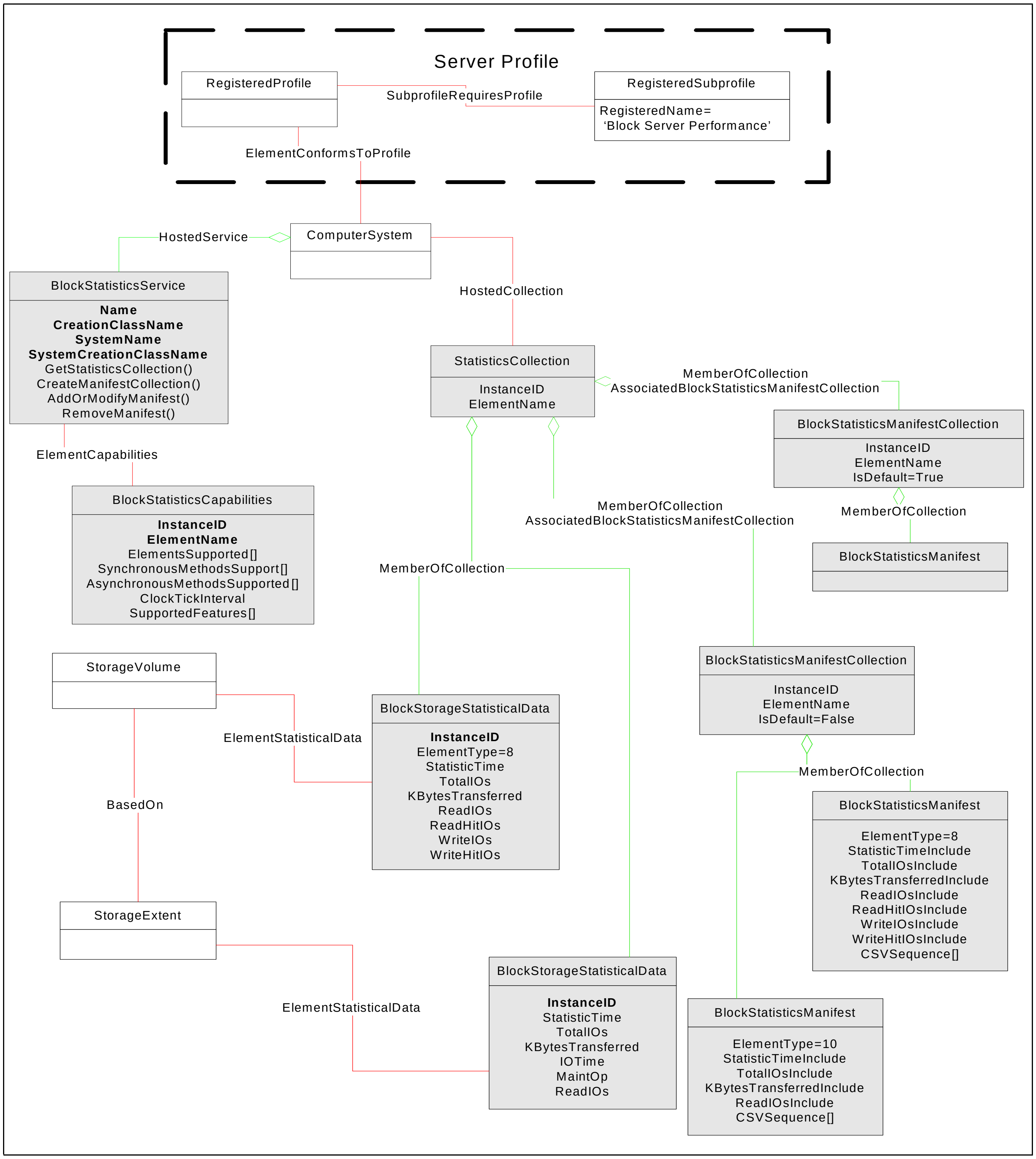

Figure 46 - Block Server Performance Manifest Collections

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 46: "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.

7.2.15 Capabilities Support for Block Server Performance Subprofile

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.

7.2.15.1 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”

7.2.15.2 SynchronousMethodsSupported

The values of interest are ”Exec Query”, “Indications”, “Query Collection”, “GetStatisticsCollection”, “Manifest Creation”, “Manifest Modification”, and “Manifest Removal”

7.2.15.3 AsynchronousMethodsSupported

For the current version of the standard this should be NULL.

EXPERIMENTAL

7.2.15.4 SupportedFeatures

The values of interest are “none” and “Client Defined Sequence”.

EXPERIMENTAL

7.2.15.5 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.

7.3 Health and Fault Management Considerations

Not defined in this standard.

7.4 Cascading Considerations

Not applicable.

7.5 Supported Subprofiles and Packages

See section 7.1.1 for the list of supported profiles and packages.

7.6 Methods of the Profile

7.6.1 Extrinsic Methods of the Profile

7.6.1.1 Overview

The methods supported by this subprofile are summarized in Table 127, and detailed in the sections that follow it.

Table 127 - 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

7.6.1.2 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"}

Note: In this version of the 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: 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 64K bytes. 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>
```

7.6.1.3 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" }

7.6.1.4 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

7.6.1.5 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" }

7.6.2 Intrinsic Methods of the Profile

Note: Basic Write intrinsic methods are not specified for StatisticsCollection, HostedCollection, BlockStorageStatisticalData, MemberOfCollection or ElementStatisticalData.

7.6.2.1 DeleteInstance (of a CIM_BlockStatisticsManifestCollection)

This 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).

7.6.2.2 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

7.6.2.3 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.

7.6.2.4 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.

7.6.2.5 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.

7.6.2.6 EXEC QUERY

This is one of the ways of retrieving statistics.

7.6.2.7 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

7.7 Client Considerations and Recipes

7.7.1 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

7.7.2 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
// know 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(
               $DisExtents[#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

7.7.3 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
            }
        }
```

7.7.4 Summary of Statistics Support by Element

Not all statistics properties are kept for all elements. Table 128 illustrates the statistics properties that are kept for each of the metered elements.

Table 128 - 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

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
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 (SCSIArbitraryLogicalUnits) and RemoteReplicaGroup are not shown in Table 128: 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 an 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.

7.7.5 Formulas and Calculations

Table 128 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 129 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 129 - Formulas and Calculations

Calculated Statistics	
New statistic	**Formula**
TimeInterval	delta StatisticTime
% utilization	100 * (delta StatisticTime - delta IdleTime)/ delta StatisticTime
I/O rate	delta TotalIOs / delta StatisticTime
I/O response time	delta IOTime / delta TotalIOs
Queue depth	delta I/O rate * delta I/O response time
Service Time	utilization / I/O rate
Wait Time	Response Time - Service Time
Average Read Size	delta KBytesRead / delta ReadIOs
Average Write Size	delta KBytesWritten / delta WriteIOs
% Read	100 * (delta ReadIOs / delta TotalIOs)
% Write	100 * (delta WriteIOs / delta TotalIOs)
% Hit	100 * ((delta ReadHitIOs + delta WriteHitIOs) / delta TotalIOs)

7.7.6 Block Server Performance Supported Capabilities Patterns

The Capabilities patterns summarized in Table 130 are formally recognized by the Block Server Performance Subprofile of the current version of SMI-S

Table 130 - Block Server Performance Subprofile Supported Capabilities Patterns

ElementSupported	SynchronousMethods Supported	AsynchronousMethods Supported
Any (at least one)	NULL	NULL
Any (at least one)	Neither GettatisticsCollection 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).

7.7.7 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 section describes which metrics can be correlated between block storage statistics and port statistics.

7.8 CIM Elements

Table 131 describes the CIM elements for Block Server Performance.

Table 131 - CIM Elements for Block Server Performance

Element Name	Requirement	Description
7.8.1 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.
7.8.2 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.
7.8.3 CIM_BlockStatisticsCapabilities	Mandatory	This defines the statistics capabilities supported by the implementation of the profile.
7.8.4 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.
7.8.5 CIM_BlockStatisticsManifest (Provider Support)	Mandatory	An instance of this class defines the statistics properties supported by the profile implementation for one element type.
7.8.6 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).
7.8.7 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).
7.8.8 CIM_BlockStatisticsService	Mandatory	This is a Service that provides (optional) services of bulk statistics retrieval and manifest set manipulation methods.
7.8.9 CIM_BlockStorageStatisticalData	Mandatory	This is a Subclass of CIM_StatisticalData for Block servers. It would be instantiated as specific block statistics for particular components.
7.8.10 CIM_ElementCapabilities	Mandatory	This associates the BlockStatisticsCapabilities to the BlockStatisticsService.

Table 131 - CIM Elements for Block Server Performance

Element Name	Requirement	Description
7.8.11 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.
7.8.12 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.
7.8.13 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.
7.8.14 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.
7.8.15 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.
7.8.16 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 131 - CIM Elements for Block Server Performance

Element Name	Requirement	Description
7.8.17 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.
7.8.18 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.
7.8.19 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.
7.8.20 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).
7.8.21 CIM_HostedCollection (Default)	Mandatory	This would associate a default BlockStatisticsManifestCollection to the top level system for the profile (e.g., array).
7.8.22 CIM_HostedCollection (Provider Supplied)	Mandatory	This would associate the StatisticsCollection to the top level system for the profile (e.g., array).
7.8.23 CIM_HostedService	Mandatory	This associates the BlockStatisticsService to the ComputerSystem that hosts it.

Table 131 - CIM Elements for Block Server Performance

Element Name	Requirement	Description
7.8.24 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.
7.8.25 CIM_MemberOfCollection (Member of pre-defined collection)	Mandatory	This would associate pre-defined Manifests to default manifest collection.
7.8.26 CIM_MemberOfCollection (Member of statistics collection)	Mandatory	This would associate all block statistics instances to the StatisticsCollection.
7.8.27 CIM_StatisticsCollection	Mandatory	This would be a collection point for all Statistics that are kept for a Block Server.
7.8.28 SNIA_BlockStatisticsCapabilities	Optional	Experimental. This is a subclass of CIM_BlockStatisticsCapabilities that adds the SupportedFeatures property.
7.8.29 SNIA_BlockStatisticsManifest (Client Defined)	Optional	Experimental. This is a subclass of CIM_BlockStatisticsManifest that adds the CSVSequence property.
7.8.30 SNIA_BlockStatisticsManifest (Provider Support)	Optional	Experimental. This is a subclass of CIM_BlockStatisticsManifest that adds the CSVSequence property.

7.8.1 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 132 describes class CIM_AssociatedBlockStatisticsManifestCollection (Client defined collection).

Table 132 - SMI Referenced Properties/Methods for CIM_AssociatedBlockStatisticsManifestCollection (Client defined collection)

Properties	Flags	Requirement	Description & Notes
Statistics		Mandatory	The StatisticsCollection to which the manifest collection applies.
ManifestCollection		Mandatory	A client defined manifest collection.

7.8.2 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 133 describes class CIM_AssociatedBlockStatisticsManifestCollection (Provider defined collection).

Table 133 - SMI Referenced Properties/Methods for CIM_AssociatedBlockStatisticsManifestCollection (Provider defined collection)

Properties	Flags	Requirement	Description & Notes
Statistics		Mandatory	The StatisticsCollection to which the manifest collection applies.
ManifestCollection		Mandatory	The default manifest collection.

7.8.3 CIM_BlockStatisticsCapabilities

An instance of the CIM_BlockStatisticsCapabilities class defines the specific support provided with the block statistics implementation. Note: 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 134 describes class CIM_BlockStatisticsCapabilities.

Table 134 - SMI Referenced Properties/Methods for CIM_BlockStatisticsCapabilities

Properties	Flags	Requirement	Description & 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	N	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 microseconds.
Caption	N	Optional	Not Specified in this version of the Profile.
Description	N	Optional	Not Specified in this version of the Profile.
CreateGoalSettings()		Optional	Not Specified in this version of the Profile.

7.8.4 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 must 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 135 describes class CIM_BlockStatisticsManifest (Client Defined).

Table 135 - SMI Referenced Properties/Methods for CIM_BlockStatisticsManifest (Client Defined)

Properties	Flags	Requirement	Description & 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	
IncludeKBytesRead		Mandatory	
IncludeWriteIOs		Mandatory	
IncludeWriteHitIOs		Mandatory	
IncludeWriteIOTimeCounter		Mandatory	
IncludeWriteHitIOTimeCounter		Mandatory	
IncludeKBytesWritten		Mandatory	

Table 135 - SMI Referenced Properties/Methods for CIM_BlockStatisticsManifest (Client Defined)

Properties	Flags	Requirement	Description & Notes
IncludeIdleTimeCounter		Mandatory	
IncludeMaintOp		Mandatory	
IncludeMaintTimeCounter		Mandatory	
Caption	N	Optional	Not Specified in this version of the Profile.
Description	N	Optional	Not Specified in this version of the Profile.
IncludeStartStatisticTime	N	Optional	Not Specified in this version of the Profile.

7.8.5 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 136 describes class CIM_BlockStatisticsManifest (Provider Support).

Table 136 - SMI Referenced Properties/Methods for CIM_BlockStatisticsManifest (Provider Support)

Properties	Flags	Requirement	Description & 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.

Table 136 - SMI Referenced Properties/Methods for CIM_BlockStatisticsManifest (Provider Support)

Properties	Flags	Requirement	Description & Notes
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	
IncludeKBytesRead		Mandatory	
IncludeWriteIOs		Mandatory	
IncludeWriteHitIOs		Mandatory	
IncludeWriteIOTimeCounter		Mandatory	
IncludeWriteHitIOTimeCounter		Mandatory	
IncludeKBytesWritten		Mandatory	
IncludeIdleTimeCounter		Mandatory	
IncludeMaintOp		Mandatory	
IncludeMaintTimeCounter		Mandatory	
Caption	N	Optional	Not Specified in this version of the Profile.

Table 136 - SMI Referenced Properties/Methods for CIM_BlockStatisticsManifest (Provider Support)

Properties	Flags	Requirement	Description & Notes
Description	N	Optional	Not Specified in this version of the Profile.
IncludeStartStatisticTime	N	Optional	Not Specified in this version of the Profile.

7.8.6 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 defined 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 must 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 137 describes class CIM_BlockStatisticsManifestCollection (Client Defined).

Table 137 - SMI Referenced Properties/Methods for CIM_BlockStatisticsManifestCollection (Client Defined)

Properties	Flags	Requirement	Description & 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	N	Optional	Not Specified in this version of the Profile.
Description	N	Optional	Not Specified in this version of the Profile.

7.8.7 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 138 describes class CIM_BlockStatisticsManifestCollection (Provider Defined).

Table 138 - SMI Referenced Properties/Methods for CIM_BlockStatisticsManifestCollection (Provider Defined)

Properties	Flags	Requirement	Description & 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	N	Optional	Not Specified in this version of the Profile.
Description	N	Optional	Not Specified in this version of the Profile.

7.8.8 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 139 describes class CIM_BlockStatisticsService.

Table 139 - SMI Referenced Properties/Methods for CIM_BlockStatisticsService

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
Name		Mandatory	
Caption	N	Optional	Not Specified in this version of the Profile.
Description	N	Optional	Not Specified in this version of the Profile.
ElementName	N	Optional	Not Specified in this version of the Profile.
OperationalStatus	N	Optional	Not Specified in this version of the Profile.
StatusDescriptions	N	Optional	Not Specified in this version of the Profile.
InstallDate	N	Optional	Not Specified in this version of the Profile.
HealthState	N	Optional	Not Specified in this version of the Profile.
TimeOfLastStateChange	N	Optional	Not Specified in this version of the Profile.
OtherEnabledState	N	Optional	Not Specified in this version of the Profile.
EnabledDefault	N	Optional	Not Specified in this version of the Profile.
RequestedState	N	Optional	Not Specified in this version of the Profile.
EnabledState	N	Optional	Not Specified in this version of the Profile.
Started	N	Optional	Not Specified in this version of the Profile.
PrimaryOwnerName	N	Optional	Not Specified in this version of the Profile.
PrimaryOwnerContact	N	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 139 - SMI Referenced Properties/Methods for CIM_BlockStatisticsService

Properties	Flags	Requirement	Description & 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.

7.8.9 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 140 describes class CIM_BlockStorageStatisticalData.

Table 140 - SMI Referenced Properties/Methods for CIM_BlockStorageStatisticalData

Properties	Flags	Requirement	Description & 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: 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: This is not SPECIFIED for CompositeExtents, ArbitraryLUs or Remote Replica Groups..

Table 140 - SMI Referenced Properties/Methods for CIM_BlockStorageStatisticalData

Properties	Flags	Requirement	Description & Notes
ReadIOs		Conditional	Conditional requirement: This property is required if the ElementType is 3, 4, 8 or 10. The cumulative count of all reads. Note: 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: This is not specified for Ports, CompositeExtents, "Back-end" component computer systems, ArbitraryLUs or Remote Replica Groups..
ReadHitIOs		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: This is mandatory for "Front-end" and "Peer" component ComputerSystems, and Volumes, but it is optional for the Top level computer system. Note: This is not specified for "Back-end" component computer systems, Ports, CompositeExtents, DiskDrives, ArbitraryLUs or Remote Replica Groups.
ReadIOTimeCounter		Optional	The cumulative elapsed time for all Read I/Os) for all cumulative Read I/Os. Note: This is optional for "Front-end" and "Peer" component ComputerSystems and the Top level computer system, Volumes and Disk Drives. Note: 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: This is optional for "Front-end" and "Peer" component ComputerSystems and the Top level computer system and Volumes. Note: 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 (1024bytes = 1KByte). Note: This is optional for all ComputerSystems, Volumes, and Disk Drives. Note: This is not specified for Ports, CompositeExtents, ArbitraryLUs or Remote Replica Groups..

Table 140 - SMI Referenced Properties/Methods for CIM_BlockStorageStatisticalData

Properties	Flags	Requirement	Description & Notes
WriteIOs		Conditional	Conditional requirement: This property is required if the ElementType is 3, 4 or 8. The cumulative count of all writes. Note: 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: 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: This is mandatory for "Front-end" and "Peer" component ComputerSystems and Volumes, but it is optional for the Top level computer system. Note: 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: This is optional for "Front-end" and "Peer" component ComputerSystems and the Top level computer system and Volumes and Disks Drives. Note: This is not specified for "Back-end" component computer systems, Ports, CompositeExtents, ArbitraryLUs or Remote Replica Groups.
WriteHitIOTimeCounter		Optional	The cumulative elapsed time for all Write I/Os written to cache for all cumulative Write I/Os. Note: This is optional for "Front-end" and "Peer" component ComputerSystems and the Top level computer system and Volumes. Note: 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 (1024bytes = 1KByte). Note: This is optional for all ComputerSystems, Volumes and Disk Drives. Note: This is not specified for Ports, CompositeExtents, ArbitraryLUs or Remote Replica Groups.

Table 140 - SMI Referenced Properties/Methods for CIM_BlockStorageStatisticalData

Properties	Flags	Requirement	Description & Notes
IdleTimeCounter		Optional	The cumulative elapsed idle time using ClockTickInterval units (Cumulative Number of Time Units for all idle time in the array). Note: This is optional for "Back-end" component ComputerSystems, Front end Ports, Volumes, Extents and Disk Drives. Note: 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: This is optional for Extents and Disk Drives. Note: 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: This is optional for Extents and Disk Drives. Note: This is not specified for ComputerSystems, Ports, Volumes, ArbitraryLUs or Remote Replica Groups.
Caption	N	Optional	Not Specified in this version of the Profile.
Description	N	Optional	Not Specified in this version of the Profile.
ElementName	N	Optional	Not Specified in this version of the Profile.
StartStatisticTime	N	Optional	Not Specified in this version of the Profile.
ResetSelectedStats()		Optional	Not Specified in this version of the Profile.

7.8.10 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 141 describes class CIM_ElementCapabilities.

Table 141 - SMI Referenced Properties/Methods for CIM_ElementCapabilities

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	The managed element (BlockStatisticsService).
Capabilities		Mandatory	The Capabilities instance associated with the BlockStatisticsService.

7.8.11 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 142 describes class CIM_ElementStatisticalData (Back end Port Stats).

Table 142 - SMI Referenced Properties/Methods for CIM_ElementStatisticalData (Back end Port Stats)

Properties	Flags	Requirement	Description & 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.

7.8.12 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 143 describes class CIM_ElementStatisticalData (Component System Stats).

Table 143 - SMI Referenced Properties/Methods for CIM_ElementStatisticalData (Component System Stats)

Properties	Flags	Requirement	Description & 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.

7.8.13 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 144 describes class CIM_ElementStatisticalData (Disk Stats).

Table 144 - SMI Referenced Properties/Methods for CIM_ElementStatisticalData (Disk Stats)

Properties	Flags	Requirement	Description & 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.

7.8.14 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 145 describes class CIM_ElementStatisticalData (Extent Stats).

Table 145 - SMI Referenced Properties/Methods for CIM_ElementStatisticalData (Extent Stats)

Properties	Flags	Requirement	Description & 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.

7.8.15 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 146 describes class CIM_ElementStatisticalData (Front end Port Stats).

Table 146 - SMI Referenced Properties/Methods for CIM_ElementStatisticalData (Front end Port Stats)

Properties	Flags	Requirement	Description & 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.

7.8.16 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 147 describes class CIM_ElementStatisticalData (Logical Disk Stats).

Table 147 - SMI Referenced Properties/Methods for CIM_ElementStatisticalData (Logical Disk Stats)

Properties	Flags	Requirement	Description & 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.

7.8.17 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 148 describes class CIM_ElementStatisticalData (Remote Copy Stats).

Table 148 - SMI Referenced Properties/Methods for CIM_ElementStatisticalData (Remote Copy Stats)

Properties	Flags	Requirement	Description & 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.

7.8.18 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 149 describes class CIM_ElementStatisticalData (Top Level System Stats).

Table 149 - SMI Referenced Properties/Methods for CIM_ElementStatisticalData (Top Level System Stats)

Properties	Flags	Requirement	Description & 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.

7.8.19 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 150 describes class CIM_ElementStatisticalData (Volume Stats).

Table 150 - SMI Referenced Properties/Methods for CIM_ElementStatisticalData (Volume Stats)

Properties	Flags	Requirement	Description & 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.

7.8.20 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 151 describes class CIM_HostedCollection (Client Defined).

Table 151 - SMI Referenced Properties/Methods for CIM_HostedCollection (Client Defined)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	The top level ComputerSystem of the profile.
Dependent		Mandatory	A client defined BlockStatisticsManifestCollection.

7.8.21 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 152 describes class CIM_HostedCollection (Default).

Table 152 - SMI Referenced Properties/Methods for CIM_HostedCollection (Default)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	The top level ComputerSystem of the profile.
Dependent		Mandatory	The provider defined BlockStatisticsManifestCollection.

7.8.22 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 153 describes class CIM_HostedCollection (Provider Supplied).

Table 153 - SMI Referenced Properties/Methods for CIM_HostedCollection (Provider Supplied)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	The top level ComputerSystem of the profile.
Dependent		Mandatory	The StatisticsCollection.

7.8.23 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 154 describes class CIM_HostedService.

Table 154 - SMI Referenced Properties/Methods for CIM_HostedService

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	The hosting System.
Dependent		Mandatory	The Service hosted on the System.

7.8.24 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 155 describes class CIM_MemberOfCollection (Member of client defined collection).

Table 155 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Member of client defined collection)

Properties	Flags	Requirement	Description & Notes
Collection		Mandatory	A client defined manifest collection.
Member		Mandatory	The individual Manifest Instance that is part of the set.

7.8.25 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 156 describes class CIM_MemberOfCollection (Member of pre-defined collection).

Table 156 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Member of pre-defined collection)

Properties	Flags	Requirement	Description & Notes
Collection		Mandatory	The provider defined default manifest collection.
Member		Mandatory	The individual Manifest Instance that is part of the set.

7.8.26 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 157 describes class CIM_MemberOfCollection (Member of statistics collection).

Table 157 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Member of statistics collection)

Properties	Flags	Requirement	Description & Notes
Collection		Mandatory	The default manifest collection.
Member		Mandatory	The individual Manifest Instance that is part of the set.

7.8.27 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 158 describes class CIM_StatisticsCollection.

Table 158 - SMI Referenced Properties/Methods for CIM_StatisticsCollection

Properties	Flags	Requirement	Description & 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.
TimeLastSampled		Mandatory	Time statistics table by object was last updated (Time Stamp in SMI 2.2 specification format).
Caption	N	Optional	Not Specified in this version of the Profile.
Description	N	Optional	Not Specified in this version of the Profile.

7.8.28 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 159 describes class SNIA_BlockStatisticsCapabilities.

Table 159 - SMI Referenced Properties/Methods for SNIA_BlockStatisticsCapabilities

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
ElementName		Mandatory	
ElementTypesSupported		Mandatory	
SynchronousMethodsSupported	N	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).

7.8.29 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 160 describes class SNIA_BlockStatisticsManifest (Client Defined).

Table 160 - SMI Referenced Properties/Methods for SNIA_BlockStatisticsManifest (Client Defined)

Properties	Flags	Requirement	Description & 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	

Table 160 - SMI Referenced Properties/Methods for SNIA_BlockStatisticsManifest (Client Defined)

Properties	Flags	Requirement	Description & Notes
IncludeMaintTimeCounter		Mandatory	
CSVSequence	N	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.

7.8.30 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 161 describes class SNIA_BlockStatisticsManifest (Provider Support).

Table 161 - SMI Referenced Properties/Methods for SNIA_BlockStatisticsManifest (Provider Support)

Properties	Flags	Requirement	Description & Notes
ElementName		Mandatory	
InstanceID		Mandatory	
ElementType		Mandatory	
IncludeStatisticTime		Mandatory	
IncludeTotalIOs		Mandatory	
IncludeKBytesTransferred		Mandatory	
IncludeIOTimeCounter		Mandatory	
IncludeReadIOs		Mandatory	
IncludeReadHitIOs		Mandatory	
IncludeReadIOTimeCounter		Mandatory	

Table 161 - SMI Referenced Properties/Methods for SNIA_BlockStatisticsManifest (Provider Support)

Properties	Flags	Requirement	Description & Notes
IncludeReadHitIOTimeCounter		Mandatory	
IncludeKBytesRead		Mandatory	
IncludeWriteIOs		Mandatory	
IncludeWriteHitIOs		Mandatory	
IncludeWriteIOTimeCounter		Mandatory	
IncludeWriteHitIOTimeCounter		Mandatory	
IncludeKBytesWritten		Mandatory	
IncludeIdleTimeCounter		Mandatory	
IncludeMaintOp		Mandatory	
IncludeMaintTimeCounter		Mandatory	
CSVSequence	N	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

Clause 8: CKD Block Services Profile

8.1 Description

8.1.1 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

8.1.2 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.

8.1.3 Implementation

8.1.3.1 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 46: "Block Services Support for Count Key Data Storage".

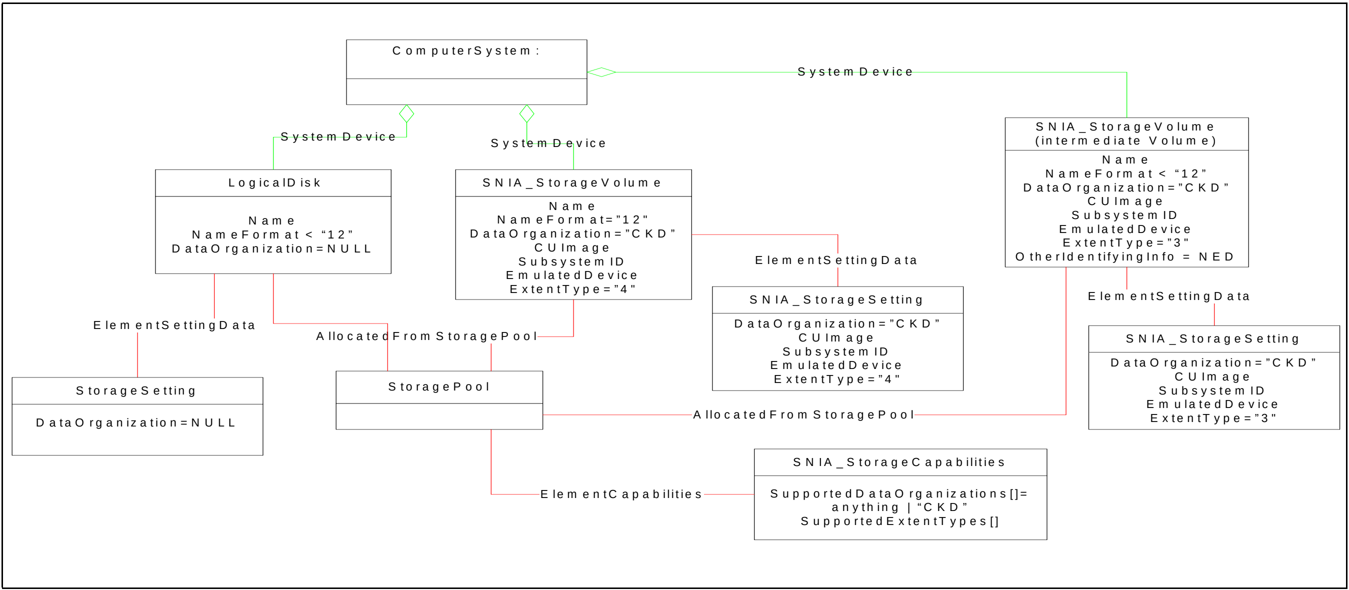

Figure 46 - Block Services Support for Count Key Data Storage

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 specification. 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 CUImage (ProtocolController). This is accommodated by the CUImage property on StorageSetting. That is, on creation of an CKD Volume the CUImage parameter is passed as part of the StorageSetting for the Volume being created. The CUImage in the SNIA_StorageSetting is the CUImage requested and the CUImage in the SNIA_StorageVolume is the CUImage assigned. CUImage 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.

8.1.3.2 Use Cases for CKD Storage

8.1.3.2.1 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.

8.1.3.2.2 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.

8.1.3.2.3 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 CUImage set to a valid CUImage value.

With the appropriate CKD Volume Setting the client would issue either CreateOrModifyElementFromStoragePool or CreateOrModifyElementFromElements.

8.2 Health and Fault Management Consideration

No change for CKD.

8.3 Cascading Considerations

No change for CKD.

8.4 Supported Profiles, Subprofiles, and Packages

Table 162 describes the supported profiles for CKD Block Services.

Table 162 - Supported Profiles for CKD Block Services

Profile Name	Organization	Version	Requirement	Description
Job Control	SNIA	1.5.0	Optional	
Extent Composition	SNIA	1.6.0	Optional	
Block Services	SNIA	1.6.0	Mandatory	

8.5 Methods of the Profile

All methods of the Block Services Package should work for CKD storage (subject to restrictions of particular profile implementations).

8.6 Client Considerations and Recipes

No change for CKD.

8.7 Registered Name and Version

CKD Block Services version 1.3.0 (Component Profile)

CIM Schema Version: 2.13

Specializes SNIA Block Services version 1.6.0

8.8 CIM Elements

Table 163 describes the CIM elements for CKD Block Services.

Table 163 - CIM Elements for CKD Block Services

Element Name	Requirement	Description
8.8.1 CIM_AllocatedFromStoragePool	Mandatory	The CIM_AllocatedFromStoragePool associates a Storage Element (Volume, LogicalDisk or StoragePool) to its parent StoragePool. There are no enhancements for CKD.
8.8.2 CIM_AllocatedFromStoragePool (Pool from Pool)	Mandatory	AllocatedFromStoragePool.
8.8.3 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.
8.8.4 CIM_ElementCapabilities	Mandatory	The CIM_ElementCapabilities associates a StoragePool or StorageConfigurationService to its Capabilities (StorageCapabilities and StorageConfigurationCapabilities). There are no enhancements for CKD.
8.8.5 CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StorageVolume or LogicalDisk)	Optional	Expressed the ability for the element to be named or have its state changed.
8.8.6 CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StoragePool)	Optional	Expressed the ability for the element to be named or have its state changed.
8.8.7 CIM_ElementCapabilities (StorageCapabilities to StorageConfigurationService)	Optional	Associates StorageCapabilities with StorageConfigurationService. This StorageCapabilities shall represent the capabilities of the entire implementation.
8.8.8 CIM_ElementCapabilities (StorageCapabilities to StoragePool)	Mandatory	Associates StorageCapabilities with StoragePool. This StorageCapabilities shall represent the capabilities of the StoragePool to which it is associated.
8.8.9 CIM_ElementCapabilities (StorageConfigurationCapabilities to StorageConfigurationService)	Mandatory	Associates StorageConfigurationCapabilities with StorageConfigurationService.
8.8.10 CIM_ElementCapabilities (StorageConfigurationCapabilities to concrete StoragePool)	Optional	Associates StorageConfigurationCapabilities with StoragePool.

Table 163 - CIM Elements for CKD Block Services

Element Name	Requirement	Description
8.8.11 CIM_ElementCapabilities (StorageConfigurationCapabilities to primordial StoragePool)	Optional	Associates StorageConfigurationCapabilities with StoragePool.
8.8.12 CIM_ElementCapabilities (Used to declare the naming capabilities of the StoragePool)	Optional	Associates EnabledLogicalElementCapabilities with StorageConfigurationService.
8.8.13 CIM_ElementCapabilities (Used to declare the naming capabilities of the StorageVolume or LogicalDisk)	Optional	Associates EnabledLogicalElementCapabilities with StorageConfigurationService.
8.8.14 CIM_ElementSettingData	Mandatory	The CIM_ElementSettingData associates a StorageVolume (or LogicalDisk) to its StorageSetting. There are no enhancements for CKD.
8.8.15 CIM_EnabledLogicalElementCapabilities (For StorageConfigurationService)	Optional	This class is used to express the naming and possible requested state change possibilities for storage elements.
8.8.16 CIM_EnabledLogicalElementCapabilities (For StoragePool)	Optional	This class is used to express the naming and possible requested state change possibilities for storage pools.
8.8.17 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.
8.8.18 CIM_FilterCollection (Block Services ProfileSpecificLifecycleIndicationFilterCollection)	Optional	Experimental. This is a collection of Lifecycle IndicationFilters defined by the Block Services Profile.
8.8.19 CIM_HostedCollection (Block Services to ProfileSpecificLifecycleIndicationFilterCollection)	Optional	Experimental.
8.8.20 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).
8.8.21 CIM_HostedService	Optional	This associates the StorageConfigurationService to its scoping system. This is unchanged for CKD storage.
8.8.22 CIM_HostedStoragePool	Mandatory	

Table 163 - CIM Elements for CKD Block Services

Element Name	Requirement	Description
8.8.23 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.
8.8.24 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.
8.8.25 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.
8.8.26 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.
8.8.27 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.
8.8.28 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.

Table 163 - CIM Elements for CKD Block Services

Element Name	Requirement	Description
8.8.29 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.
8.8.30 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.
8.8.31 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.
8.8.32 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.
8.8.33 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.
8.8.34 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.

Table 163 - CIM Elements for CKD Block Services

Element Name	Requirement	Description
8.8.35 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).
8.8.36 CIM_MemberOfCollection (Block Services ProfileSpecificLifecycleIndicationFilterCollection to Block Services Filters)	Optional	Experimental. This associates the Block Services ProfileSpecificLifecycleIndicationFilterCollection to the Filters defined by the Block Services Profile.
8.8.37 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.
8.8.38 CIM_OwningJobElement	Conditional	Conditional requirement: Support for Job Control profile.
8.8.39 CIM_StorageConfigurationCapabilities	Optional	These Capabilities define the capabilities provided by the CIM_StorageConfigurationService.
8.8.40 CIM_StorageConfigurationCapabilities (Concrete)	Conditional	Conditional requirement: Support for the Storage Relocation profile.
8.8.41 CIM_StorageConfigurationCapabilities (Global)	Conditional	Conditional requirement: Support for StorageConfigurationService.
8.8.42 CIM_StorageConfigurationCapabilities (Primordial)	Conditional	Conditional requirement: Support for the Storage Relocation profile.
8.8.43 CIM_StorageConfigurationService	Optional	This service provides method for volume and pool manipulation. There are no enhancements for CKD storage.
8.8.44 CIM_StoragePool	Mandatory	Primordial and Concrete Pools. These are unchanged for CKD storage.
8.8.45 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.

Table 163 - CIM Elements for CKD Block Services

Element Name	Requirement	Description
8.8.46 CIM_StoragePool (Empty)	Optional	An empty StoragePool is a special case of a StoragePool (Concrete or Primordial) where the StoragePool contains no capacity.
8.8.47 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.
8.8.48 CIM_StorageSettingWithHints	Optional	These are hints that can be added to StorageSetting. There are no enhancements for CKD.
8.8.49 CIM_StorageSettingsAssociatedToCapabilitie s	Optional	This class associates the StorageCapabilities with the pre-defined setting. There are no enhancements for CKD.
8.8.50 CIM_StorageSettingsGeneratedFromCapabilit ies	Optional	This class associates the StorageCapabilities with the StorageSetting generated from it via the CreateSetting method. There are no enhancements for CKD.
8.8.51 CIM_SystemDevice (System to StorageVolume or LogicalDisk)	Mandatory	Associates top level system from Array, Virtualizer, ... to StorageVolume or LogicalDisk.
8.8.52 SNIA_StorageCapabilities	Mandatory	These Capabilities define the capabilities provided by a CIM_StoragePool. This includes the capability to support SCSI and/or CKD storage.
8.8.53 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.
8.8.54 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.
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_StoragePool	Mandatory	Creation/Deletion of StoragePool. See section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.24 CIM_IndicationFilter (Storage Pool Creation).
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_StoragePool	Mandatory	Deletion of StoragePool. See section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.25 CIM_IndicationFilter (Storage Pool Deletion).

Table 163 - CIM Elements for CKD Block Services

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 section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.27 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 section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.28 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 section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.31 CIM_IndicationFilter (WQL Storage Volume OperationalStatus).
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 section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.29 CIM_IndicationFilter (Storage Volume OperationalStatus).

Table 163 - CIM Elements for CKD Block Services

Element Name	Requirement	Description
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 section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.21 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 section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.22 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 section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.30 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 section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.23 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 section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.26 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.

Table 163 - CIM Elements for CKD Block Services

Element Name	Requirement	Description
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.
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. Deprecated 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.

8.8.1 CIM_AllocatedFromStoragePool

This class is unchanged from the Block Services Package.

Requirement: Mandatory

8.8.2 CIM_AllocatedFromStoragePool (Pool from Pool)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 164 describes class CIM_AllocatedFromStoragePool (Pool from Pool).

Table 164 - SMI Referenced Properties/Methods for CIM_AllocatedFromStoragePool (Pool from Pool)

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

8.8.3 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 165 describes class CIM_AllocatedFromStoragePool (Volume or LogicalDisk from Pool).

Table 165 - SMI Referenced Properties/Methods for CIM_AllocatedFromStoragePool (Volume or LogicalDisk from Pool)

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

8.8.4 CIM_ElementCapabilities

This class is unchanged from the Block Services Package.

Requirement: Mandatory

8.8.5 CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StorageVolume or LogicalDisk)

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

Table 166 describes class CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StorageVolume or LogicalDisk).

Table 166 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StorageVolume or LogicalDisk)

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

8.8.6 CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StoragePool)

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

Table 167 describes class CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StoragePool).

Table 167 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StoragePool)

Properties	Flags	Requirement	Description & 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.

8.8.7 CIM_ElementCapabilities (StorageCapabilities to StorageConfigurationService)

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

Table 168 describes class CIM_ElementCapabilities (StorageCapabilities to StorageConfigurationService).

Table 168 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (StorageCapabilities to StorageConfigurationService)

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

8.8.8 CIM_ElementCapabilities (StorageCapabilities to StoragePool)

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

Table 169 describes class CIM_ElementCapabilities (StorageCapabilities to StoragePool).

Table 169 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (StorageCapabilities to StoragePool)

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

8.8.9 CIM_ElementCapabilities (StorageConfigurationCapabilities to StorageConfigurationService)

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

Table 170 describes class CIM_ElementCapabilities (StorageConfigurationCapabilities to StorageConfigurationService).

Table 170 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (StorageConfigurationCapabilities to StorageConfigurationService)

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

8.8.10 CIM_ElementCapabilities (StorageConfigurationCapabilities to concrete StoragePool)

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

Table 171 describes class CIM_ElementCapabilities (StorageConfigurationCapabilities to concrete StoragePool).

Table 171 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (StorageConfigurationCapabilities to concrete StoragePool)

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

8.8.11 CIM_ElementCapabilities (StorageConfigurationCapabilities to primordial StoragePool)

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

Table 172 describes class CIM_ElementCapabilities (StorageConfigurationCapabilities to primordial StoragePool).

Table 172 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (StorageConfigurationCapabilities to primordial StoragePool)

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

8.8.12 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 173 describes class CIM_ElementCapabilities (Used to declare the naming capabilities of the StoragePool).

Table 173 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (Used to declare the naming capabilities of the StoragePool)

Properties	Flags	Requirement	Description & 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.

8.8.13 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 174 describes class CIM_ElementCapabilities (Used to declare the naming capabilities of the StorageVolume or LogicalDisk).

Table 174 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (Used to declare the naming capabilities of the StorageVolume or LogicalDisk)

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

8.8.14 CIM_ElementSettingData

This class is unchanged from the Block Services Package.

Requirement: Mandatory

8.8.15 CIM_EnabledLogicalElementCapabilities (For StorageConfigurationService)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 175 describes class CIM_EnabledLogicalElementCapabilities (For StorageConfigurationService).

Table 175 - SMI Referenced Properties/Methods for CIM_EnabledLogicalElementCapabilities (For StorageConfigurationService)

Properties	Flags	Requirement	Description & 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 Capacilities.
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.

8.8.16 CIM_EnabledLogicalElementCapabilities (For StoragePool)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 176 describes class CIM_EnabledLogicalElementCapabilities (For StoragePool).

Table 176 - SMI Referenced Properties/Methods for CIM_EnabledLogicalElementCapabilities (For StoragePool)

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
ElementName		Mandatory	For this usage of the capabilities this should be 'StoragePool Enabled Capabilities'.

Table 176 - SMI Referenced Properties/Methods for CIM_EnabledLogicalElementCapabilities (For StoragePool)

Properties	Flags	Requirement	Description & Notes
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.

8.8.17 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 177 describes class CIM_FilterCollection (Block Services Predefined FilterCollection).

Table 177 - SMI Referenced Properties/Methods for CIM_FilterCollection (Block Services Predefined FilterCollection)

Properties	Flags	Requirement	Description & 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:Predefined'.

8.8.18 CIM_FilterCollection (Block Services ProfileSpecificLifecycleIndicationFilterCollection)

Experimental. This is a collection of Lifecycle IndicationFilters defined by the Block Services Profile.

Requirement: Optional

Table 178 describes class CIM_FilterCollection (Block Services ProfileSpecificLifecycleIndicationFilterCollection).

Table 178 - SMI Referenced Properties/Methods for CIM_FilterCollection (Block Services Profile-SpecificLifecycleIndicationFilterCollection)

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	See the InstanceID definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.13 CIM_FilterCollection (StaticFilterCollection).
CollectionName		Mandatory	The value of CollectionName shall be 'SNIA:Block Services:ProfileSpecificLifecycleIndicationFilterCollection'.

8.8.19 CIM_HostedCollection (Block Services to ProfileSpecificLifecycleIndicationFilterCollection)

Experimental.

Requirement: Optional

Table 179 describes class CIM_HostedCollection (Block Services to ProfileSpecificLifecycleIndicationFilterCollection).

Table 179 - SMI Referenced Properties/Methods for CIM_HostedCollection (Block Services to ProfileSpecificLifecycleIndicationFilterCollection)

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	Reference to the ProfileSpecificLifecycleIndicationFilterCollection for the Block Services Profile.
Antecedent		Mandatory	Reference to the 'Top level' System.

8.8.20 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 180 describes class CIM_HostedCollection (System to predefined IndicationFilters).

Table 180 - SMI Referenced Properties/Methods for CIM_HostedCollection (System to predefined IndicationFilters)

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

8.8.21 CIM_HostedService

This class is unchanged from the Block Services Package.

Requirement: Optional

8.8.22 CIM_HostedStoragePool

Requirement: Mandatory

Table 181 describes class CIM_HostedStoragePool.

Table 181 - SMI Referenced Properties/Methods for CIM_HostedStoragePool

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

8.8.23 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 182 describes class CIM_IndicationFilter (Logical Disk Creation).

Table 182 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Logical Disk Creation)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Name		Mandatory	This shall be 'SNIA:Block Services:LogicalDiskCreation'.

Table 182 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Logical Disk Creation)

Properties	Flags	Requirement	Description & Notes
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
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 section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).

8.8.24 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 183 describes class CIM_IndicationFilter (Logical Disk Deletion).

Table 183 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Logical Disk Deletion)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).

Table 183 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Logical Disk Deletion)

Properties	Flags	Requirement	Description & Notes
Name		Mandatory	This shall be 'SNIA:Block Services:LogicalDiskDeletion'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
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 section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).

8.8.25 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 184 describes class CIM_IndicationFilter (Logical Disk OperationalStatus).

Table 184 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Logical Disk OperationalStatus)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).

Table 184 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Logical Disk OperationalStatus)

Properties	Flags	Requirement	Description & Notes
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Name		Mandatory	This shall be 'SNIA:Block Services:LogicalDiskOperationalStatus'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
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 section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).

8.8.26 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 185 describes class CIM_IndicationFilter (Storage Pool Creation).

Table 185 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Storage Pool Creation)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Name		Mandatory	This shall be 'SNIA:Block Services:StoragePoolCreation'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
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 section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).

8.8.27 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 186 describes class CIM_IndicationFilter (Storage Pool Deletion).

Table 186 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Storage Pool Deletion)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Name		Mandatory	This shall be 'SNIA:Block Services:StoragePoolDeletion'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
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 section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).

8.8.28 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 187 describes class CIM_IndicationFilter (Storage Pool TotalManagedSpace).

Table 187 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Storage Pool TotalManagedSpace)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Name		Mandatory	This shall be 'SNIA:Block Services:StoragePoolTotalManagedSpace'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
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 section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).

8.8.29 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 188 describes class CIM_IndicationFilter (Storage Volume Creation).

Table 188 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Storage Volume Creation)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Name		Mandatory	This shall be 'SNIA:Block Services:StorageVolumeCreation'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
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 section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).

8.8.30 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 189 describes class CIM_IndicationFilter (Storage Volume Deletion).

Table 189 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Storage Volume Deletion)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Name		Mandatory	This shall be 'SNIA:Block Services:StorageVolumeDeletion'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
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 section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).

8.8.31 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 190 describes class CIM_IndicationFilter (Storage Volume OperationalStatus).

Table 190 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Storage Volume OperationalStatus)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Name		Mandatory	This shall be 'SNIA:Block Services:StorageVolumeOperationalStatus'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
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 section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).

8.8.32 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 191 describes class CIM_IndicationFilter (WQL Logical Disk OperationalStatus).

Table 191 - SMI Referenced Properties/Methods for CIM_IndicationFilter (WQL Logical Disk OperationalStatus)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Name		Mandatory	This shall be 'SNIA:Block Services:LogicalDiskOperationalStatusWQL'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
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 section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).

8.8.33 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 192 describes class CIM_IndicationFilter (WQL Storage Volume OperationalStatus).

Table 192 - SMI Referenced Properties/Methods for CIM_IndicationFilter (WQL Storage Volume OperationalStatus)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Name		Mandatory	This shall be 'SNIA:Block Services:StorageVolumeOperationalStatusWQL'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
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 section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).

8.8.34 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. EDITORIAL NOTE: Although there is no immediate need for a LogicalDisk to be CKD, this is likely to show up in SMI-S sometime. DataOrganization is defined on the StorageExtent, so I think it makes sense to show it as a property of LogicalDisk. The class definition specializes the CIM_LogicalDisk 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 Volume Management - LogicalDisk is mandatory.

Table 193 describes class CIM_LogicalDisk.

Table 193 - SMI Referenced Properties/Methods for CIM_LogicalDisk

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	Opaque identifier.
ElementName		Optional	User-friendly name.
Name		Mandatory	OS Device Name.
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). In addition, the secondary OperationalStatus may be 19 (Relocating) with 2|3|15 (OK or Degraded 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	

Table 193 - SMI Referenced Properties/Methods for CIM_LogicalDisk

Properties	Flags	Requirement	Description & Notes
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'.
NumExtentsMigrating		Optional	Experimental. The number of Extents in the process of migrating for this logical disk when the logical disk relocation is on going.
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".

8.8.35 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 194 describes class CIM_MemberOfCollection (Block Services Filter Collection to FilterCollection).

Table 194 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Block Services Filter Collection to FilterCollection)

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

8.8.36 CIM_MemberOfCollection (Block Services ProfileSpecificLifecycleIndicationFilterCollection to Block Services Filters)

Experimental. This associates the Block Services ProfileSpecificLifecycleIndicationFilterCollection to the Filters defined by the Block Services Profile.

Requirement: Optional

Table 195 describes class CIM_MemberOfCollection (Block Services ProfileSpecificLifecycleIndicationFilterCollection to Block Services Filters).

Table 195 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Block Services ProfileSpecificLifecycleIndicationFilterCollection to Block Services Filters)

Properties	Flags	Requirement	Description & Notes
Collection		Mandatory	Reference to the Block Services ProfileSpecificLifecycleIndicationFilterCollection.
Member		Mandatory	Reference to the IndicationFilters defined by the Block Services Profile.

8.8.37 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 196 describes class CIM_MemberOfCollection (Predefined Filter Collection to Block Services Filters).

Table 196 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Predefined Filter Collection to Block Services Filters)

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

8.8.38 CIM_OwningJobElement

Conditional on support for Job Control profile.

Requirement: Support for Job Control profile.

Table 197 describes class CIM_OwningJobElement.

Table 197 - SMI Referenced Properties/Methods for CIM_OwningJobElement

Properties	Flags	Requirement	Description & Notes
OwnedElement		Mandatory	
OwningElement		Mandatory	

8.8.39 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

8.8.40 CIM_StorageConfigurationCapabilities (Concrete)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Support for the Storage Relocation profile.

Table 198 describes class CIM_StorageConfigurationCapabilities (Concrete).

Table 198 - SMI Referenced Properties/Methods for CIM_StorageConfigurationCapabilities (Concrete)

Properties	Flags	Requirement	Description & 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		Conditional	Conditional requirement: Support for the Storage Relocation profile. Lists what actions, invoked through StorageConfigurationService methods, shall not produce Concrete jobs. This version of the 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) or "17" (StorageVolume To StoragePool Relocation) or "18" (StoragePool To StoragePool Relocation) or "19" (StorageVolume To StorageExtent Relocation) or "20" (StoragePool To StorageExtent Relocation) or "21" (LogicalDisk To StorageExtent Relocation) or "22" (Multiple Storage Element Creation) or "23" (Multiple Storage Element Return).
SupportedStorageElementTypes		Mandatory	Lists the type of storage elements that are supported by this implementation. This version of the standard recognizes '2' (StorageVolume) or '4' (LogicalDisk).

Table 198 - SMI Referenced Properties/Methods for CIM_StorageConfigurationCapabilities (Concrete)

Properties	Flags	Requirement	Description & Notes
SupportedAsynchronousActions		Conditional	Conditional requirement: Support for the Storage Relocation profile. Lists what actions, invoked through StorageConfigurationService methods, may produce Concrete jobs. This version of the 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) or "17" (StorageVolume To StoragePool Relocation) or "18" (StoragePool To StoragePool Relocation) or "19" (StorageVolume To StorageExtent Relocation) or "20" (StoragePool To StorageExtent Relocation) or "21" (LogicalDisk To StorageExtent Relocation) or "22" (Multiple Storage Element Creation) or "23" (Multiple Storage Element Return)..
SupportedStorageElementFeatures		Conditional	Conditional requirement: Support for the Storage Relocation profile. Lists actions supported through the invocation of StorageServiceService.CreateOrModifyElementFromStoragePool(). Matches 3|8|14|15|16|17|18 (StorageVolume Creation or LogicalDisk Creation or StorageVolume To StoragePool Relocation or StoragePool To StoragePool Relocation or StorageVolume To StorageExtent Relocation or StoragePool To StorageExtent Relocation LogicalDisk To StorageExtent Relocation).
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		Conditional	Conditional requirement: Support for the Storage Relocation profile. Indicates the intended usage or any restrictions that may have been imposed on storage pools.
ClientSettablePoolUsage		Conditional	Conditional requirement: Support for the Storage Relocation profile. Indicates the intended usage or any restrictions that may have been imposed on the usage of a client-settable storage pool.
MaximumElementCreateCount		Optional	Indicates the maximum number of elements that can be specified to be created in a single method call. If 0 or null, there is no limit.
MaximumElementDeleteCount		Conditional	Conditional requirement: Support for the Storage Relocation profile. Indicates the maximum number of elements that can be deleted in a single method call. If 0 or null, there is no limit.

Table 198 - SMI Referenced Properties/Methods for CIM_StorageConfigurationCapabilities (Concrete)

Properties	Flags	Requirement	Description & Notes
MultipleElementCreateFeatures		Optional	Enumeration indicating features offered by the multiple element create method. "2" (Single instance creation indication).
MultipleElementDeleteFeatures		Optional	Enumeration indicating features offered by the multiple element delete method. "2" (Continue on nonexistent element) or "3" (Return error on nonexistent element).

8.8.41 CIM_StorageConfigurationCapabilities (Global)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Support for StorageConfigurationService.

Table 199 describes class CIM_StorageConfigurationCapabilities (Global).

Table 199 - SMI Referenced Properties/Methods for CIM_StorageConfigurationCapabilities (Global)

Properties	Flags	Requirement	Description & 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		Conditional	Conditional requirement: Support for the Storage Relocation profile. 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		Conditional	Conditional requirement: Support for the Storage Relocation profile. Lists what actions, invoked through StorageConfigurationService methods, may produce Concrete jobs.

Table 199 - SMI Referenced Properties/Methods for CIM_StorageConfigurationCapabilities (Global)

Properties	Flags	Requirement	Description & Notes
SupportedStorageElementFeatures		Conditional	Conditional requirement: Support for the Storage Relocation profile. Lists actions supported through the invocation of StorageServiceService.CreateOrModifyElementFromStoragePool(). Matches 3|5|8|9|11|12|13|14|15|16|17|18 (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 or StorageVolume To StoragePool Relocation or StoragePool To StoragePool Relocation or StorageVolume To StorageExtent Relocation or 'StoragePool To StorageExtent Relocation or LogicalDisk To StorageExtent Relocation).
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		Conditional	Conditional requirement: Support for the Storage Relocation profile. Indicates the intended usage or any restrictions that may have been imposed on storage pools.
ClientSettablePoolUsage		Conditional	Conditional requirement: Support for the Storage Relocation profile. Indicates the intended usage or any restrictions that may have been imposed on the usage of a client-settable storage pool.
MaximumElementCreateCount		Optional	Indicates the maximum number of elements that can be specified to be created in a single method call. If 0 or null, there is no limit.
MaximumElementDeleteCount		Conditional	Conditional requirement: Support for the Storage Relocation profile. Indicates the maximum number of elements that can be deleted in a single method call. If 0 or null, there is no limit.
MultipleElementCreateFeatures		Optional	Enumeration indicating features offered by the multiple element create method. "2" (Single instance creation indication).
MultipleElementDeleteFeatures		Optional	Enumeration indicating features offered by the multiple element delete method. "2" (Continue on nonexistent element) or "3" (Return error on nonexistent element).

8.8.42 CIM_StorageConfigurationCapabilities (Primordial)

Created By: Static
Modified By: Static

Deleted By: Static

Requirement: Support for the Storage Relocation profile.

Table 200 describes class CIM_StorageConfigurationCapabilities (Primordial).

Table 200 - SMI Referenced Properties/Methods for CIM_StorageConfigurationCapabilities (Primordial)

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
ElementName		Mandatory	
SupportedStoragePoolFeatures		Optional	Lists what StorageConfigurationService functionalities are implemented. Matches 2|3 (InExtents or Single InPool).
SupportedSynchronousActions		Conditional	Conditional requirement: Support for the Storage Relocation profile. Lists what actions, invoked through StorageConfigurationService methods, shall not produce Concrete jobs. This version of the standard recognizes "2" (Storage Pool Creation), "12" (Storage Element from Element Creation) or "15" (StoragePool Usage Modification) or "17" (StorageVolume To StoragePool Relocation) or "18" (StoragePool To StoragePool Relocation) or "19" (StorageVolume To StorageExtent Relocation) or "20" (StoragePool To StorageExtent Relocation) or "21" (LogicalDisk To StorageExtent Relocation).
SupportedStorageElementTypes		Optional	Lists the type of storage elements that are supported by this implementation. This version of the standard does not recognize any values for this property.
SupportedAsynchronousActions		Conditional	Conditional requirement: Support for the Storage Relocation profile. Lists what actions, invoked through StorageConfigurationService methods, may produce Concrete jobs. This version of the standard recognizes "2" (Storage Pool Creation), "12" (Storage Element from Element Creation) or "15" (StoragePool Usage Modification) or "17" (StorageVolume To StoragePool Relocation) or "18" (StoragePool To StoragePool Relocation) or "19" (StorageVolume To StorageExtent Relocation) or "20" (StoragePool To StorageExtent Relocation) or "21" (LogicalDisk To StorageExtent Relocation) or "22" (Multiple Storage Element Creation) or "23" (Multiple Storage Element Return).

Table 200 - SMI Referenced Properties/Methods for CIM_StorageConfigurationCapabilities (Primordial)

Properties	Flags	Requirement	Description & Notes
SupportedStorageElementFeatures		Conditional	Conditional requirement: Support for the Storage Relocation profile. Lists actions supported through the invocation of StorageServiceService.CreateOrModifyElementFromStoragePool(). This version of the 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) or 14 (StorageVolume To StoragePool Relocation) or 15 (StoragePool To StoragePool Relocation) or 16 (StorageVolume To StorageExtent Relocation) or 17 (StoragePool To StorageExtent Relocation) or 18 (LogicalDisk To StorageExtent Relocation).
SupportedStorageElementUsage		Optional	For Primordial StorageConfigurationCapabilities, this shall be NULL.
ClientSettableElementUsage		Optional	For Primordial StorageConfigurationCapabilities, this shall be NULL.
SupportedStoragePoolUsage		Conditional	Conditional requirement: Support for the Storage Relocation profile. Indicates the intended usage or any restrictions that may have been imposed on storage pools.
ClientSettablePoolUsage		Conditional	Conditional requirement: Support for the Storage Relocation profile. Indicates the intended usage or any restrictions that may have been imposed on the usage of a client-settable storage pool.

8.8.43 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

8.8.44 CIM_StoragePool

This class is unchanged from the Block Services Package. Changes for StoragePools are in the StorageCapabilities associated to the StoragePools.

Requirement: Mandatory

8.8.45 CIM_StoragePool (Concrete)

Created By: Extrinsic: StorageConfigurationService.CreateOrModifyStoragePool

Modified By: Extrinsic: StorageConfigurationService.CreateOrModifyStoragePool

Deleted By: Extrinsic: StorageConfigurationService.DeleteStoragePool
Requirement: Mandatory

Table 201 describes class CIM_StoragePool (Concrete).

Table 201 - SMI Referenced Properties/Methods for CIM_StoragePool (Concrete)

Properties	Flags	Requirement	Description & 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.
OperationalStatus		Conditional	Experimental. Conditional requirement: Support for the Storage Relocation profile. Value shall be 2|3|6|15 (OK or Degraded or Error or Dormant). In addition, the secondary OperationalStatus may be 19 (Relocating) with 2|3|15 (OK or Degraded or Dormant).
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.
CapacityInMigratingSource		Optional	Experimental. The total capacity of extents migrating out from this storage pool.
CapacityInMigratingTarget		Optional	Experimental. The total capacity of extents migrating into this storage pool.
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.

8.8.46 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 202 describes class CIM_StoragePool (Empty).

Table 202 - SMI Referenced Properties/Methods for CIM_StoragePool (Empty)

Properties	Flags	Requirement	Description & 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	
OperationalStatus		Conditional	Experimental. Conditional requirement: Support for the Storage Relocation profile. Value shall be 2|3|6|15 (OK or Degraded or Error or Dormant). In addition, the secondary OperationalStatus may be 19 (Relocating) with 2|3|15 (OK or Degraded or Dormant).
TotalManagedSpace		Mandatory	This shall be 0 for an empty StoragePool.
RemainingManagedSpace		Mandatory	
Usage		Optional	
OtherUsageDescription		Optional	
ClientSettableUsage		Optional	
CapacityInMigratingSource		Optional	Experimental. The total capacity of extents migrating out from this storage pool.
CapacityInMigratingTarget		Optional	Experimental. The total capacity of extents migrating into this storage pool.
GetSupportedSizes()		Conditional	Conditional requirement: Support for StorageConfigurationService.
GetSupportedSizeRange()		Conditional	Conditional requirement: Support for StorageConfigurationService.
GetAvailableExtents()		Optional	

8.8.47 CIM_StoragePool (Primordial)

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

Table 203 describes class CIM_StoragePool (Primordial).

Table 203 - SMI Referenced Properties/Methods for CIM_StoragePool (Primordial)

Properties	Flags	Requirement	Description & 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.
OperationalStatus		Conditional	Experimental. Conditional requirement: Support for the Storage Relocation profile. Value shall be 2|3|6|15 (OK or Degraded or Error or Dormant). In addition, the secondary OperationalStatus may be 19 (Relocating) with 2|3|15 (OK or Degraded or Dormant).
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.
CapacityInMigratingSource		Optional	Experimental. The total capacity of extents migrating out from this storage pool.
CapacityInMigratingTarget		Optional	Experimental. The total capacity of extents migrating into this storage pool.
GetSupportedSizes()		Conditional	Conditional requirement: Support for StorageConfigurationService. List the discrete storage element sizes that can be created or expanded from this Pool.

Table 203 - SMI Referenced Properties/Methods for CIM_StoragePool (Primordial)

Properties	Flags	Requirement	Description & Notes
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.

8.8.48 CIM_StorageSettingWithHints

This class is unchanged from the Block Services Package.

Requirement: Optional

8.8.49 CIM_StorageSettingsAssociatedToCapabilities

This class is unchanged from the Block Services Package.

Requirement: Optional

8.8.50 CIM_StorageSettingsGeneratedFromCapabilities

This class is unchanged from the Block Services Package.

Created By: Extrinsic: CreateSetting
Modified By: ModifyInstance
Deleted By: DeleteInstance
Requirement: Optional

8.8.51 CIM_SystemDevice (System to StorageVolume or LogicalDisk)

Created By: Static
Modified By: Static
Deleted By: Extrinsic: StorageConfigurationService.ReturnToStoragePool
Requirement: Mandatory

Table 204 describes class CIM_SystemDevice (System to StorageVolume or LogicalDisk).

Table 204 - SMI Referenced Properties/Methods for CIM_SystemDevice (System to StorageVolume or LogicalDisk)

Properties	Flags	Requirement	Description & Notes
PartComponent		Mandatory	
GroupComponent		Mandatory	

8.8.52 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: 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 205 describes class SNIA_StorageCapabilities.

Table 205 - SMI Referenced Properties/Methods for SNIA_StorageCapabilities

Properties	Flags	Requirement	Description & Notes
InstanceID (overridden)		Mandatory	
ElementName (overridden)		Mandatory	
ElementType (overridden)		Mandatory	
NoSinglePointOfFailure (overridden)		Mandatory	
NoSinglePointOfFailureDefault (overridden)		Mandatory	
DataRedundancyMin (overridden)		Mandatory	
DataRedundancyMax (overridden)		Mandatory	
DataRedundancyDefault (overridden)		Mandatory	
PackageRedundancyMin (overridden)		Mandatory	

Table 205 - SMI Referenced Properties/Methods for SNIA_StorageCapabilities

Properties	Flags	Requirement	Description & Notes
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	

8.8.53 SNIA_StorageSetting

The SNIA_StorageSetting is subclassed from CIM_StorageSetting and is enhanced to add the DataOrganization, CUImage, 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 206 describes class SNIA_StorageSetting.

Table 206 - SMI Referenced Properties/Methods for SNIA_StorageSetting

Properties	Flags	Requirement	Description & 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.

Table 206 - SMI Referenced Properties/Methods for SNIA_StorageSetting

Properties	Flags	Requirement	Description & Notes
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)).
CUImage (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., 3380 or 3390) that is emulated by the volume.

8.8.54 SNIA_StorageVolume

The SNIA_StorageVolume is subclassed from CIM_StorageVolume and enhances that class to add the DataOrganization, CUImage, 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 207 describes class SNIA_StorageVolume.

Table 207 - SMI Referenced Properties/Methods for SNIA_StorageVolume

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName (overridden)		Mandatory	
SystemName (overridden)		Mandatory	
CreationClassName (overridden)		Mandatory	
DeviceID (overridden)		Mandatory	

Table 207 - SMI Referenced Properties/Methods for SNIA_StorageVolume

Properties	Flags	Requirement	Description & Notes
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	
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.

Table 207 - SMI Referenced Properties/Methods for SNIA_StorageVolume

Properties	Flags	Requirement	Description & Notes
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.
NumExtentsMigrating		Optional	Experimental. The number of Extents in the process of migrating for this storage volume when the volume relocation is on going.
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)).
CUImage (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., 3380 or 3390) that is emulated by the volume.

EXPERIMENTAL

STABLE

Clause 9: Copy Services Subprofile

9.1 Description

9.1.1 Synopsis

Profile Name: Copy Services (Component Profile)

Version: 1.5.0

Organization: SNIA

CIM Schema Version: 2.23

Table 208 describes the related profiles for Copy Services.

Table 208 - Related Profiles for Copy Services

Profile Name	Organization	Version	Requirement	Description
Block Services	SNIA	1.6.0	Mandatory	
Job Control	SNIA	1.5.0	Optional	

Central Class: N/A

Scoping Class: ComputerSystem

9.1.2 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 9.6.9.

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

9.1.3 Copy Services Discovery

The extrinsic methods invoked to create and manage replicas are defined in the StorageConfigurationService class shown in Figure 47.

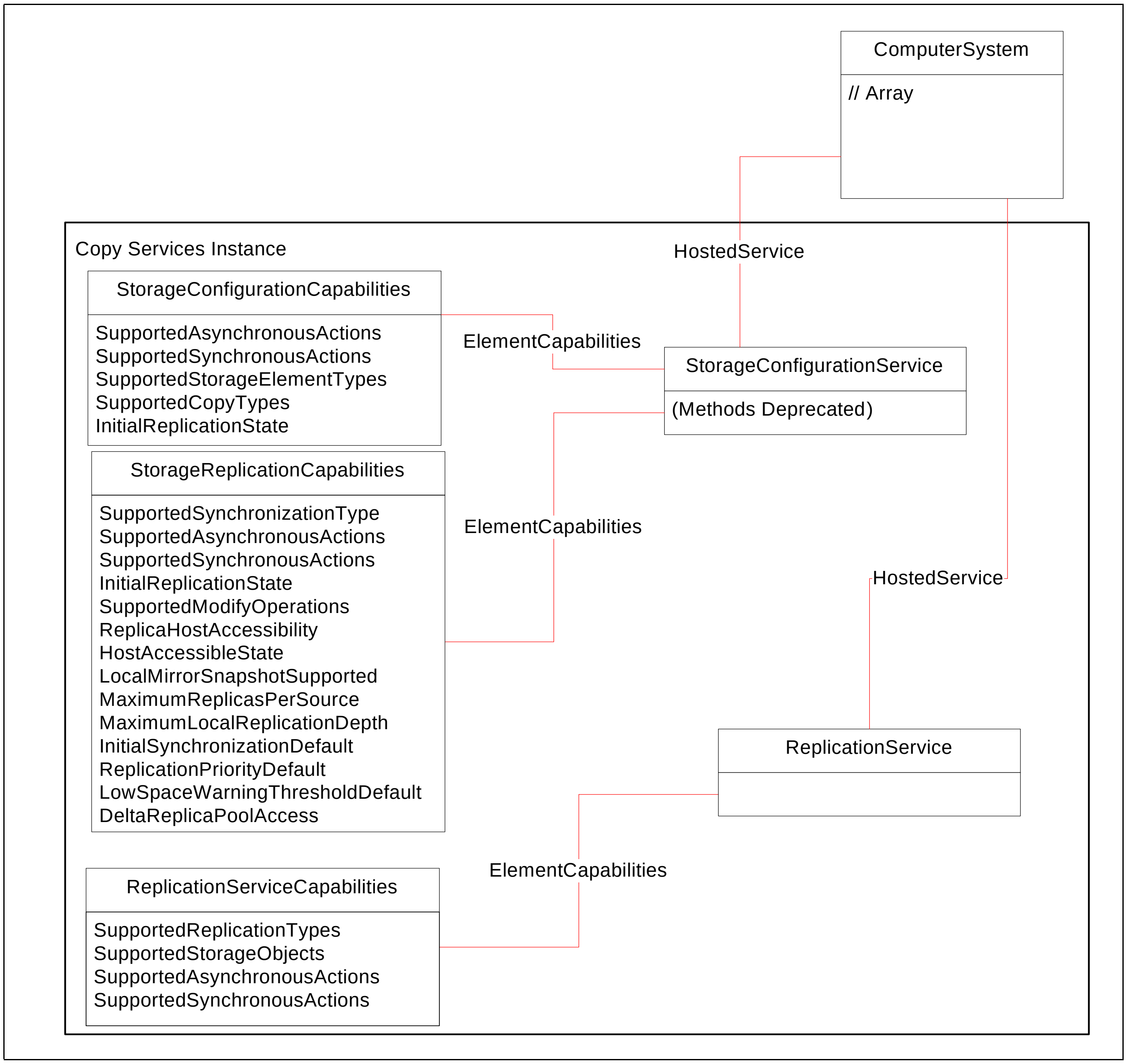
ComputerSystem
// Array
Copy Services Instance
HostedService
StorageConfigurationCapabilities
SupportedAsynchronousActions
SupportedSynchronousActions
SupportedStorageElementTypes
SupportedCopyTypes
InitialReplicationState
ElementCapabilities
StorageConfigurationService
(Methods Deprecated)
StorageReplicationCapabilities
SupportedSynchronizationType
SupportedAsynchronousActions
SupportedSynchronousActions
InitialReplicationState
SupportedModifyOperations
ReplicaHostAccessibility
HostAccessibleState
LocalMirrorSnapshotSupported
MaximumReplicasPerSource
MaximumLocalReplicationDepth
InitialSynchronizationDefault
ReplicationPriorityDefault
LowSpaceWarningThresholdDefault
DeltaReplicaPoolAccess
ElementCapabilities
HostedService
ReplicationService
ElementCapabilities
ReplicationServiceCapabilities
SupportedReplicationTypes
SupportedStorageObjects
SupportedAsynchronousActions
SupportedSynchronousActions

Figure 47 - 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

9.1.4 Copy Services Capabilities

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 47: "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 section discusses both sets of capabilities and how they relate.

EXPERIMENTAL

9.1.4.1 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 209 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 209 - 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 must remain associated with the source element; Independent indicates the target element can exist without the source element.

9.1.4.2 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.

9.1.4.3 Alignment of SupportedSynchronizationType and SupportedReplicationType

The values for SupportedSynchronizationType (in StorageReplicationCapabilities) and SupportedReplicationType (in ReplicationServiceCapabilities) should be aligned with each other. Table 210 the alignment of these properties.

Table 210 - 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

Table 210 - Alignment of SupportedSynchronizationType and SupportedReplicationType

Supported ReplicationType	Supported Synchronization Type	Notes
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

9.1.4.4 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 9.6 "Client Considerations and Recipes".

9.1.5 Replication modeling

Figure 48: "Local Replica" shows the basic model of a local replica.

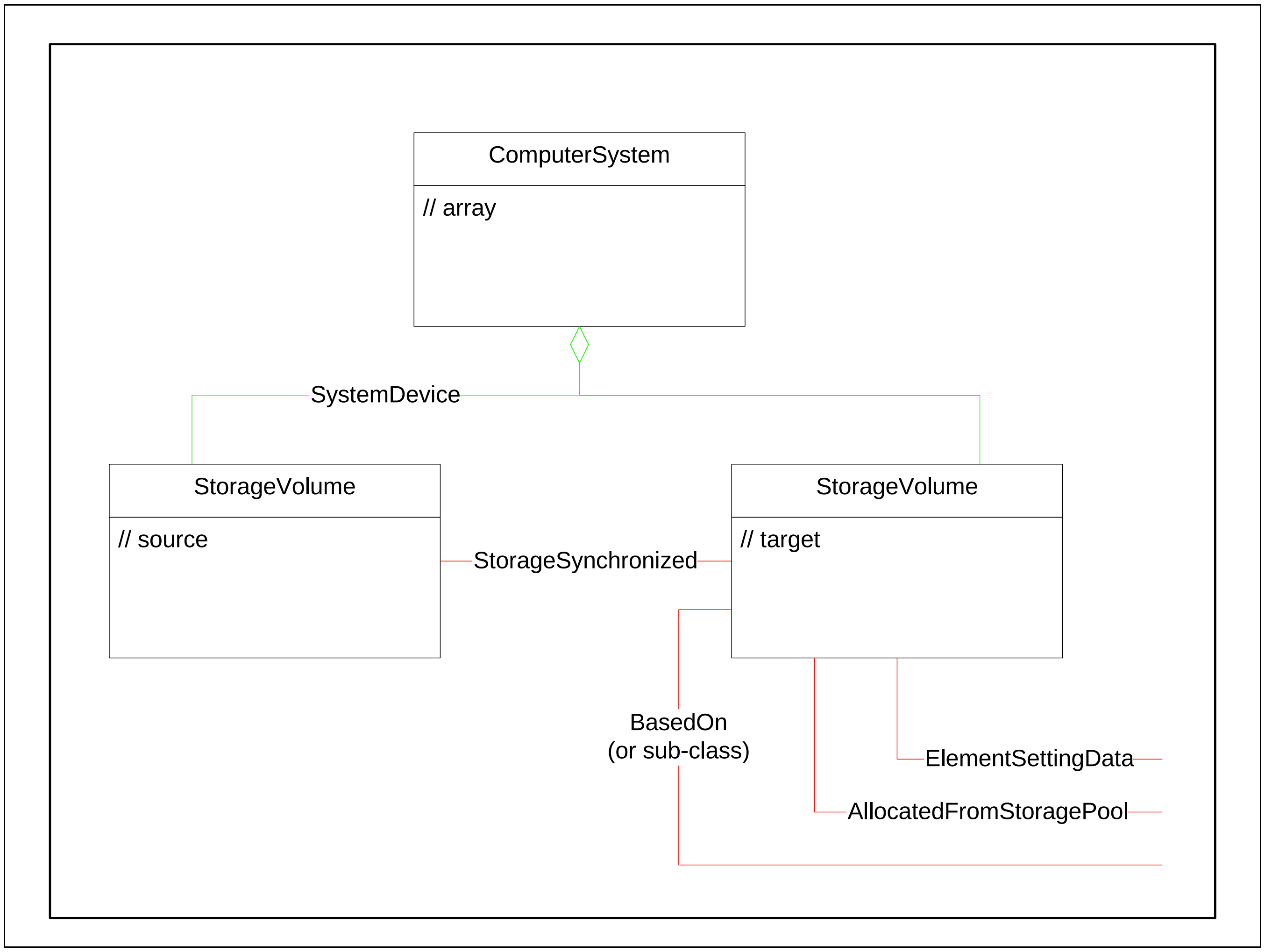

Figure 48 - Local Replica

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

9.1.5.1 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 49: "Multi-Level Local Replication" show the basic model for local multi-level replication.

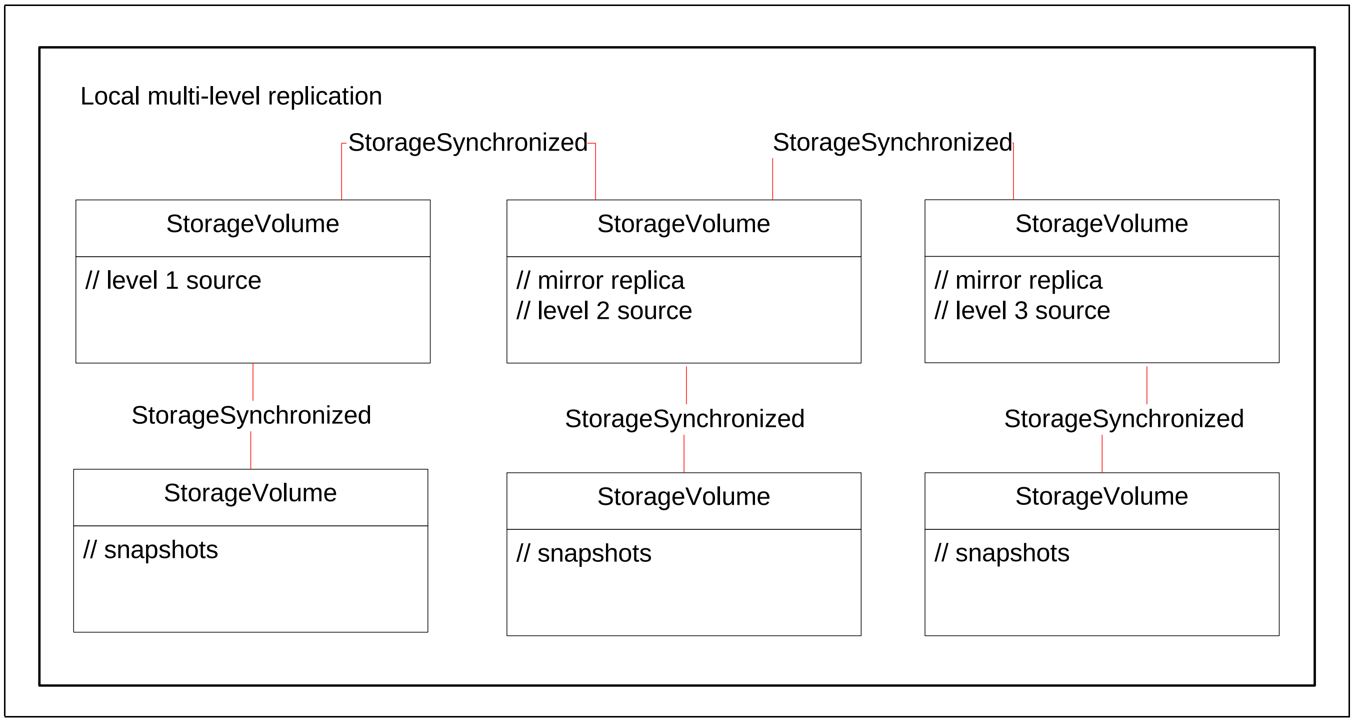

Figure 49 - Multi-Level Local Replication

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, must set the property ReplicationSettingData.Multihop appropriately; see 9.7 "CIM Elements" for details on Multihop specification. The capabilities method GetSupportedMaximum indicates the maximum number of hops supported by the implementation.

EXPERIMENTAL

9.1.5.2 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 9.6 "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 50: "Multiple Snapshots Per Source Element" shows the basic model of snapshots created as delta replicas.

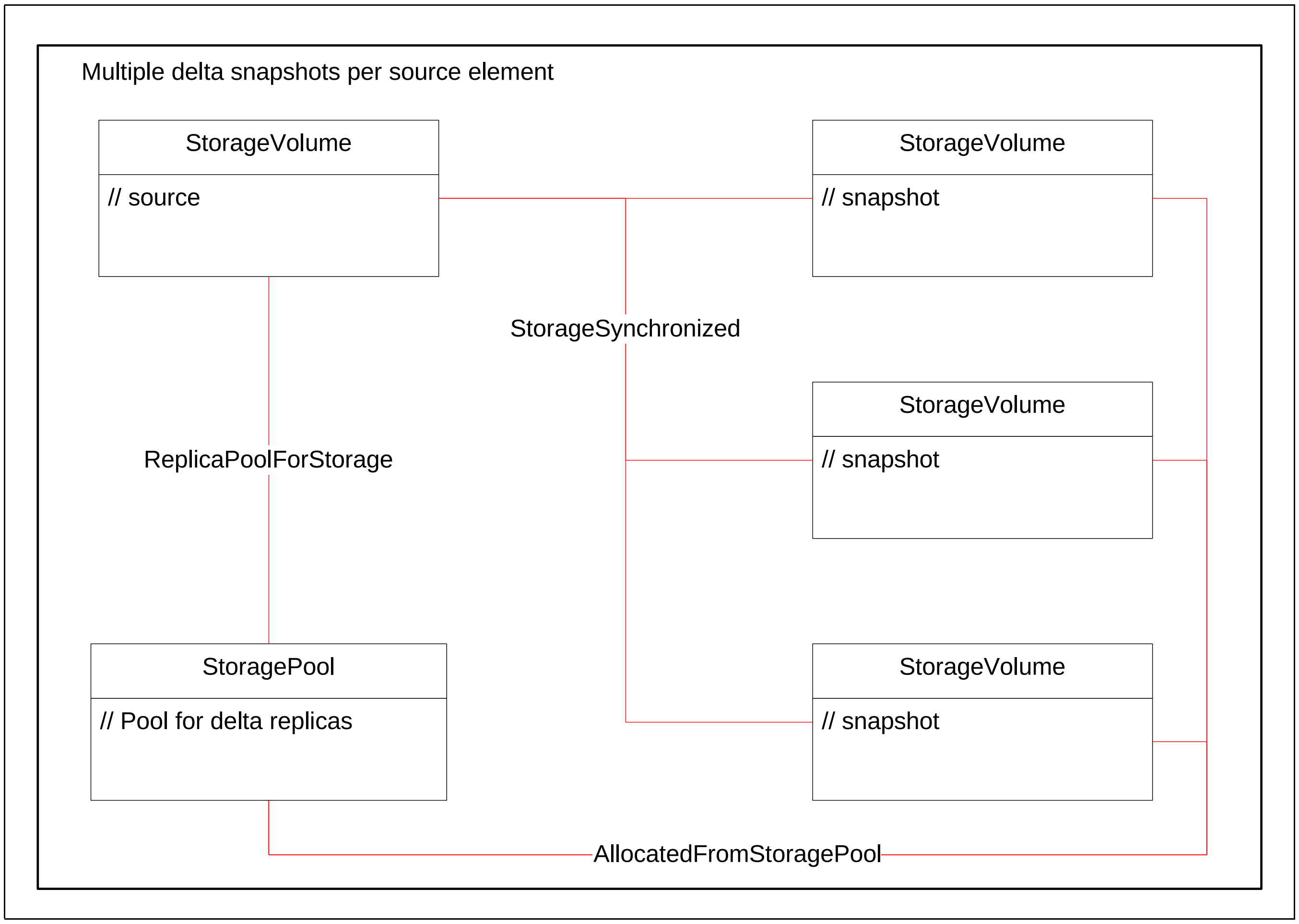

Figure 50 - Multiple Snapshots Per Source Element

9.1.6 Associations

Copy Services utilizes associations.

9.1.6.1 StorageSynchronized Association

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.

9.1.6.1.1 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 211.

Table 211 - 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 212:

Table 212 - 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”.
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”.

Table 212 - Alignment of CopyState and SyncState

CopyState / ProgressStatus	SyncState	Notes
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”.
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”.

Table 212 - Alignment of CopyState and SyncState

CopyState / ProgressStatus	SyncState	Notes
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”.

9.1.6.2 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 51). 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).

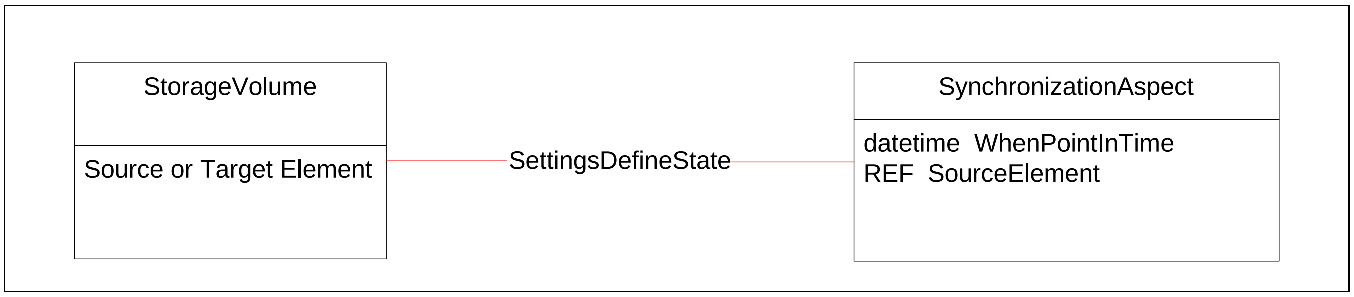

Figure 51 - SettingsDefineState Association

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 52 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.”

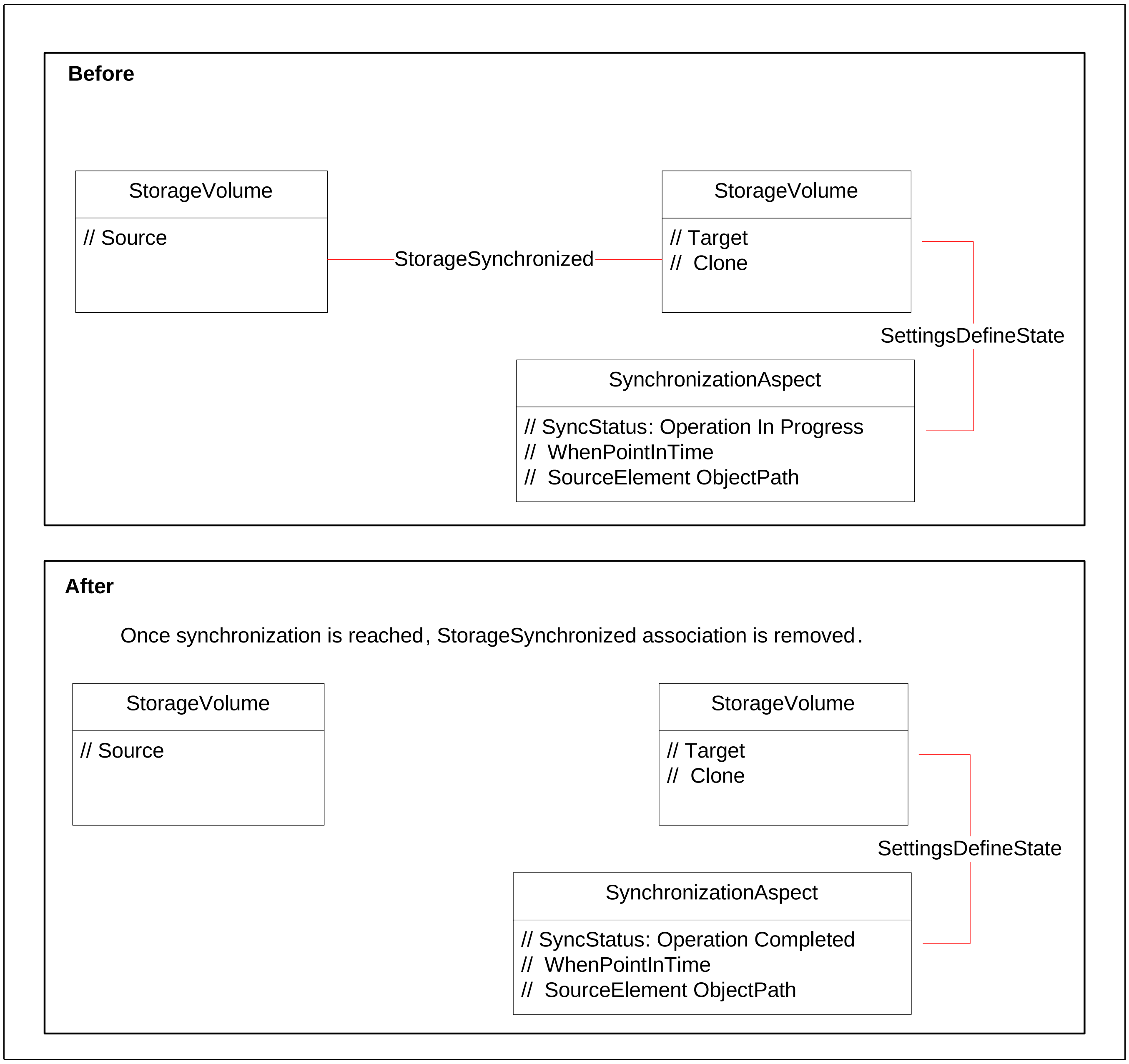

Figure 52 - SynchronizationAspect Instance

EXPERIMENTAL

9.1.7 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.

9.1.8 Accessibility to Created Elements

DEPRECATED

9.1.8.1 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

9.1.8.2 Using ReplicationService Methods

Not defined in this version of the standard.

EXPERIMENTAL

9.1.9 Completion of Long Operations

DEPRECATED

9.1.9.1 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

9.1.9.2 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 to 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 55, Inactive and Synchronized states are considered steady states; whereas Initialized and Unsynchronized are transient states.

EXPERIMENTAL

9.1.10 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 213 shows the classification.

Table 213 - 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

Table 213 - Synchronization Operation Support Requirements (Continued)

ModifySynchronization Operation	Mirror Replicas	Snapshot Replicas
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 214.

Table 214 - 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
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.

Table 215 describes the supported CopyStates.

Table 215 - 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

9.1.11 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.

9.1.12 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 HostAccesibleState[] 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 216. The table does not apply to CopyType “UnSyncUnAssoc”.

Table 216 - 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

Table 216 - SyncMaintained and WhenSynced Properties

Synchronization State	SyncMaintained		WhenSynced	
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.

9.1.13 State Transitions

Figure 53: "State Transitions for Mirrors and Clones" shows state transitions for mirrors and clones:

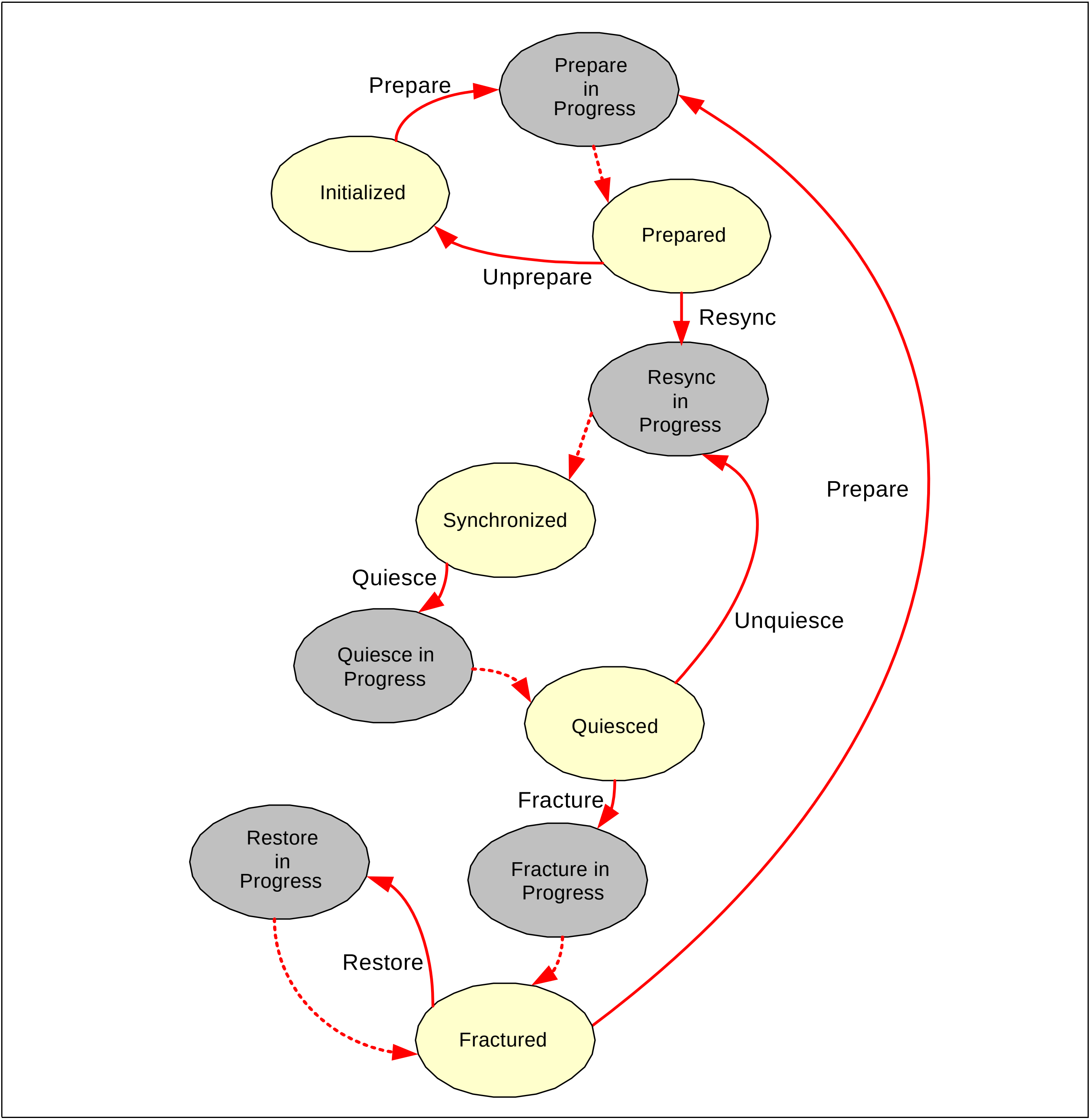

Figure 53 - State Transitions for Mirrors and Clones

Figure 54: "State Transitions for Snapshots and Migration" shows state transitions for snapshots:

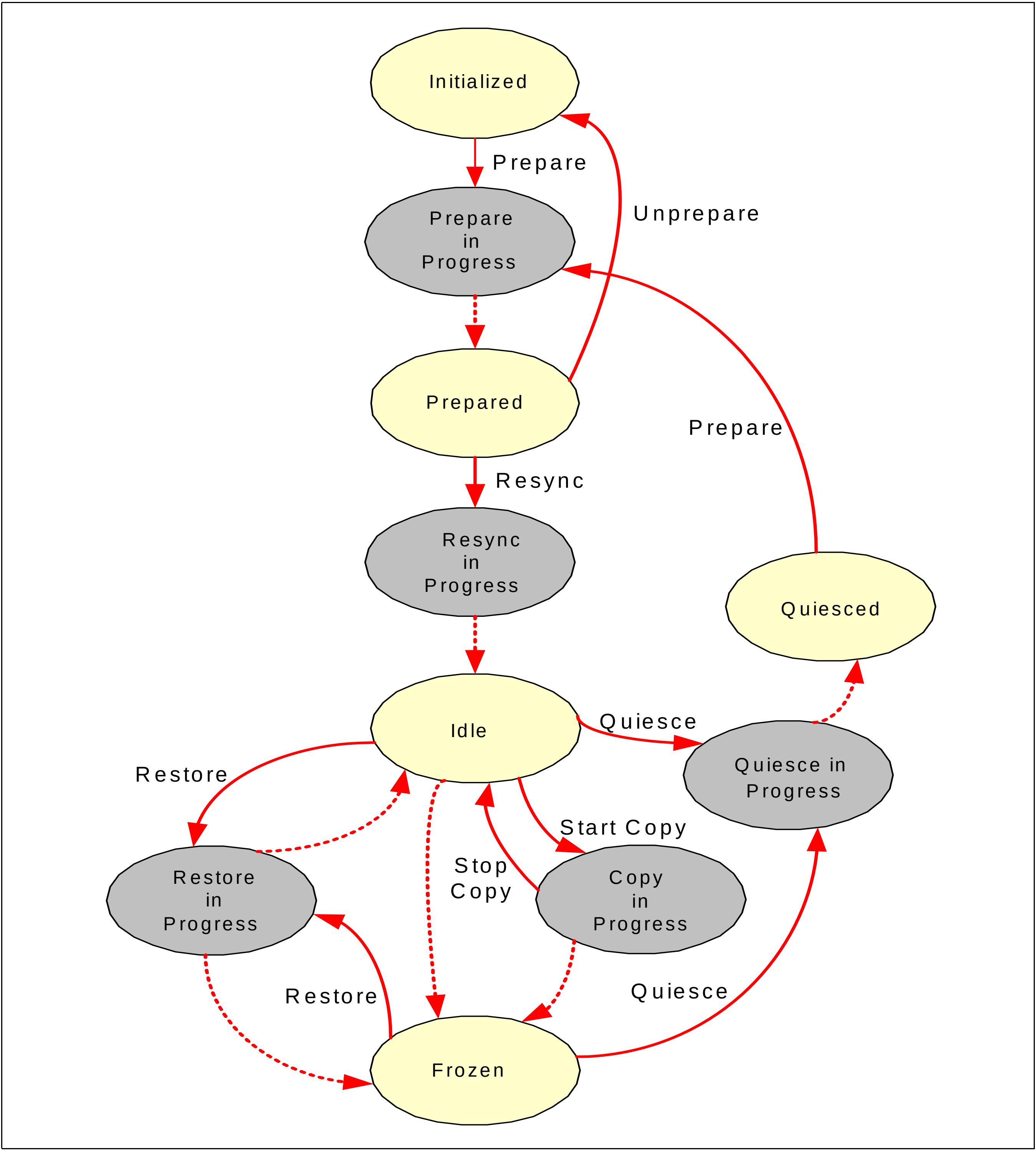

Figure 54 - State Transitions for Snapshots and Migration

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 55 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 such time a Modify method is used to Activate the synchronization.

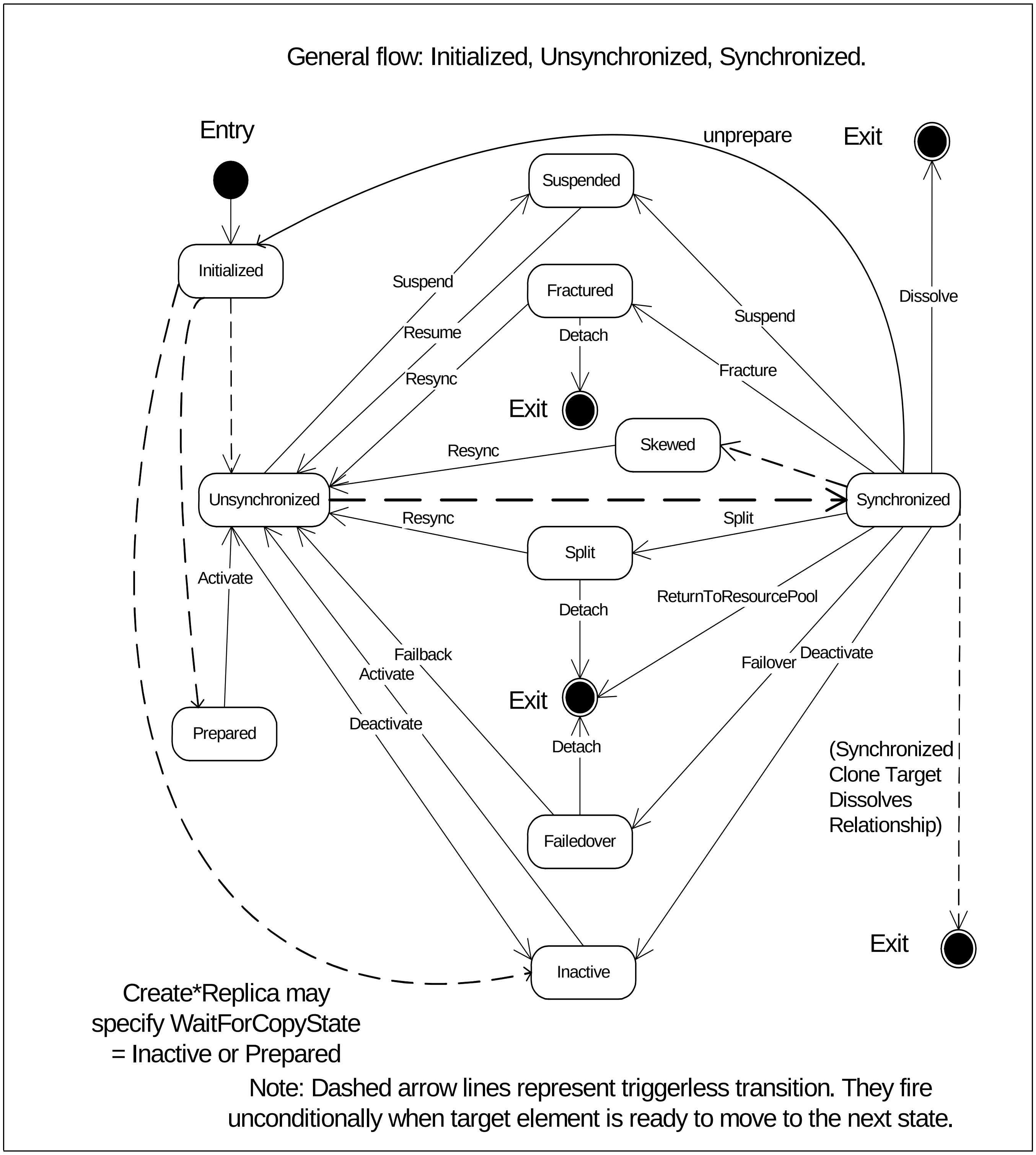
General flow: Initialized, Unsynchronized, Synchronized.
Entry
unprepare
Exit
Suspended
Initialized
Suspend
Fractured
Dissolve
Suspend
Resume
Detach
Fracture
Resync
Exit
Skewed
Resync
Unsynchronized
Synchronized
Resync
Split
Split
Activate
Detach
ReturnToResourcePool
Failback
Activate
Failover
Deactivate
Exit
Deactivate
Prepared
Detach
(Synchronized Clone Target Dissolves Relationship)
Failedover
Exit
Inactive
Create*Replica may specify WaitForCopyState = Inactive or Prepared
Note: Dashed arrow lines represent triggerless transition. They fire unconditionally when target element is ready to move to the next state.

Figure 55 - CopyState Transitions

9.1.13.1 Alignment of State Transitions

Both SyncState and the combination of CopyState and ProgressStatus should be reported and the values need to be aligned. Table 212 addresses the basic alignment. This section 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 there are 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 must 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: 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: 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”.

9.1.13.2 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 56 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”.

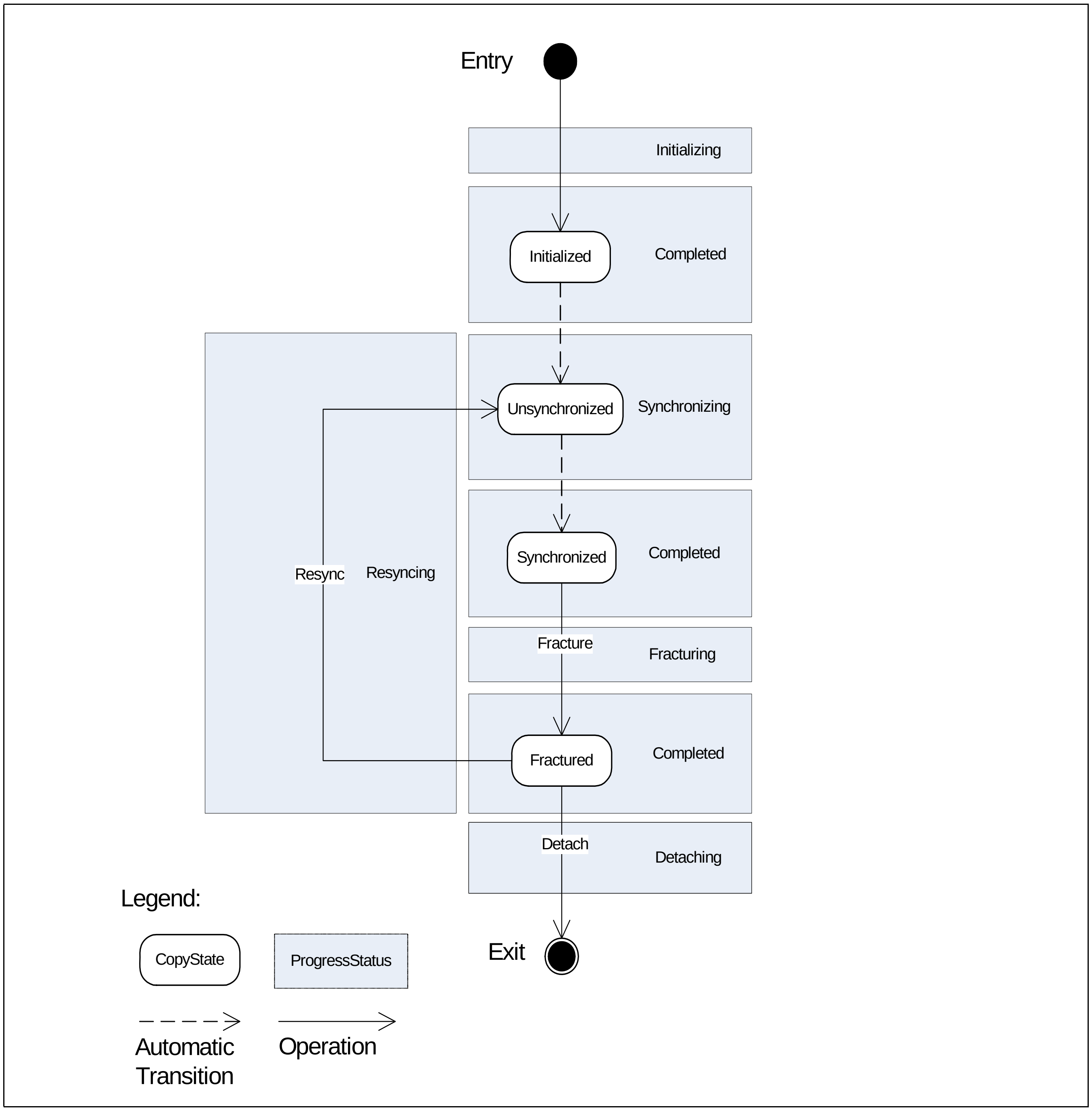

Figure 56 - Sample CopyState and ProgressStatus Transitions

9.1.14 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

9.1.15 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 18: 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 must 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 18: 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

9.1.16 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 9.6 "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 5: 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

9.1.17 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”.

9.1.18 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

9.1.19 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.

9.1.20 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 5: "Block Services Package" for the details of creating and modifying a storage setting.

9.1.21 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 5: "Block Services Package" for creating (specialized) elements and modifying the Usage value of existing elements.

Refer to 9.5.2.4.9 "GetDefaultReplicationSettingData" and 9.5.2.4.4 "GetSupportedFeatures" to determine if the implementation automatically creates target elements, and if specialized elements are required for the desired SyncType.

9.1.22 Using StoragePools for Replicas

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”.

9.1.22.1 Delta Replica StoragePools

Depending on the implementation, the Snapshot targets may require a fixed space consumption or variable space consumption. Refer to 9.5.2.4.4 "GetSupportedFeatures" to determine if specialized storage pool are 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 57 and Figure 58 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.

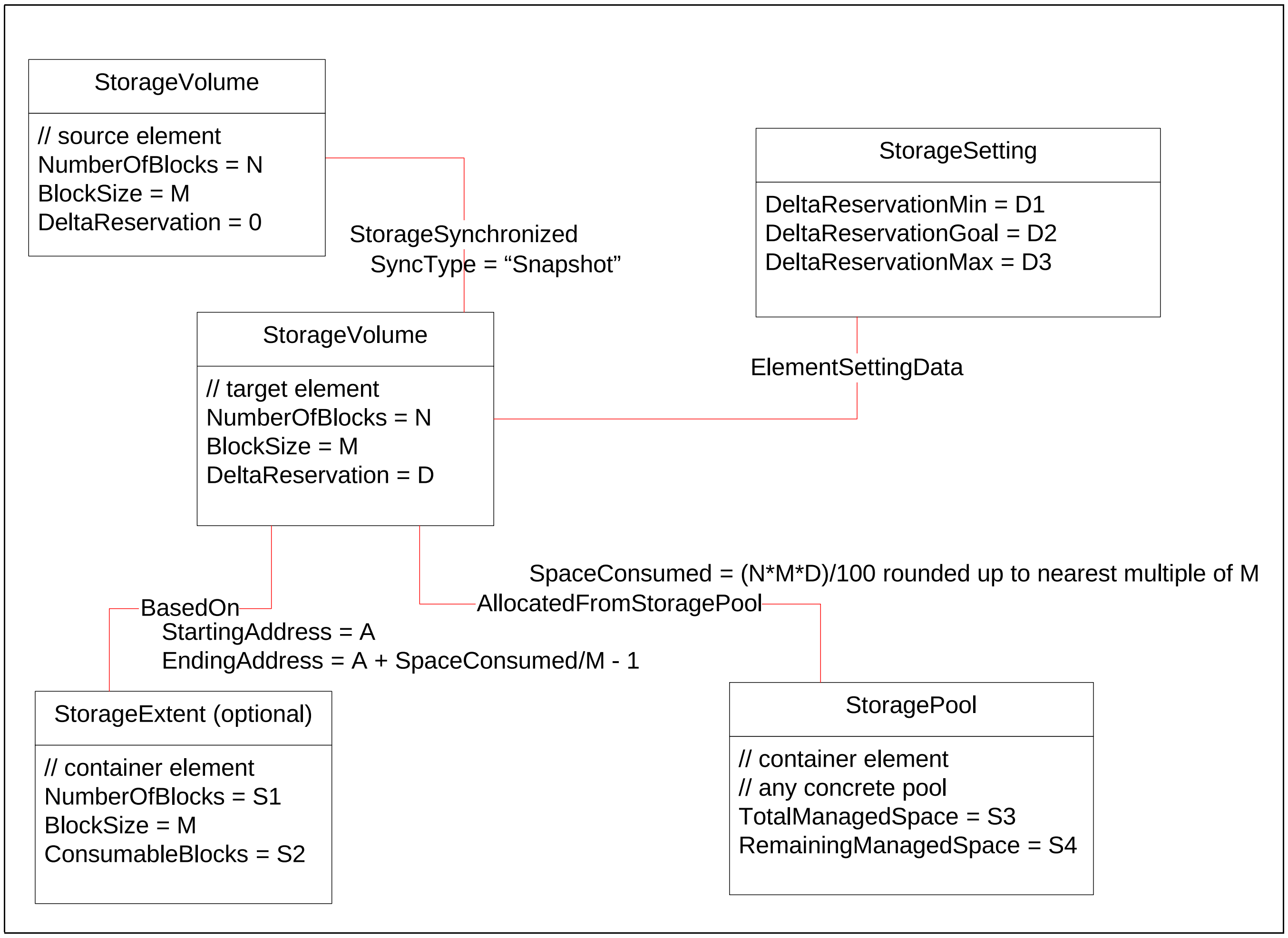

Figure 57 - Fixed Space Consumption

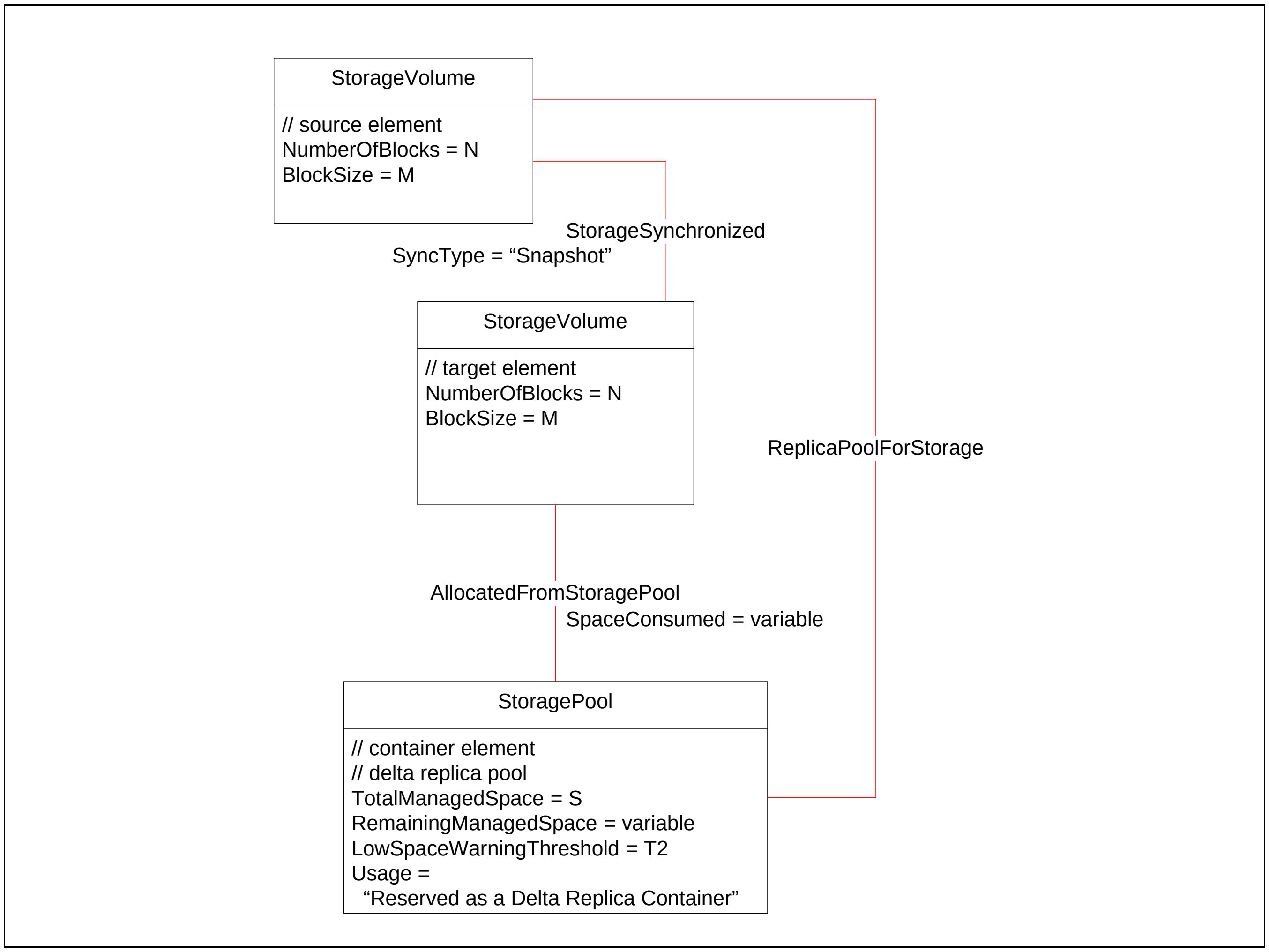

Figure 58 - Variable Space Consumption

9.1.23 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 9.5.2.4.7, 9.7.16, and 9.5.2.4.9.

9.1.24 Indication Events

Depending on the implementation, the Copy Services Profile generates a number of different alert and life cycle indications, shown in Table 217. Clients decide what indications they wish to receive by subscribing to the appropriate indications.

Table 217 - 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 *Broken*.

EXPERIMENTAL

9.1.24.1 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.

9.1.24.2 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

9.1.24.3 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 9.1.24.4).

DEPRECATED

EXPERIMENTAL

9.1.24.4 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.

9.1.24.5 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.

9.1.24.6 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.

9.1.24.7 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.

9.1.24.8 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.

9.1.24.9 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

9.2 Health and Fault Management Considerations

9.2.1 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 218, "Copy Services Alert Indications".

EXPERIMENTAL

The Copy Services Subprofile generates alert indications, shown in Table 218, 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 218 - Copy Services Alert Indications

AlertingManaged Element	PerceivedSeverity	ProbableCause	ProbableCauseDescription
Storage pool	Minor (4)	Threshold Crossed (52)	Pool at low space warning threshold: RemainingManagedSpace/ 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

9.2.2 Replication Error Messages

DEPRECATED

9.2.2.1 Storage Configuration Service Method Messages

The Copy Services Subprofile returns the error responses listed in Table 219 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 219 - 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

9.2.2.2 Replication Service Method Messages

Not defined in this version of the standard.

EXPERIMENTAL

9.3 Cascading Considerations

Not defined in this standard.

9.4 Supported Subprofiles and Packages

See 9.1.1 "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 42: Indication Profile.

9.5 Methods of the Profile

9.5.1 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.

9.5.2 Extrinsic Methods of the Profile

EXPERIMENTAL

9.5.2.1 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

9.5.2.2 StorageConfigurationService Methods

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 220 summarizes the extrinsic methods for replica creation and management in the StorageConfigurationService.

Table 220 - Extrinsic Methods of StorageConfigurationService

Method	Described in
ModifySynchronization()	Table 221, “ModifySynchronization”
CreateReplica()	Table 222, “CreateReplica Method”
AttachReplica()	Not documented

9.5.2.2.1 ModifySynchronization Method

Table 221 lists and describes the ModifySynchronization Method.

Table 221 - 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	Synchronizatio n	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 9.1.8 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.

9.5.2.2.2 CreateReplica Method

.Table 222 describes the CreateReplica Method.

Table 222 - 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 223

Table 223 - 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. 9.1.8 Accessibility to Created Elements explains when a reference to the new replica element is available to the client.

9.5.2.2.3 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,
```

7.3.3.8.8 Client Considerations

9.5.2.2.4 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

9.5.2.3 ReplicationService Methods

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 224 summarizes the extrinsic methods for replica creation and management in the ReplicationService.

Table 224 - Extrinsic Methods of ReplicationService

Method	Described in
CreateElementReplica	Section 9.5.2.3.1
CreateSynchronizationAspect	Section 9.5.2.3.2
ModifyReplicaSynchronization	Section 9.5.2.3.3
ModifyListSynchronization	Section 9.5.2.3.4
ModifySettingsDefineState	Section 9.5.2.3.5
GetAvailableTargetElements	Section 9.5.2.3.6
GetReplicationRelationships	Section 9.5.2.3.7

9.5.2.3.1 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 9.5.2.4.9.
  - 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 215, “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.

9.5.2.3.2 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 9.5.2.3.1: CreateElementReplica's parameters.
- Mode: See 9.5.2.3.1: CreateElementReplica's parameters.
- SourceGroup: This should be null for ungrouped copies.
- SourceElement: See 9.5.2.3.1: CreateElementReplica's parameters.
- Consistency: This should be null for ungrouped copies.
- ReplicationSettingData: See 9.5.2.3.1: CreateElementReplica's parameters.
- Job: See 9.5.2.3.1: 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.

9.5.2.3.3 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 9.5.2.3.1: CreateElementReplica's parameters.
- SyncPair[]: For operations on ungrouped elements, this parameter should be NULL.
- Job: See 9.5.2.3.1: 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 9.5.2.3.1: CreateElementReplica's parameters.

9.5.2.3.4 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 must be valid for the ReplicationType -- SyncType, Mode.
- ReplicationSettingData: See 9.5.2.3.1: CreateElementReplica's parameters.
- Job: See 9.5.2.3.1: 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 9.5.2.3.1: CreateElementReplica's parameters. All the supplied synchronization associations must reach at least the specified CopyState before the method returns.

9.5.2.3.5 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 (9.5.2.3.1).
- Job: See CreateElementReplica’s parameters (9.5.2.3.1).
- TargetSettingGoal: See CreateElementReplica’s parameters (9.5.2.3.1).
- TargetPool: See CreateElementReplica’s parameters (9.5.2.3.1).
- WaitForCopyState: See CreateElementReplica’s parameters (9.5.2.3.1).

9.5.2.3.6 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 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 225 describes the GetAvailableTargetElements Method.

Table 225 - GetAvailableTargetElements Method

Method: GetAvailableTargetElements			
Errors: DRM25, DRM27, MP5, MP11			
Parameters:			
Qualifiers	Name	Type	Description/Values

Table 225 - GetAvailableTargetElements Method

Method: GetAvailableTargetElements			
IN, REQ	SourceElement	LogicalElement REF	The original source volume for the pair
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 (9.5.2.3.1).
- Mode: See CreateElementReplica's parameters (9.5.2.3.1).
- ReplicationSettingData: See CreateElementReplica's parameters (9.5.2.3.1). 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 (9.5.2.3.1).
- Candidates[]: The list of the candidate target elements found.

9.5.2.3.7 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 (9.5.2.3.1). If this parameter is not supplied, all SyncTypes are retrieved.
- Mode: See CreateElementReplica’s parameters (9.5.2.3.1). If this parameter is not supplied, all Modes are retrieved.
- CopyState: Only retrieve synchronization relationships that currently this CopyState (see Table 215, “CopyStates Values,”). If this parameter is not supplied, relationships are retrieved regardless of their current CopyState.
- Job: See CreateElementReplica’s parameters (9.5.2.3.1).
- Synchronizations[]: An array of elements found.

9.5.2.4 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 226 summarizes the extrinsic methods for replica creation and management in the ReplicationService.

Table 226 - Extrinsic Methods of ReplicationServiceCapabilities

Method	Described in
ConvertSyncTypeToReplicationType	Section 9.5.2.4.1
ConvertReplicationTypeToSyncType	Section 9.5.2.4.2
GetSupportedCopyStates	Section 9.5.2.4.3
GetSupportedFeatures	Section 9.5.2.4.4
GetSupportedOperations	Section 9.5.2.4.5
GetSupportedSettingsDefineStateOperations	Section 9.5.2.4.6
GetSupportedThinProvisioningFeatures	Section 9.5.2.4.7
GetSupportedMaximum	Section 9.5.2.4.8
GetDefaultReplicationSettingData	Section 9.5.2.4.9
GetSupportedReplicationSettingData	Section 9.5.2.4.10

9.5.2.4.1 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 227, Table 228, Table 229, and Table 230 show the values for the CovertSyncTypeToReplicationType parameters. These values also appear in the value maps in the appropriate MOF files.

Table 227 - SyncTypes

SyncType	Value
Mirror	6
Snapshot	7
Clone	8

Table 228 - Modes

Mode	Value
Synchronous	2
Asynchronous	3

Table 229 - Local or Remote

LocalOrRemote	Value
Local	2
Remote	3

Table 230 - 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

9.5.2.4.2 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.

9.5.2.4.3 GetSupportedCopyStates

```
uint32 ReplicationServiceCapabilities.GetSupportedCopyStates(
     [IN]   uint16 ReplicationType,
     [OUT]  uint16 SupportedCopyStates[],
     [OUT]  boolean HostAccessible[] );
```

For a given ReplicationType, this method returns the supported CopyStates (Table 215) and a parallel array to indicate whether for a given CopyState the target element is host accessible or not (true or false).

9.5.2.4.4 GetSupportedFeatures

```
uint32 ReplicationServiceCapabilities.GetSupportedFeatures(
     [IN]   uint16 ReplicationType,
     [OUT]  uint16 Features[] );
```

For a given ReplicationType, this method returns the supported features listed in Table 231.

Table 231 - 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.
"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 must 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 = *Broken*), 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.

9.5.2.4.5 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 232.

Refer to Figure 55, “CopyState Transitions” for additional information.

Table 232 - Operations

Operation	Description	Special Consideration
“Abort”	Abort the copy operation if it is possible.	
"Activate Consistency"	Enable consistency.	
“Activate”	Activate an inactive StorageSynchronized association.	
"AddSyncPair"	Add source and target elements of a StorageSynchronized association to the source and target replication groups. The SyncType of the associations must 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.	

Table 232 - Operations

Operation	Description	Special Consideration
"Restore from Replica"	Copy a fractured target element to the source element.	
"Resume"	Continue the copy operation of a suspended (or *Broken*) relationship.	To continue from the *Broken* 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 *consistent* manner.	
"Suspend"	Stop the copy engine in such a way that it can be resumed.	

Table 233 compares the action of similar Operations.

Table 233 - 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.

Table 233 - Comparison of Similar Operations

Operations	Description
Detach versus Dissolve	Detach: The association between the source and target element must 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 9.5.2.4.4) Capabilities.

9.5.2.4.6 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 234.

Table 234 - 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).	

9.5.2.4.7 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.

Table 235 - 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.

9.5.2.4.8 GetSupportedMaximum

```
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 236 shows the list of components that can be specified.

Table 236 - 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.

9.5.2.4.9 GetDefaultReplicationSettingData

```
uint32 ReplicationServiceCapabilities.GetDefaultReplicationSettingData(
     [IN]   uint16 ReplicationType,
     [OUT, EmbeddedObject]
          string DefaultInstance );
```

This method for a given ReplicationType returns the default ReplicationSettingData as an instance.

9.5.2.4.10 GetSupportedReplicationSettingData

Not defined in this version of the standard.

EXPERIMENTAL

9.6 Client Considerations and Recipes

9.6.1 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 237.

Table 237 - 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 238.

Table 238 - 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

9.6.2 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 9.1.4.1.
- Locate the hosted instance of ReplicationService. See 9.1.3.
- 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 9.1.4.

- 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 9.5.2.3.6.
- Verify StoragePools have sufficient free capacity for the target elements. See 9.1.22.
- Invoke the appropriate extrinsic method of the ReplicationService to create a replica. See 9.5.2.3.1.
- 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 9.1.6 and 9.1.24.
- Invoke the method ReplicationService.ModifyReplicaSynchronization to modify a replica. For example, "split" a replica from its source element. See 9.5.2.3.3.

EXPERIMENTAL

9.6.3 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.

9.6.4 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 9.6.8 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 sections), 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

9.6.5 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:

```
TotalManagedSpace = RemainingManagedSpace + SUM(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

9.6.6 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 5: Block Services Package to create a new replica target candidate. Note: 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 9.1.10 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.

9.6.7 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 9.6.9 "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

9.6.8 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 59: "Fixed Space

Consumption" and Figure 60: "Variable Space Consumption" show the fixed and variable space consumption models for delta snapshots:

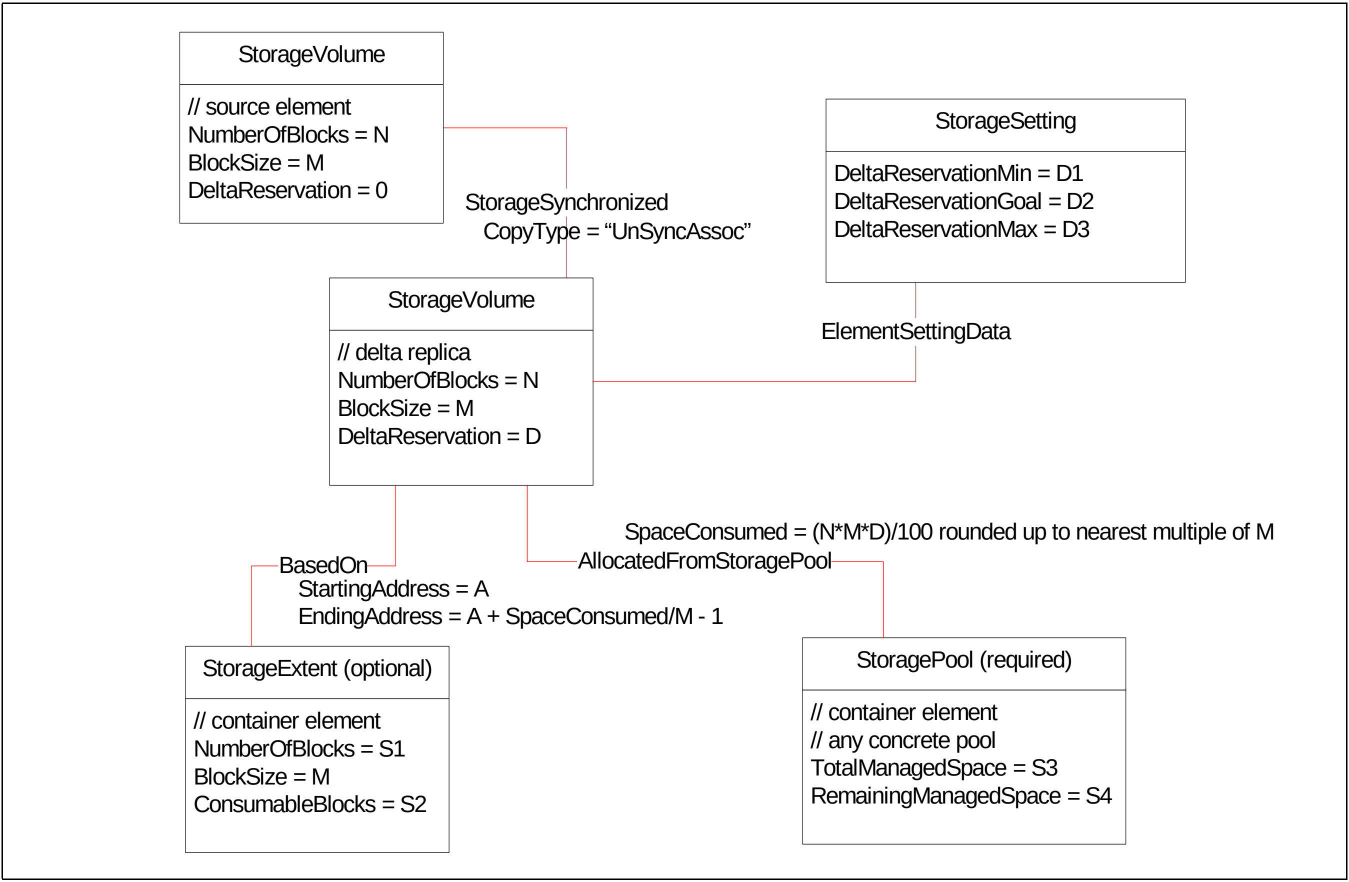

Figure 59 - Fixed Space Consumption

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.

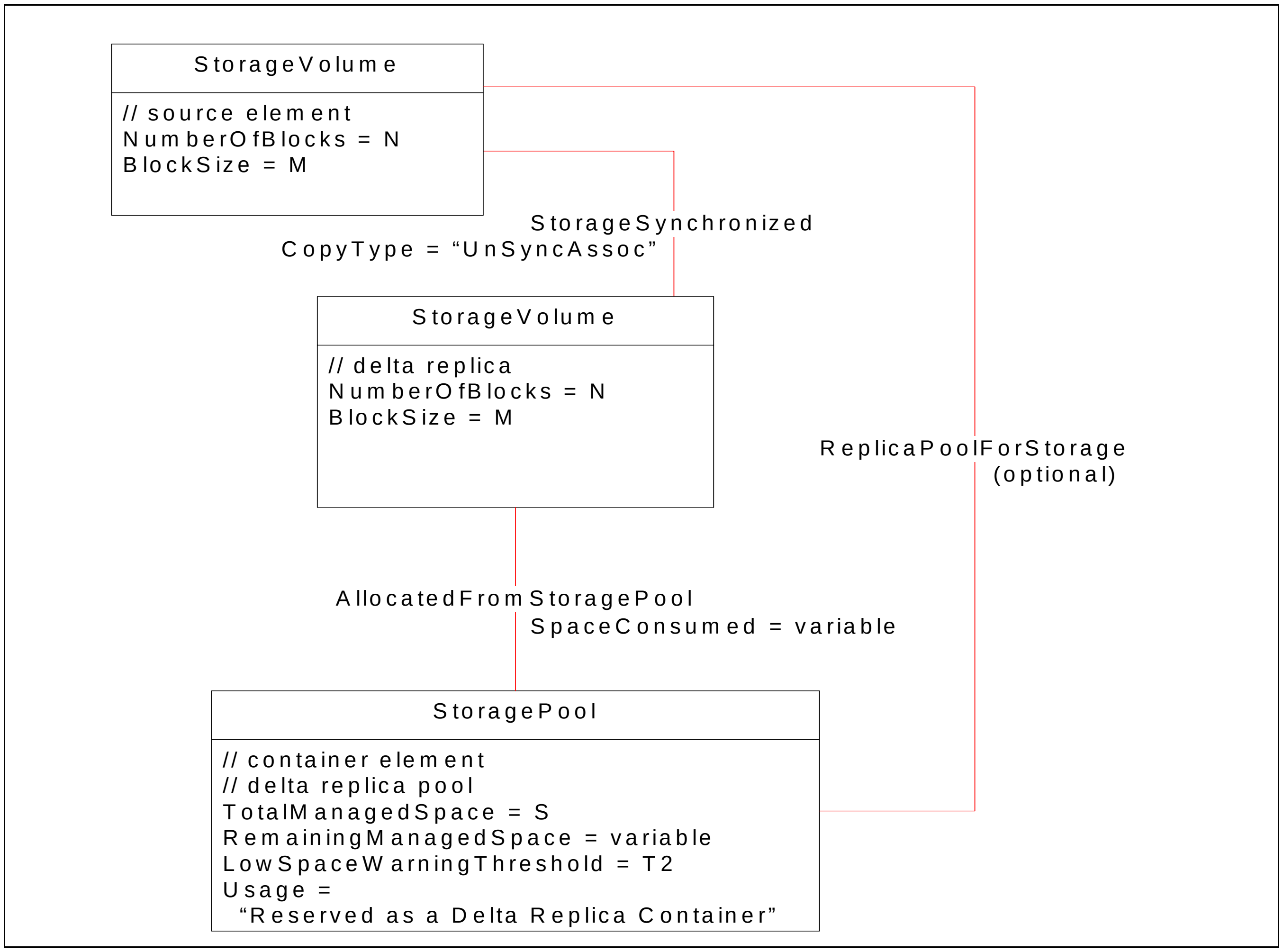

Figure 60 - Variable Space Consumption

The instances of StorageReplicationCapabilities for “UnSyncAssoc-Delta” and “UnSyncAssoc-Full” may use the patterns detailed in Table 239.

Table 239 - Patterns Supported for StorageReplicationCapabilities

SupportedSynchronizationType	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 9.6.5 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:

DeltaReservationMin <= DeltaReservationGoal <= 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 9.6.4 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: 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 240 are used to monitor and manage space consumption for delta replicas using a variable space consumption pattern.

Table 240 - Space Consumption Properties

Delta Replica Property – Variable Space Consumption	Value	Modifiab le
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 241 are used to monitor and manage space consumption for delta replicas using a fixed space consumption pattern.

Table 241 - Space Consumption Properties, Fixed Pattern

Delta Replica Property – FixedSpace Consumption	Value	Modifiab le
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.

9.6.9 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

9.6.10 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.

9.6.11 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 prerequsite 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 ModifySyncronization.
%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.
```

9.6.12 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.
```

9.7 CIM Elements

Table 242 describes the CIM elements for Copy Services.

Table 242 - CIM Elements for Copy Services

Element Name	Requirement	Description
9.7.1 CIM_ElementCapabilities (Associates ReplicationServiceCapabilities and ReplicationService)	Conditional	Experimental. Conditional requirement: The ReplicationService is implemented.
9.7.2 CIM_ElementCapabilities (Associates StorageReplicationCapabilities and StorageConfigurationService)	Mandatory	
9.7.3 CIM_ElementCapabilities (StorageConfigurationCapabilities to StorageConfigurationService)	Mandatory	Associates StorageConfigurationCapabilities with StorageConfigurationService.
9.7.4 CIM_ElementCapabilities (StorageConfigurationCapabilities to StoragePool)	Optional	Associates StorageConfigurationCapabilities with StoragePool.
9.7.5 CIM_HostedService (Replication Service)	Conditional	Experimental. Conditional requirement: The ReplicationService is implemented.
9.7.6 CIM_HostedService (Storage Configuration Service)	Mandatory	
9.7.7 CIM_ReplicaPoolForStorage	Optional	Experimental. Associates special storage pool for Snapshots (delta replicas) to a source element.
9.7.8 CIM_ReplicationService	Optional	Experimental. Base class for Replication Services. Methods are described in the Extrinsic Methods clause.
9.7.9 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.
9.7.10 CIM_ReplicationSettingData	Optional	Experimental. Contains special options for use by methods of Replication Services.
9.7.11 CIM_SettingsDefineState	Optional	Experimental. Associates a storage object to an instance of SynchronizationAspect.
9.7.12 CIM_StorageCapabilities	Mandatory	Base definition is in Block Services Package.
9.7.13 CIM_StorageConfigurationCapabilities	Mandatory	Base definition is in Block Services Package. Adds two properties.
9.7.14 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.

Table 242 - CIM Elements for Copy Services

Element Name	Requirement	Description
9.7.15 CIM_StoragePool	Mandatory	Base definition is in Block Services Package.
9.7.16 CIM_StorageReplicationCapabilities	Mandatory	A set of properties that describe the capabilities of a copy services provider.
9.7.17 CIM_StorageSetting	Mandatory	Base definition is in Block Services Package.
9.7.18 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.
9.7.19 CIM_StorageSynchronized (Between StorageExtent elements)	Mandatory	Associates replica target element to a source element.
9.7.20 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 9.1.24.1 InstCreation on StorageSynchronized.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_StorageSynchronized	Mandatory	All instance deletion indications for StorageSynchronized. See 9.1.24.2 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 9.1.24.3 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 9.1.24.3 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 9.1.24.4 Qualified InstDeletion on StorageSynchronized.

Table 242 - CIM Elements for Copy Services

Element Name	Requirement	Description
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 9.1.24.5 Qualified InstModification on CopyState.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_StorageSynchronized AND SourceInstance.CIM_StorageSynchronized::ProgressStatus <> PreviousInstance.CIM_StorageSynchronized::ProgressStatus AND OBJECTPATH(SourceInstanceModelpath) = OBJECTPATH('string-key-of-storage-synchronized')	Optional	Experimental. CQL -Progress status transition for a specific replica association. See 9.1.24.6 Qualified InstModification on ProgressStatus.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_StorageSynchronized AND SourceInstance.CIM_StorageSynchronized::ProgressStatus <> PreviousInstance.CIM_StorageSynchronized::ProgressStatus	Optional	Experimental. CQL -Progress status transition for replica associations. See 9.1.24.7 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 9.1.24.8 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 9.1.24.9 AlertIndication on StoragePool.

9.7.1 CIM_ElementCapabilities (Associates ReplicationServiceCapabilities and ReplicationService)

Experimental.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: The ReplicationService is implemented.

Table 243 describes class CIM_ElementCapabilities (Associates ReplicationServiceCapabilities and ReplicationService).

Table 243 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (Associates ReplicationServiceCapabilities and ReplicationService)

Properties	Flags	Requirement	Description & Notes
Capabilities		Mandatory	
ManagedElement		Mandatory	

9.7.2 CIM_ElementCapabilities (Associates StorageReplicationCapabilities and StorageConfigurationService)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 244 describes class CIM_ElementCapabilities (Associates StorageReplicationCapabilities and StorageConfigurationService).

Table 244 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (Associates StorageReplicationCapabilities and StorageConfigurationService)

Properties	Flags	Requirement	Description & Notes
Capabilities		Mandatory	
ManagedElement		Mandatory	

9.7.3 CIM_ElementCapabilities (StorageConfigurationCapabilities to StorageConfigurationService)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 245 describes class CIM_ElementCapabilities (StorageConfigurationCapabilities to StorageConfigurationService).

Table 245 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (StorageConfigurationCapabilities to StorageConfigurationService)

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

9.7.4 CIM_ElementCapabilities (StorageConfigurationCapabilities to StoragePool)

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

Table 246 describes class CIM_ElementCapabilities (StorageConfigurationCapabilities to StoragePool).

Table 246 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (StorageConfigurationCapabilities to StoragePool)

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

9.7.5 CIM_HostedService (Replication Service)

Experimental.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: The ReplicationService is implemented.

Table 247 describes class CIM_HostedService (Replication Service).

Table 247 - SMI Referenced Properties/Methods for CIM_HostedService (Replication Service)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	The hosting System.
Dependent		Mandatory	The Replication Service hosted on the System.

9.7.6 CIM_HostedService (Storage Configuration Service)

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

Table 248 describes class CIM_HostedService (Storage Configuration Service).

Table 248 - SMI Referenced Properties/Methods for CIM_HostedService (Storage Configuration Service)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	The hosting System.
Dependent		Mandatory	The Storage Configuration Service hosted on the System.

9.7.7 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 249 describes class CIM_ReplicaPoolForStorage.

Table 249 - SMI Referenced Properties/Methods for CIM_ReplicaPoolForStorage

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

9.7.8 CIM_ReplicationService

Experimental. Base class for Replication Services.

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

Table 250 describes class CIM_ReplicationService.

Table 250 - SMI Referenced Properties/Methods for CIM_ReplicationService

Properties	Flags	Requirement	Description & Notes
CreateElementReplica()		Mandatory	
CreateSynchronizationAspect()		Optional	
ModifyReplicaSynchronization()		Mandatory	
ModifyListSynchronization()		Optional	

Table 250 - SMI Referenced Properties/Methods for CIM_ReplicationService

Properties	Flags	Requirement	Description & Notes
ModifySettingsDefineState()		Optional	
GetAvailableTargetElements()		Optional	
GetReplicationRelationships()		Optional	

9.7.9 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 251 describes class CIM_ReplicationServiceCapabilities.

Table 251 - SMI Referenced Properties/Methods for CIM_ReplicationServiceCapabilities

Properties	Flags	Requirement	Description & 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.

Table 251 - SMI Referenced Properties/Methods for CIM_ReplicationServiceCapabilities

Properties	Flags	Requirement	Description & Notes
SupportedAsynchronousActions	N	Mandatory	Identify replication methods using job control. Values: 2: CreateReplica 4: CreateSynchronizationAspect 5: ModifySynchronization 7: ModifySettingsDefineState 8: GetAvailableTargetElements 10: GetReplicationRelationships.
SupportedSynchronousActions	N	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	

Table 251 - SMI Referenced Properties/Methods for CIM_ReplicationServiceCapabilities

Properties	Flags	Requirement	Description & Notes
GetDefaultReplicationSettingData()		Optional	
GetSupportedReplicationSettingData()		Optional	

9.7.10 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 252 describes class CIM_ReplicationSettingData.

Table 252 - SMI Referenced Properties/Methods for CIM_ReplicationSettingData

Properties	Flags	Requirement	Description & Notes
Pairing	MN	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	MN	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.

Table 252 - SMI Referenced Properties/Methods for CIM_ReplicationSettingData

Properties	Flags	Requirement	Description & Notes
TargetElementSupplier	MN	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	MN	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	MN	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	MN	Optional	If non-zero, it specifies the interval between the snapshots of source element, for example, every 23 minutes (00000000002300.000000:000). If zero or NULL, the implementation decides.
Multihop	MN	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.

9.7.11 CIM_SettingsDefineState

Experimental. Associates a storage object to an instance of SynchronizationAspect.

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

Table 253 describes class CIM_SettingsDefineState.

Table 253 - SMI Referenced Properties/Methods for CIM_SettingsDefineState

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	Storage Element.
SettingData		Mandatory	Synchronization Aspect.

9.7.12 CIM_StorageCapabilities

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

Table 254 describes class CIM_StorageCapabilities.

Table 254 - SMI Referenced Properties/Methods for CIM_StorageCapabilities

Properties	Flags	Requirement	Description & Notes
DeltaReservationMin		Mandatory	Refer to property descriptions for CIM_StorageSetting class.
DeltaReservationMax		Mandatory	
DeltaReservationDefault		Mandatory	Initial value for CIM_StorageSetting.DeltaReservationGoal.

9.7.13 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 255 describes class CIM_StorageConfigurationCapabilities.

Table 255 - SMI Referenced Properties/Methods for CIM_StorageConfigurationCapabilities

Properties	Flags	Requirement	Description & Notes
SupportedAsynchronousActions	N	Mandatory	Identify replication methods using job control. Values: 8: Replica Creation 9: Replica Modification 10: Replica Attachment.
SupportedSynchronousActions	N	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.

9.7.14 CIM_StorageConfigurationService

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

Table 256 describes class CIM_StorageConfigurationService.

Table 256 - SMI Referenced Properties/Methods for CIM_StorageConfigurationService

Properties	Flags	Requirement	Description & 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.

9.7.15 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 257 describes class CIM_StoragePool.

Table 257 - SMI Referenced Properties/Methods for CIM_StoragePool

Properties	Flags	Requirement	Description & Notes
LowSpaceWarningThreshold	M	Optional	Experimental. Percentage of TotalManagedSpace triggering an alert indication. When RemainingManagedSpace reaches or falls below this percentage, the indication is generated.

9.7.16 CIM_StorageReplicationCapabilities

This class defines all of the capability properties for a replication service. A provider must supply one instance for each SupportedSynchronizationType value supported.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 258 describes class CIM_StorageReplicationCapabilities.

Table 258 - SMI Referenced Properties/Methods for CIM_StorageReplicationCapabilities

Properties	Flags	Requirement	Description & Notes
SupportedSynchronizationType		Mandatory	Provider must supply one instance of this class for each supported value. Values: 2: Async 3: Sync 4: UnSyncAssoc-Full 5: UnSyncAssoc-Delta 6: UnSyncUnAssoc.
SupportedAsynchronousActions	N	Mandatory	Identify replication methods using job control. Values: 2: Local Replica Creation 4: Local Replica Modification 6: Local Replica Attachment.
SupportedSynchronousActions	N	Mandatory	Identify replication methods not using job control. Values: 2: Local Replica Creation 4: Local Replica Modification 6: Local Replica Attachment.
InitialReplicationState		Mandatory	The initial SyncState when replica creation is completed. Values: 2: Initialized 3: Prepared 4: Synchronized 5: Idle.

Table 258 - SMI Referenced Properties/Methods for CIM_StorageReplicationCapabilities

Properties	Flags	Requirement	Description & Notes
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.

Table 258 - SMI Referenced Properties/Methods for CIM_StorageReplicationCapabilities

Properties	Flags	Requirement	Description & Notes
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.
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.

Table 258 - SMI Referenced Properties/Methods for CIM_StorageReplicationCapabilities

Properties	Flags	Requirement	Description & Notes
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.

9.7.17 CIM_StorageSetting

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

Table 259 describes class CIM_StorageSetting.

Table 259 - SMI Referenced Properties/Methods for CIM_StorageSetting

Properties	Flags	Requirement	Description & Notes
DeltaReservationMin	M	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	M	Mandatory	Maximum space reserved for a delta replica at time of creation. Value 0 to 100 is a percentage of the source element size.
DeltaReservationGoal	M	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.

Table 259 - SMI Referenced Properties/Methods for CIM_StorageSetting

Properties	Flags	Requirement	Description & Notes
InitialSynchronization	M	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	M	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.

9.7.18 CIM_StorageSynchronized

Experimental. Associates replica target element to source element. CIM_StorageSynchronized is subclassed from CIM_StorageSynchronized.

Created By: Extrinsics: CreateReplica, AttachReplica, CreateElementReplica
Modified By: Extrinsics: ModifySynchronization, ModifyReplicaSynchronization
Deleted By: Extrinsics: ModifySynchronization, ModifyReplicaSynchronization
Requirement: The ReplicationService is implemented.

Table 260 describes class CIM_StorageSynchronized.

Table 260 - SMI Referenced Properties/Methods for CIM_StorageSynchronized

Properties	Flags	Requirement	Description & Notes
WhenEstablished	N	Optional	Specifies when the association was established.
WhenActivated	N	Optional	Specifies when the association was activated.
WhenSuspended	N	Optional	Specifies when the association was suspended.

Table 260 - SMI Referenced Properties/Methods for CIM_StorageSynchronized

Properties	Flags	Requirement	Description & Notes
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.
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.

Table 260 - SMI Referenced Properties/Methods for CIM_StorageSynchronized

Properties	Flags	Requirement	Description & Notes
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.
PercentSynced	N	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 derived by subtracting PercentSynched from 100.
SyncedElement		Mandatory	
SystemElement		Mandatory	

9.7.19 CIM_StorageSynchronized (Between StorageExtent elements)

Created By: Extrinsics: CreateReplica, AttachReplica, CreateElementReplica
Modified By: Extrinsics: ModifySynchronization, ModifyReplicaSynchronization
Deleted By: Extrinsics: ModifySynchronization, ModifyReplicaSynchronization
Requirement: Mandatory

Table 261 describes class CIM_StorageSynchronized (Between StorageExtent elements).

Table 261 - SMI Referenced Properties/Methods for CIM_StorageSynchronized (Between Storage-Extent elements)

Properties	Flags	Requirement	Description & Notes
WhenSynced	N	Mandatory	If the replica is a PIT image, this value is the date/time created.
SyncMaintained		Mandatory	Boolean indicating whether synchronization is maintained.

Table 261 - SMI Referenced Properties/Methods for CIM_StorageSynchronized (Between Storage-Extent elements)

Properties	Flags	Requirement	Description & Notes
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 261 - SMI Referenced Properties/Methods for CIM_StorageSynchronized (Between Storage-Extent elements)

Properties	Flags	Requirement	Description & Notes
CopyPriority	M	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	

9.7.20 CIM_SynchronizationAspect

Experimental. Keeps track of source of a copy operation and point-in-time.

Created By: Extrinsics: CreateElementReplica, CreateSynchronizationAspect
Modified By: Extrinsic: ModifyReplicaSynchronization
Deleted By: Extrinsics: ModifyReplicaSynchronization, ModifySettingsDefineState
Requirement: Optional

Table 262 describes class CIM_SynchronizationAspect.

Table 262 - SMI Referenced Properties/Methods for CIM_SynchronizationAspect

Properties	Flags	Requirement	Description & 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.

Table 262 - SMI Referenced Properties/Methods for CIM_SynchronizationAspect

Properties	Flags	Requirement	Description & Notes
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	N	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

Clause 10: Disk Drive Subprofile

The functionality of the Disk Drive Subprofile has been subsumed by the Clause 11: Disk Drive Lite Subprofile.

The Disk Drive Subprofile is defined in section 7.3.3.4 of SMI-S 1.0.2.

DEPRECATED

STABLE

Clause 11: Disk Drive Lite Subprofile

11.1 Description

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.

11.1.1 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 must define the "PortType" property as mandatory so that it can be used to determine the interface on the drive.

Note: The logicalPort class, ProtocolEndpoint, and the DiskDrive properties DiskType, FormFactor, and Encryption will be made mandatory in the future.

11.1.2 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.

11.1.3 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 of the indicator is support and it's value. When written the indicator's state is set.

11.1.4 Diagram of CIM Elements

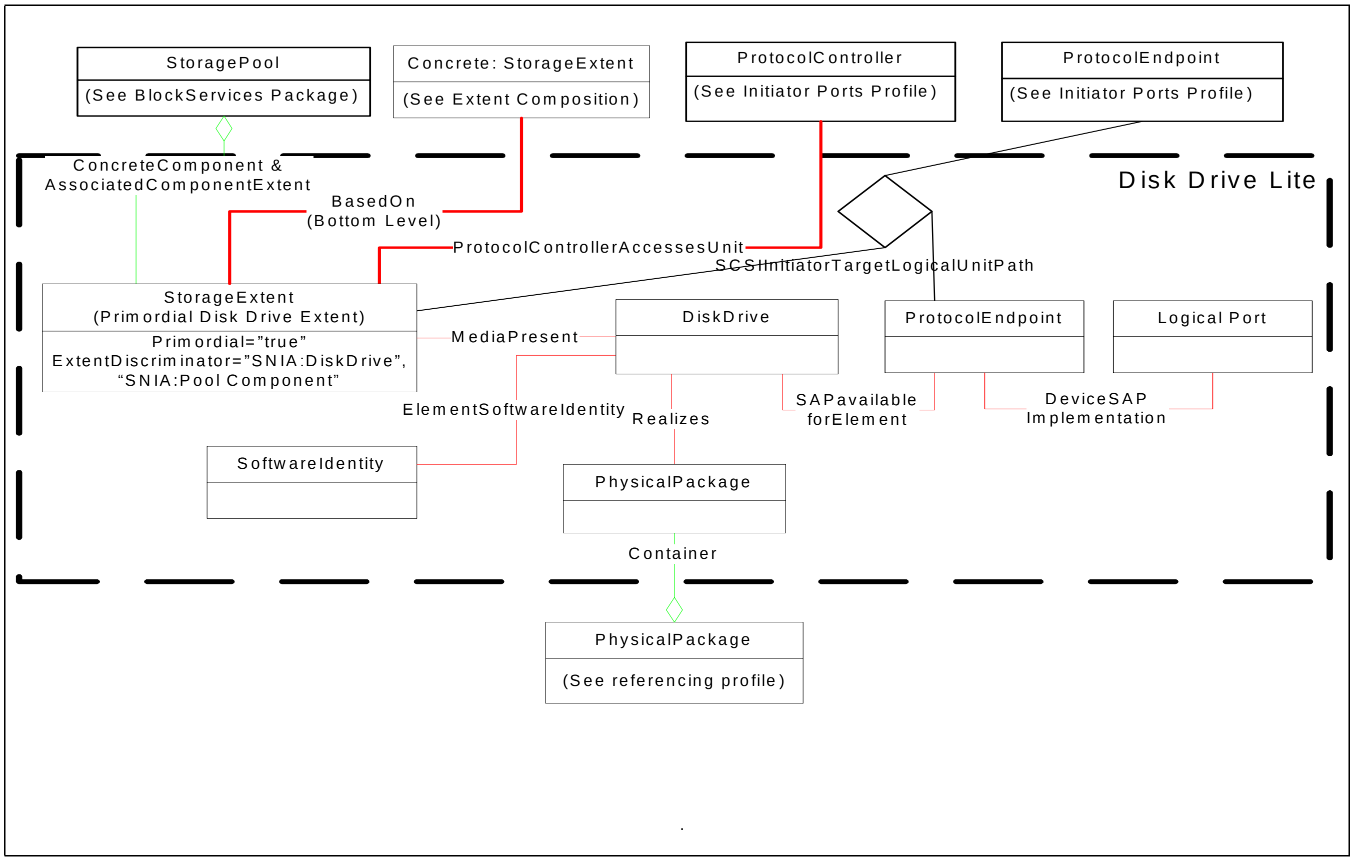

Figure 61 - CIM Elements in the Disk Drive Model

Figure 61 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.

11.1.5 Durable Names and Correlatable IDs of the Profile

None.

11.1.6 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.

11.1.7 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.

11.2 Health and Fault Management Considerations

The DiskDrive.OperationalStatus contains the overall status of the disk, summarized in Table 263.

Table 263 - OperationalStatus For DiskDrive

Primary Operational Status	Subsidiary Operational Status	Description
2 “OK”		Disk Drive is enabled.
5 “Predictive Failure”		Disk Drive is functionality 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.

Table 263 - OperationalStatus For DiskDrive (Continued)

Primary Operational Status	Subsidiary Operational Status	Description
9 “Stopping”		Disk Drive is being disabled.
10 “Stopped”		Disk Drive is disabled.

Table 264 shows the relationship between the EnabledState of a disk drive to the drives OperationalStatus and the disk drive StorageExtent OperationalStatus.

Table 264 - 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

11.3 Cascading Considerations

Not defined in this standard.

11.4 Supported Profiles, Subprofiles and Packages

Related Profiles for Disk Drive Lite: Not defined in this standard.

11.5 Methods of this Profile

11.5.1 Extrinsic Methods on Disk Drives

11.5.1.1 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.

11.6 Registered Name and Version

Disk Drive Lite version 1.6.0 (Component Profile)

CIM Schema Version: 2.29

11.7 CIM Elements

Table 265 describes the CIM elements for Disk Drive Lite.

Table 265 - CIM Elements for Disk Drive Lite

Element Name	Requirement	Description
11.7.1 CIM_ATAPort (Disk Drive Target ATA Port)	Optional	Represents an ATA target port for the disk drive.
11.7.2 CIM_ATAProtocolEndpoint (Disk Drive target ATA Protocol Endpoint)	Optional	A target ATA protocol endpoint for a disk drive if ATA protocols are supported.
11.7.3 CIM_AssociatedComponentExtent (Pool Component to Primordial Pool)	Mandatory	
11.7.4 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.
11.7.5 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.
11.7.6 CIM_Container	Optional	Associates a disk drive physical package to its higher level package.
11.7.7 CIM_DeviceSAPImplementation (ATA)	Optional	Associates a target ATA protocol endpoint to the target port for the drive.
11.7.8 CIM_DeviceSAPImplementation (SCSI)	Optional	Associates a target SCSI protocol endpoint to the target port for the drive.
11.7.9 CIM_DiskDrive	Mandatory	Represents the disk drive.
11.7.10 CIM_ElementSoftwareIdentity	Mandatory	Associates the firmware (SoftwareIdentity) to a disk drive.
11.7.11 CIM_FCPort (Disk Drive Target FC Port)	Optional	Represents an FC target port for the disk drive.

Table 265 - CIM Elements for Disk Drive Lite

Element Name	Requirement	Description
11.7.12 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.
11.7.13 CIM_FilterCollection (Disk Drive Lite ProfileSpecificLifecycleIndicationFilterCollection)	Optional	Experimental. This is a collection of Lifecycle IndicationFilters defined by the Disk Drive Lite Profile.
11.7.14 CIM_HostedCollection (System to ProfileSpecificLifecycleIndicationFilterCollection)	Optional	Experimental.
11.7.15 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).
11.7.16 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.
11.7.17 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.
11.7.18 CIM_MediaPresent	Mandatory	Associates a disk drive to its storage extent.
11.7.19 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).

Table 265 - CIM Elements for Disk Drive Lite

Element Name	Requirement	Description
11.7.20 CIM_MemberOfCollection (Disk Drive Lite ProfileSpecificLifecycleIndicationFilterCollection to Disk Drive Lite Filters)	Optional	Experimental. This associates the Disk Drive Lite ProfileSpecificLifecycleIndicationFilterCollection to the Filters defined by the Disk Drive Lite Profile.
11.7.21 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.
11.7.22 CIM_PhysicalPackage	Mandatory	The physical package for the disk drive.
11.7.23 CIM_ProtocolControllerAccessesUnit	Optional	Deprecated. Associates an initiator protocol controller to the disk drive storage extent.
11.7.24 CIM_Realizes	Mandatory	Associates the disk drive to its physical package.
11.7.25 CIM_SAPAvailableForElement	Optional	Associates the target protocol endpoint to the disk drive.
11.7.26 CIM_SASPort (Disk Drive Target SAS Port)	Optional	Represents a SAS target port for the disk drive.
11.7.27 CIM_SCSIInitiatorTargetLogicalUnitPath	Optional	Associates protocol endpoints of the initiator and target ports to the extent that is exposed through the ports.
11.7.28 CIM_SCSIProtocolEndpoint (Disk Drive target SCSI Protocol Endpoint)	Optional	A target SCSI protocol endpoint for a disk drive if SCSI protocols are supported.
11.7.29 CIM_SPIPort (Disk Drive Target Parallel SCSI Port)	Optional	Represents a parallel SCSI target port for the disk drive.
11.7.30 CIM_SoftwareIdentity	Mandatory	Represents the firmware information for the disk drive.
11.7.31 CIM_StorageExtent (Primordial Disk Drive Extent)	Mandatory	The storage extent that represents the storage of the disk drive.
11.7.32 CIM_SystemDevice (Disk Drive System)	Mandatory	Associates DiskDrive to a hosting computer system.
11.7.33 CIM_SystemDevice (Port System)	Optional	Associates disk drive Ports to a hosting computer system.
11.7.34 CIM_SystemDevice (Storage Extent System)	Mandatory	Associates a StorageExtent to a hosting computer system.

Table 265 - CIM Elements for Disk Drive Lite

Element Name	Requirement	Description
11.7.35 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.

11.7.1 CIM_ATAPort (Disk Drive Target ATA Port)

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

Table 266 describes class CIM_ATAPort (Disk Drive Target ATA Port).

Table 266 - SMI Referenced Properties/Methods for CIM_ATAPort (Disk Drive Target ATA Port)

Properties	Flags	Requirement	Description & 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) .

11.7.2 CIM_ATAProtocolEndpoint (Disk Drive target ATA Protocol Endpoint)

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

Table 267 describes class CIM_ATAProtocolEndpoint (Disk Drive target ATA Protocol Endpoint).

Table 267 - SMI Referenced Properties/Methods for CIM_ATAProtocolEndpoint (Disk Drive target ATA Protocol Endpoint)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
Name		Mandatory	
Role		Mandatory	Shall be 3 (Target).
ProtocolIFType		Mandatory	
OtherTypeDescription		Mandatory	
ConnectionType		Mandatory	

11.7.3 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 268 describes class CIM_AssociatedComponentExtent (Pool Component to Primordial Pool).

Table 268 - SMI Referenced Properties/Methods for CIM_AssociatedComponentExtent (Pool Component to Primordial Pool)

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	The Primordial StoragePool.
PartComponent		Mandatory	The disk drive storage extent that is a component of the primordial storage pool.

11.7.4 CIM_BasedOn (Bottom Level BasedOn)

Created By: External
Modified By: External
Deleted By: External
Requirement: Implementation of the Extent Composition profile.

Table 269 describes class CIM_BasedOn (Bottom Level BasedOn).

Table 269 - SMI Referenced Properties/Methods for CIM_BasedOn (Bottom Level BasedOn)

Properties	Flags	Requirement	Description & 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.

11.7.5 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 270 describes class CIM_ConcreteComponent (Disk Extent to Primordial Pool).

Table 270 - SMI Referenced Properties/Methods for CIM_ConcreteComponent (Disk Extent to Primordial Pool)

Properties	Flags	Requirement	Description & 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.

11.7.6 CIM_Container

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 271 describes class CIM_Container.

Table 271 - SMI Referenced Properties/Methods for CIM_Container

Properties	Flags	Requirement	Description & 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.

11.7.7 CIM_DeviceSAPImplementation (ATA)

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

Table 272 describes class CIM_DeviceSAPImplementation (ATA).

Table 272 - SMI Referenced Properties/Methods for CIM_DeviceSAPImplementation (ATA)

Properties	Flags	Requirement	Description & 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).

11.7.8 CIM_DeviceSAPImplementation (SCSI)

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

Table 273 describes class CIM_DeviceSAPImplementation (SCSI).

Table 273 - SMI Referenced Properties/Methods for CIM_DeviceSAPImplementation (SCSI)

Properties	Flags	Requirement	Description & 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).

11.7.9 CIM_DiskDrive

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

Table 274 describes class CIM_DiskDrive.

Table 274 - SMI Referenced Properties/Methods for CIM_DiskDrive

Properties	Flags	Requirement	Description & 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	

11.7.10 CIM_ElementSoftwareIdentity

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 275 describes class CIM_ElementSoftwareIdentity.

Table 275 - SMI Referenced Properties/Methods for CIM_ElementSoftwareIdentity

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	A reference to an instance of CIM_SoftwareIdentity that represents the software the disk drive.
Dependent		Mandatory	A reference to an instance of CIM_DiskDrive or SNIA_diskdrive.

11.7.11 CIM_FCPort (Disk Drive Target FC Port)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 276 describes class CIM_FCPort (Disk Drive Target FC Port).

Table 276 - SMI Referenced Properties/Methods for CIM_FCPort (Disk Drive Target FC Port)

Properties	Flags	Requirement	Description & 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	

11.7.12 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 277 describes class CIM_FilterCollection (Disk Drive Lite Predefined FilterCollection).

Table 277 - SMI Referenced Properties/Methods for CIM_FilterCollection (Disk Drive Lite Predefined FilterCollection)

Properties	Flags	Requirement	Description & 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:Predefined'.

11.7.13 CIM_FilterCollection (Disk Drive Lite ProfileSpecificLifecycleIndicationFilterCollection)

Experimental. This is a collection of Lifecycle IndicationFilters defined by the Disk Drive Lite Profile.

Requirement: Optional

Table 278 describes class CIM_FilterCollection (Disk Drive Lite ProfileSpecificLifecycleIndicationFilterCollection).

Table 278 - SMI Referenced Properties/Methods for CIM_FilterCollection (Disk Drive Lite ProfileSpecificLifecycleIndicationFilterCollection)

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	See the InstanceID definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.13 CIM_FilterCollection (StaticFilterCollection).
CollectionName		Mandatory	The value of CollectionName shall be 'SNIA:Disk Drive Lite:ProfileSpecificLifecycleIndicationFilterCollection'.

11.7.14 CIM_HostedCollection (System to ProfileSpecificLifecycleIndicationFilterCollection)

Experimental.

Requirement: Optional

Table 279 describes class CIM_HostedCollection (System to ProfileSpecificLifecycleIndicationFilterCollection).

Table 279 - SMI Referenced Properties/Methods for CIM_HostedCollection (System to Profile-SpecificLifecycleIndicationFilterCollection)

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	Reference to the ProfileSpecificLifecycleIndicationFilterCollection for the Disk Drive Lite Profile.
Antecedent		Mandatory	Reference to the 'Top level' System.

11.7.15 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 280 describes class CIM_HostedCollection (System to predefined IndicationFilters).

Table 280 - SMI Referenced Properties/Methods for CIM_HostedCollection (System to predefined IndicationFilters)

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	Reference to the predefined FilterCollection for Disk Drive Lite.
Antecedent		Mandatory	Reference to the 'Top level' System.

11.7.16 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 281 describes class CIM_IndicationFilter (Disk Drive Creation).

Table 281 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Disk Drive Creation)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Name		Mandatory	This shall be 'SNIA:Disk Drive Lite:DiskDriveCreation'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Optional	Experimental. See the SourceNamespaces definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
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	See the ElementName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).

11.7.17 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 282 describes class CIM_IndicationFilter (Disk Drive Deletion).

Table 282 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Disk Drive Deletion)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Name		Mandatory	This shall be 'SNIA:Disk Drive Lite:DiskDriveDeletion'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Optional	Experimental. See the SourceNamespaces definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
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	See the ElementName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).

11.7.18 CIM_MediaPresent

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

Table 283 describes class CIM_MediaPresent.

Table 283 - SMI Referenced Properties/Methods for CIM_MediaPresent

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	A reference to an instance of CIM_StorageExtent with the Primordial propery set to true (a disk drive extent).
Antecedent		Mandatory	A reference to an instance of CIM_DiskDrive or SNIA_DiskDrive.

11.7.19 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 284 describes class CIM_MemberOfCollection (Disk Drive Lite Filter Collection to FilterCollection).

Table 284 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Disk Drive Lite Filter Collection to FilterCollection)

Properties	Flags	Requirement	Description & Notes
Collection		Mandatory	Reference to the Disk Drive Lite predefined FilterCollection.
Member		Mandatory	Reference to the Disk Drive Lite predefined FilterCollection.

11.7.20 CIM_MemberOfCollection (Disk Drive Lite ProfileSpecificLifecycleIndicationFilterCollection to Disk Drive Lite Filters)

Experimental. This associates the Disk Drive Lite ProfileSpecificLifecycleIndicationFilterCollection to the Filters defined by the Disk Drive Lite Profile.

Requirement: Optional

Table 285 describes class CIM_MemberOfCollection (Disk Drive Lite ProfileSpecificLifecycleIndicationFilterCollection to Disk Drive Lite Filters).

Table 285 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Disk Drive Lite ProfileSpecificLifecycleIndicationFilterCollection to Disk Drive Lite Filters)

Properties	Flags	Requirement	Description & Notes
Collection		Mandatory	Reference to the Disk Drive Lite ProfileSpecificLifecycleIndicationFilterCollection.
Member		Mandatory	Reference to the IndicationFilters defined by the Disk Drive Lite Profile.

11.7.21 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 286 describes class CIM_MemberOfCollection (Predefined Filter Collection to Disk Drive Lite Filters).

Table 286 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Predefined Filter Collection to Disk Drive Lite Filters)

Properties	Flags	Requirement	Description & Notes
Collection		Mandatory	Reference to the Disk Drive Lite predefined FilterCollection.
Member		Mandatory	Reference to the predefined IndicationFilters of the Disk Drive Lite implementation.

11.7.22 CIM_PhysicalPackage

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

Table 287 describes class CIM_PhysicalPackage.

Table 287 - SMI Referenced Properties/Methods for CIM_PhysicalPackage

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

11.7.23 CIM_ProtocolControllerAccessesUnit

Deprecated.

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

Table 288 describes class CIM_ProtocolControllerAccessesUnit.

Table 288 - SMI Referenced Properties/Methods for CIM_ProtocolControllerAccessesUnit

Properties	Flags	Requirement	Description & 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).

11.7.24 CIM_Realizes

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

Table 289 describes class CIM_Realizes.

Table 289 - SMI Referenced Properties/Methods for CIM_Realizes

Properties	Flags	Requirement	Description & 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.

11.7.25 CIM_SAPAvailableForElement

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

Table 290 describes class CIM_SAPAvailableForElement.

Table 290 - SMI Referenced Properties/Methods for CIM_SAPAvailableForElement

Properties	Flags	Requirement	Description & 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.

11.7.26 CIM_SASPort (Disk Drive Target SAS Port)

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

Table 291 describes class CIM_SASPort (Disk Drive Target SAS Port).

Table 291 - SMI Referenced Properties/Methods for CIM_SASPort (Disk Drive Target SAS Port)

Properties	Flags	Requirement	Description & 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).

11.7.27 CIM_SCSIInitiatorTargetLogicalUnitPath

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

Table 292 describes class CIM_SCSIInitiatorTargetLogicalUnitPath.

Table 292 - SMI Referenced Properties/Methods for CIM_SCSIInitiatorTargetLogicalUnitPath

Properties	Flags	Requirement	Description & 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.

11.7.28 CIM_SCSIProtocolEndpoint (Disk Drive target SCSI Protocol Endpoint)

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

Table 293 describes class CIM_SCSIProtocolEndpoint (Disk Drive target SCSI Protocol Endpoint).

Table 293 - SMI Referenced Properties/Methods for CIM_SCSIProtocolEndpoint (Disk Drive target SCSI Protocol Endpoint)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
Name		Mandatory	
Role		Mandatory	Shall be 3 (Target).
ProtocolIFType		Mandatory	
OtherTypeDescription		Mandatory	
ConnectionType		Mandatory	

11.7.29 CIM_SPIPort (Disk Drive Target Parallel SCSI Port)

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

Table 294 describes class CIM_SPIPort (Disk Drive Target Parallel SCSI Port).

Table 294 - SMI Referenced Properties/Methods for CIM_SPIPort (Disk Drive Target Parallel SCSI Port)

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

Table 294 - SMI Referenced Properties/Methods for CIM_SPIPort (Disk Drive Target Parallel SCSI Port)

Properties	Flags	Requirement	Description & Notes
OperationalStatus		Optional	
UsageRestriction		Mandatory	Shall be 2 for disk drive target ports.
PortType		Mandatory	Shall be 101 (SCSI Parallel).

11.7.30 CIM_SoftwareIdentity

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

Table 295 describes class CIM_SoftwareIdentity.

Table 295 - SMI Referenced Properties/Methods for CIM_SoftwareIdentity

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
VersionString		Mandatory	
Manufacturer		Optional	
BuildNumber		Optional	
MajorVersion		Optional	
RevisionNumber		Optional	
MinorVersion		Optional	

11.7.31 CIM_StorageExtent (Primordial Disk Drive Extent)

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

Table 296 describes class CIM_StorageExtent (Primordial Disk Drive Extent).

Table 296 - SMI Referenced Properties/Methods for CIM_StorageExtent (Primordial Disk Drive Extent)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	
BlockSize		Mandatory	
NumberOfBlocks		Mandatory	The number of blocks as reported by the hardware.
ConsumableBlocks		Mandatory	The number of usable blocks.
Primordial		Mandatory	Shall be true.
ExtentStatus		Mandatory	
OperationalStatus		Mandatory	
ExtentDiscriminator		Mandatory	Experimental. This is an array of values that shall contain 'SNIA:Pool Component' and 'SNIA:DiskDrive'.

11.7.32 CIM_SystemDevice (Disk Drive System)

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

Table 297 describes class CIM_SystemDevice (Disk Drive System).

Table 297 - SMI Referenced Properties/Methods for CIM_SystemDevice (Disk Drive System)

Properties	Flags	Requirement	Description & 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.

11.7.33 CIM_SystemDevice (Port System)

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

Requirement: Optional

Table 298 describes class CIM_SystemDevice (Port System).

Table 298 - SMI Referenced Properties/Methods for CIM_SystemDevice (Port System)

Properties	Flags	Requirement	Description & 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.

11.7.34 CIM_SystemDevice (Storage Extent System)

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

Table 299 describes class CIM_SystemDevice (Storage Extent System).

Table 299 - SMI Referenced Properties/Methods for CIM_SystemDevice (Storage Extent System)

Properties	Flags	Requirement	Description & 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.

11.7.35 SNIA_DiskDrive

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

Table 300 describes class SNIA_DiskDrive.

Table 300 - SMI Referenced Properties/Methods for SNIA_DiskDrive

Properties	Flags	Requirement	Description & 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).

Table 300 - SMI Referenced Properties/Methods for SNIA_DiskDrive

Properties	Flags	Requirement	Description & Notes
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 propery 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

Clause 12: Disk Sparing Subprofile

12.1 Description

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 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 62: "Sparing Instance Diagram".

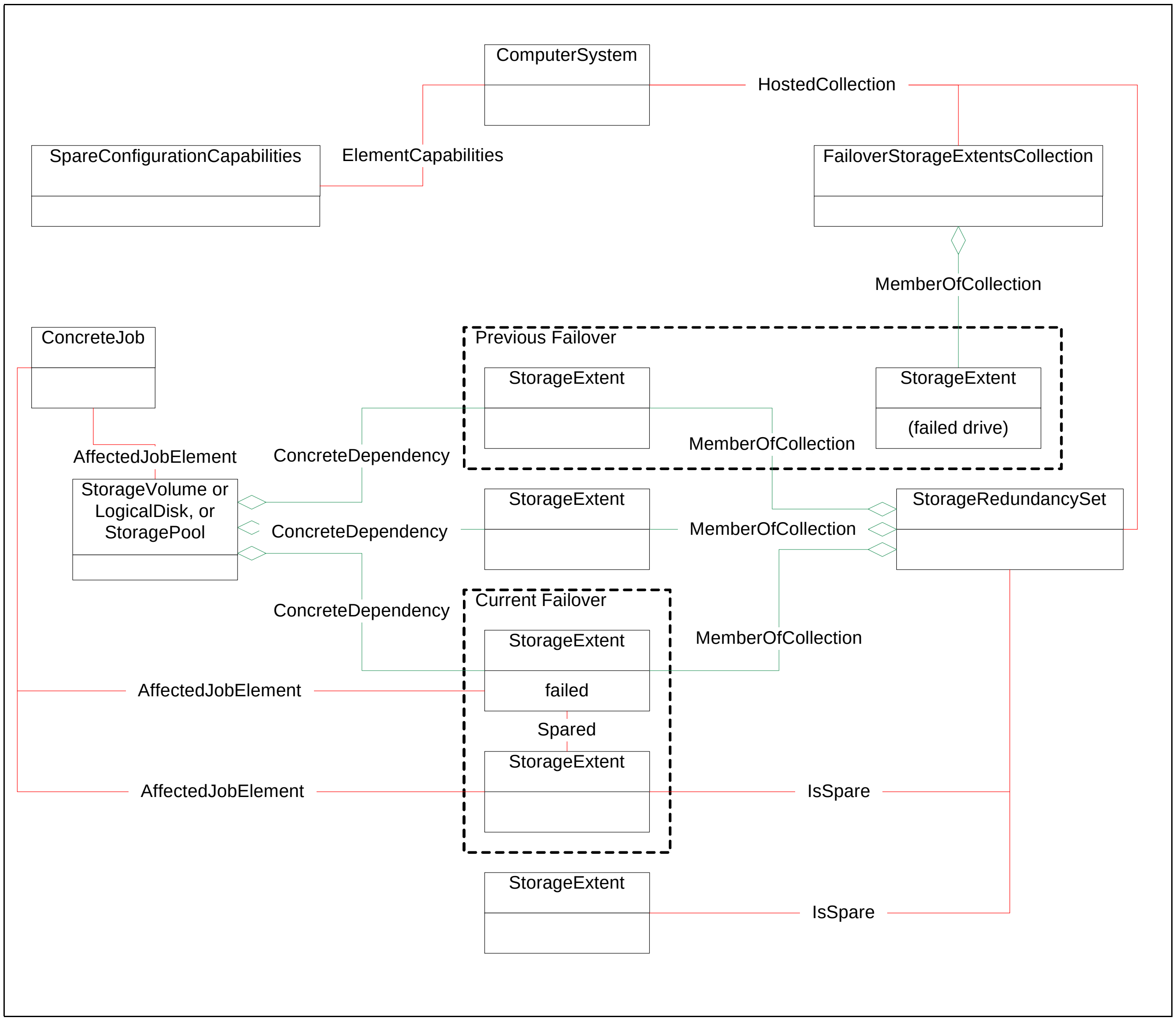

Figure 62 - Sparing Instance Diagram

Clause 14: "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.

Clause 11: "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 *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* Clause 25: 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.

12.1.1 Durable Names and Correlatable IDs of the Profile

The StorageVolumes are required to provide the correlatable ID, Name. See *Storage Management Technical Specification, Part 1 Common Architecture, 1.6.0 Rev 4* 7.2, "Guidelines for SCSI Logical Unit Names".

12.1.2 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 StorageRedudancySet. 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 63: "Variations of RS per Storage Element".

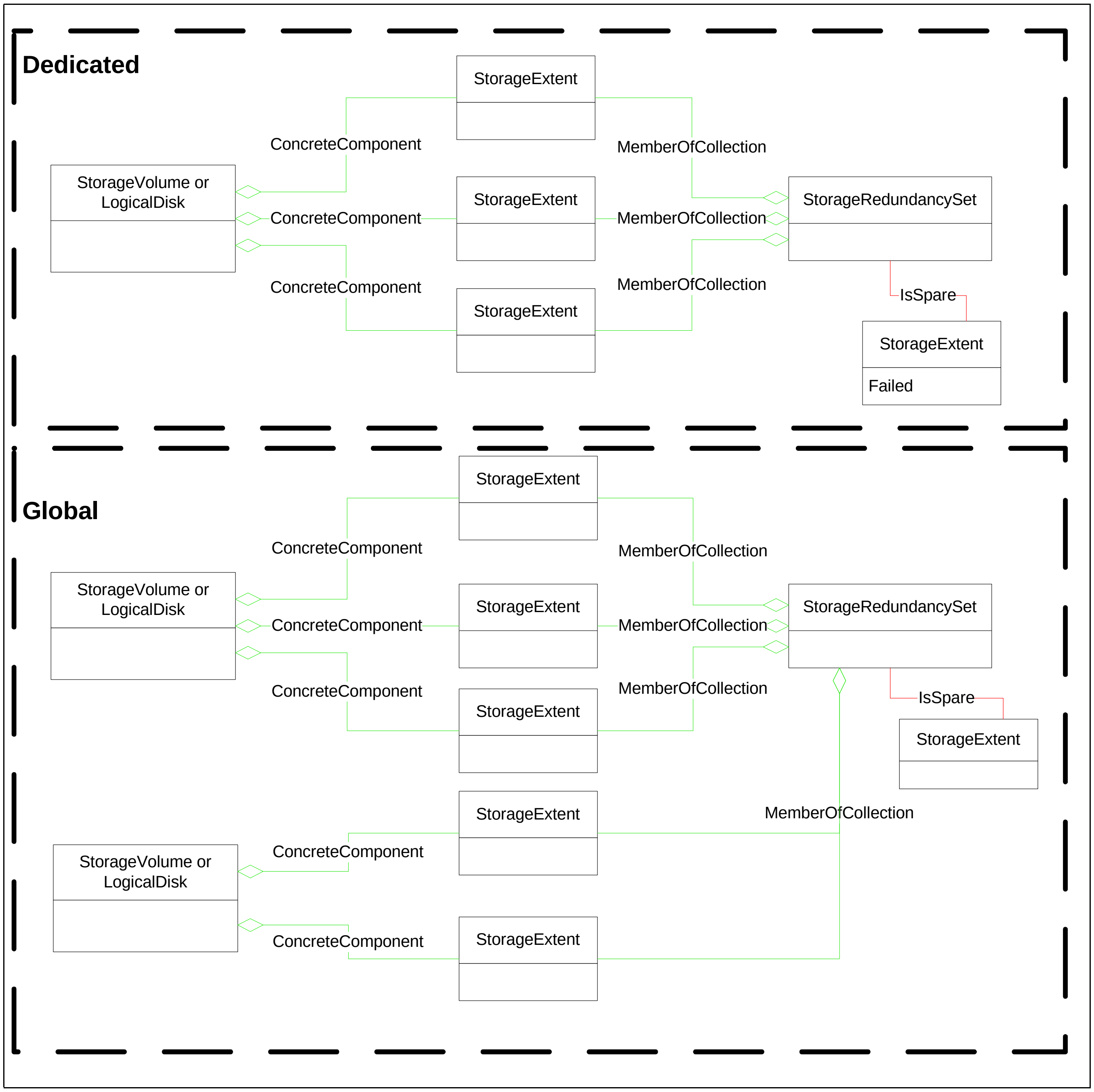

Figure 63 - Variations of RS per Storage Element

In the case where spares are not dedicated, the decision to group Extents with a given StorageRedundancySet depends of 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 5:, "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 in removed from the system through a means not defined in this profile, i.e., the drive FRU 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 301 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 SupportedSynchronousActions array. If a sparing method is supported asynchronously, then the action name for the method shall be present in SupportedAsynchronousActions array.

Table 301 - 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

12.1.3 Modeling Fail Over, Past and Present

This section illustrates the requirements for modeling spare fail over in three cases, before the failure, during the fail over, and after the fail over.

Figure 64: "Before Failure" shows a dedicated RedundancySet with a single spare.

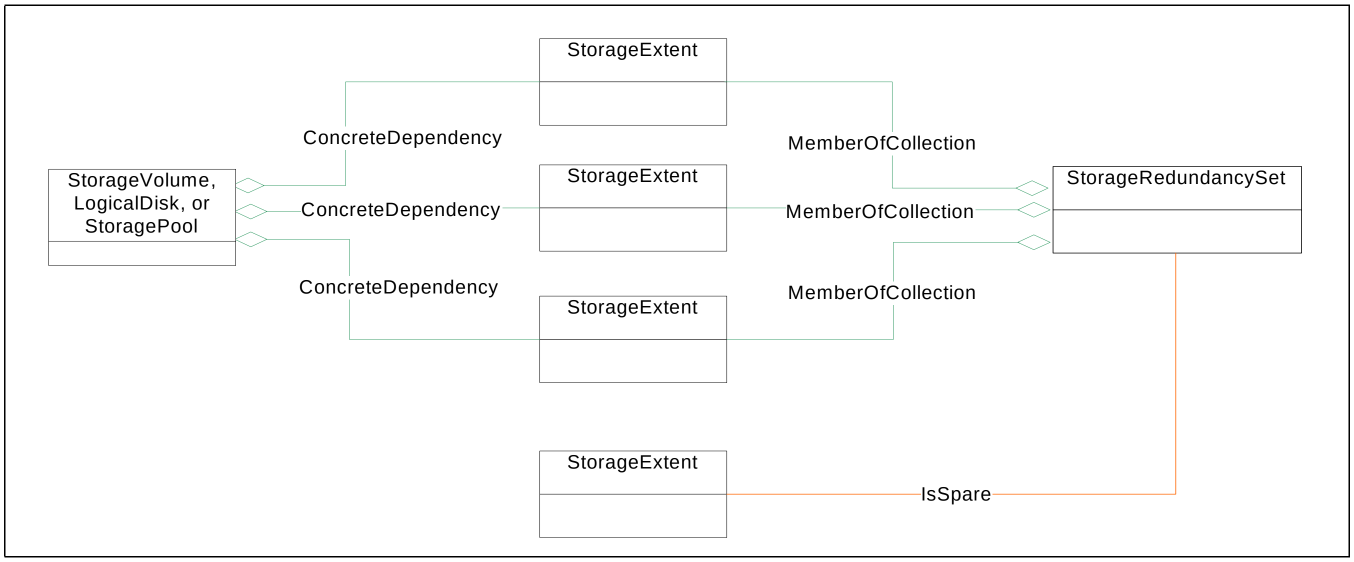

Figure 64 - Before Failure

Once the failure has occurred, a ConcreteJob is created to represent the fail over process, as shown in Figure 65: "During Failure".

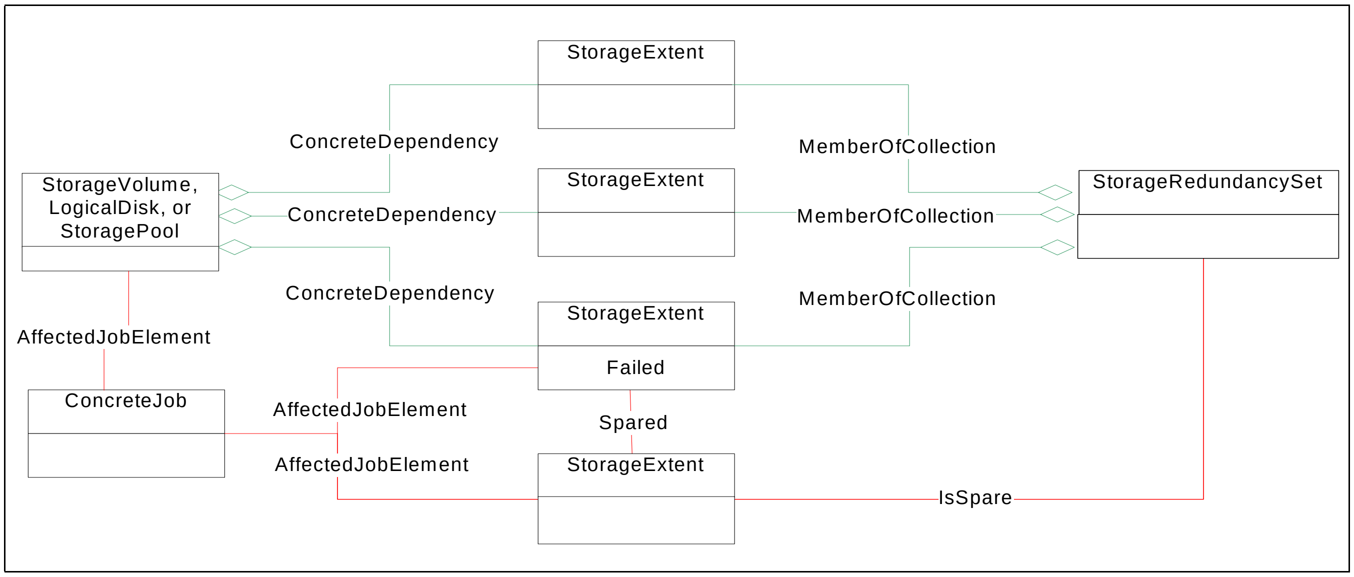

Figure 65 - During Failure

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 *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* Clause 26: "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 *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* Clause 26: "Job Control Subprofile", shall contain the StorageRedundancySet.Failover() method name and parameters.

Once the fail over is complete, the failed Extent shall no longed 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 66: "After Failure".

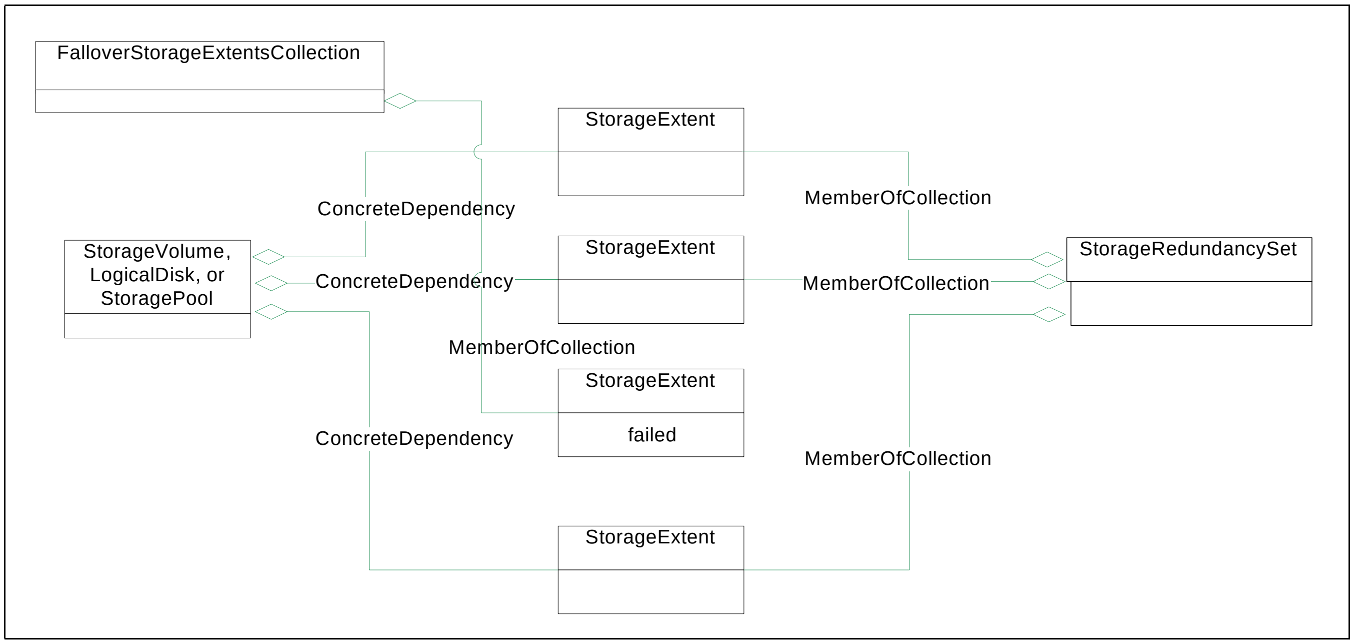

Figure 66 - After Failure

EXPERIMENTAL

12.1.4 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 *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* Clause 26: "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 to allow an observer to tell that, for example, a RepairParity task is executing.

EXPERIMENTAL

12.2 Health and Fault Management Considerations

One of the primary reasons for this subprofile 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.

12.3 Cascading Conjurations

Not defined in this standard.

12.4 Supported Subprofiles and Packages

Table 302 describes the supported profiles for Disk Sparing.

Table 302 - Supported Profiles for Disk Sparing

Profile Name	Organization	Version	Requirement	Description
Job Control	SNIA	1.5.0	Mandatory	

12.5 Methods of the Profile

EXPERIMENTAL

12.5.1 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 follow 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.

12.5.2 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.

12.5.3 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.

12.5.4 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.

12.5.5 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.

12.5.6 CheckParityConsistency

```
uint32 CheckParityConsistency(
                           [Out] CIM_ConcreteJob REF Job
                           [In] CIM_StorageExtent REF Target)
```

This method is used to check 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.

12.5.7 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.

12.5.8 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

12.6 Client Considerations and Recipes

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 remove 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 (12.6.1) for details.

12.6.1 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 occured
            before this point in the code.>
```

12.7 Registered Name and Version

Disk Sparing version 1.5.0 (Component Profile)

12.8 CIM Elements

Table 303 describes the CIM elements for Disk Sparing.

Table 303 - CIM Elements for Disk Sparing

Element Name	Requirement	Description
12.8.1 CIM_AssociatedComponentExtent (Spare to Storage Pool)	Conditional	Conditional requirement: Implementation of the Extent Composition profile.
12.8.2 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.
12.8.3 CIM_ConcreteDependency (Extent to Pool)	Mandatory	Represents the group of StorageExtents that form the redundancy of a StoragePool.
12.8.4 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.

Table 303 - CIM Elements for Disk Sparing

Element Name	Requirement	Description
12.8.5 CIM_ElementCapabilities	Optional	Associates SpareConfigurationCapabilities with the Block Server's ComputerSystem instance.
12.8.6 CIM_HostedCollection (ComputerSystem to FailoverStorageExtentsCollection)	Optional	Associates FailoverStorageExtentsCollection with the Block Server's ComputerSystem instance.
12.8.7 CIM_HostedCollection (ComputerSystem to RedundancySet)	Mandatory	Associates StorageRedundancySet with the Block Server's ComputerSystem instance.
12.8.8 CIM_HostedService (ComputerSystem to SpareConfigurationService)	Optional	Associates SpareConfigurationService with the Block Server's ComputerSystem instance.
12.8.9 CIM_IsSpare	Mandatory	Represents the spare that may be used as a spare for any StorageExtents that is not a spare.
12.8.10 CIM_LogicalDisk	Conditional	Conditional requirement: Referenced from Volume Management - LogicalDisk is mandatory.
12.8.11 CIM_MemberOfCollection	Mandatory	Represents the relationship between the StorageExtents that form the redundancy of a StoragePool, StorageVolume, or LogicalDisk.
12.8.12 CIM_Spared	Mandatory	Represents the relationship between the spare and the StorageExtent that has failed and is being spared.
12.8.13 CIM_StorageExtent (Spare)	Mandatory	Represents the redundant or spare capacity.
12.8.14 CIM_StoragePool	Mandatory	Elements to Primordial and Concrete Pools.
12.8.15 CIM_StorageRedundancySet	Mandatory	Represents the group of spare StorageExtents and StorageExtents that these spares will substitute for case of failure.
12.8.16 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).
12.8.17 SNIA_FailoverStorageExtentsCollection	Optional	The collection of StorageExtents that have failed.
12.8.18 SNIA_SpareConfigurationCapabilities	Optional	Instances of this class define the behavior supported by this sparing implementation.
12.8.19 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.

12.8.1 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 304 describes class CIM_AssociatedComponentExtent (Spare to Storage Pool).

Table 304 - SMI Referenced Properties/Methods for CIM_AssociatedComponentExtent (Spare to Storage Pool)

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	The StoragePool.
PartComponent		Mandatory	The spare storage extent that is a component of the storage pool.

12.8.2 CIM_ConcreteDependency (Extent to LogicalDisk)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Referenced from Volume Management - LogicalDisk is mandatory.

Table 305 describes class CIM_ConcreteDependency (Extent to LogicalDisk).

Table 305 - SMI Referenced Properties/Methods for CIM_ConcreteDependency (Extent to LogicalDisk)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	An underlying Storage Extent.
Dependent		Mandatory	A Logical Disk.

12.8.3 CIM_ConcreteDependency (Extent to Pool)

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

Table 306 describes class CIM_ConcreteDependency (Extent to Pool).

Table 306 - SMI Referenced Properties/Methods for CIM_ConcreteDependency (Extent to Pool)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

12.8.4 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 307 describes class CIM_ConcreteDependency (Extent to StorageVolume).

Table 307 - SMI Referenced Properties/Methods for CIM_ConcreteDependency (Extent to StorageVolume)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	An underlying primordial Extent.
Dependent		Mandatory	A StorageVolume.

12.8.5 CIM_ElementCapabilities

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

Table 308 describes class CIM_ElementCapabilities.

Table 308 - SMI Referenced Properties/Methods for CIM_ElementCapabilities

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	The hosting System.
Capabilities		Mandatory	The support spare configuration capabilities.

12.8.6 CIM_HostedCollection (ComputerSystem to FailoverStorageExtentsCollection)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 309 describes class CIM_HostedCollection (ComputerSystem to FailoverStorageExtentsCollection).

Table 309 - SMI Referenced Properties/Methods for CIM_HostedCollection (ComputerSystem to FailoverStorageExtentsCollection)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	The hosting System.
Dependent		Mandatory	Indicates which FailoverStorageExtentsCollection are part of Disk Sparing implementation.

12.8.7 CIM_HostedCollection (ComputerSystem to RedundancySet)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 310 describes class CIM_HostedCollection (ComputerSystem to RedundancySet).

Table 310 - SMI Referenced Properties/Methods for CIM_HostedCollection (ComputerSystem to RedundancySet)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	The hosting System.
Dependent		Mandatory	Indicate which StorageRedundancySets are part of Disk Sparing implementation.

12.8.8 CIM_HostedService (ComputerSystem to SpareConfigurationService)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 311 describes class CIM_HostedService (ComputerSystem to SpareConfigurationService).

Table 311 - SMI Referenced Properties/Methods for CIM_HostedService (ComputerSystem to SpareConfigurationService)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	The hosting System.
Dependent		Mandatory	The support spare configuration service.

12.8.9 CIM_IsSpare

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

Table 312 describes class CIM_IsSpare.

Table 312 - SMI Referenced Properties/Methods for CIM_IsSpare

Properties	Flags	Requirement	Description & Notes
SpareStatus		Mandatory	
FailoverSupported		Mandatory	
Antecedent		Mandatory	A Spare Storage Extent.
Dependent		Mandatory	

12.8.10 CIM_LogicalDisk

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Referenced from Volume Management - LogicalDisk is mandatory.

Table 313 describes class CIM_LogicalDisk.

Table 313 - SMI Referenced Properties/Methods for CIM_LogicalDisk

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

Table 313 - SMI Referenced Properties/Methods for CIM_LogicalDisk

Properties	Flags	Requirement	Description & Notes
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.

12.8.11 CIM_MemberOfCollection

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

Table 314 describes class CIM_MemberOfCollection.

Table 314 - SMI Referenced Properties/Methods for CIM_MemberOfCollection

Properties	Flags	Requirement	Description & Notes
Member		Mandatory	
Collection		Mandatory	

12.8.12 CIM_Spared

Created By: Static
Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 315 describes class CIM_Spared.

Table 315 - SMI Referenced Properties/Methods for CIM_Spared

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	A reference to the StorageExtent that as replaced another StorageExtent.
Dependent		Mandatory	The StorageExtent that has failed and is being replaced.

12.8.13 CIM_StorageExtent (Spare)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 316 describes class CIM_StorageExtent (Spare).

Table 316 - SMI Referenced Properties/Methods for CIM_StorageExtent (Spare)

Properties	Flags	Requirement	Description & 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.

12.8.14 CIM_StoragePool

Requirement: Mandatory

Table 317 describes class CIM_StoragePool.

Table 317 - SMI Referenced Properties/Methods for CIM_StoragePool

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
ElementName		Mandatory	
PoolID		Mandatory	A unique name in the context of this system that identifies this Pool.

12.8.15 CIM_StorageRedundancySet

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

Table 318 describes class CIM_StorageRedundancySet.

Table 318 - SMI Referenced Properties/Methods for CIM_StorageRedundancySet

Properties	Flags	Requirement	Description & 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.

12.8.16 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 319 describes class CIM_StorageVolume.

Table 319 - SMI Referenced Properties/Methods for CIM_StorageVolume

Properties	Flags	Requirement	Description & 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.

12.8.17 SNIA_FailoverStorageExtentsCollection

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 320 describes class SNIA_FailoverStorageExtentsCollection.

Table 320 - SMI Referenced Properties/Methods for SNIA_FailoverStorageExtentsCollection

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
ElementName		Mandatory	User friendly name.

12.8.18 SNIA_SpareConfigurationCapabilities

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 321 describes class SNIA_SpareConfigurationCapabilities.

Table 321 - SMI Referenced Properties/Methods for SNIA_SpareConfigurationCapabilities

Properties	Flags	Requirement	Description & 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.
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.

12.8.19 SNIA_SpareConfigurationService

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 322 describes class SNIA_SpareConfigurationService.

Table 322 - SMI Referenced Properties/Methods for SNIA_SpareConfigurationService

Properties	Flags	Requirement	Description & 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

Clause 13: Erasure Profile

13.1 Description

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.

13.1.1 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.

Table 67 shows the new properties and method introduced by this subprofile. While a StorageVolume is shown, the same shall apply to LogicalDisk.

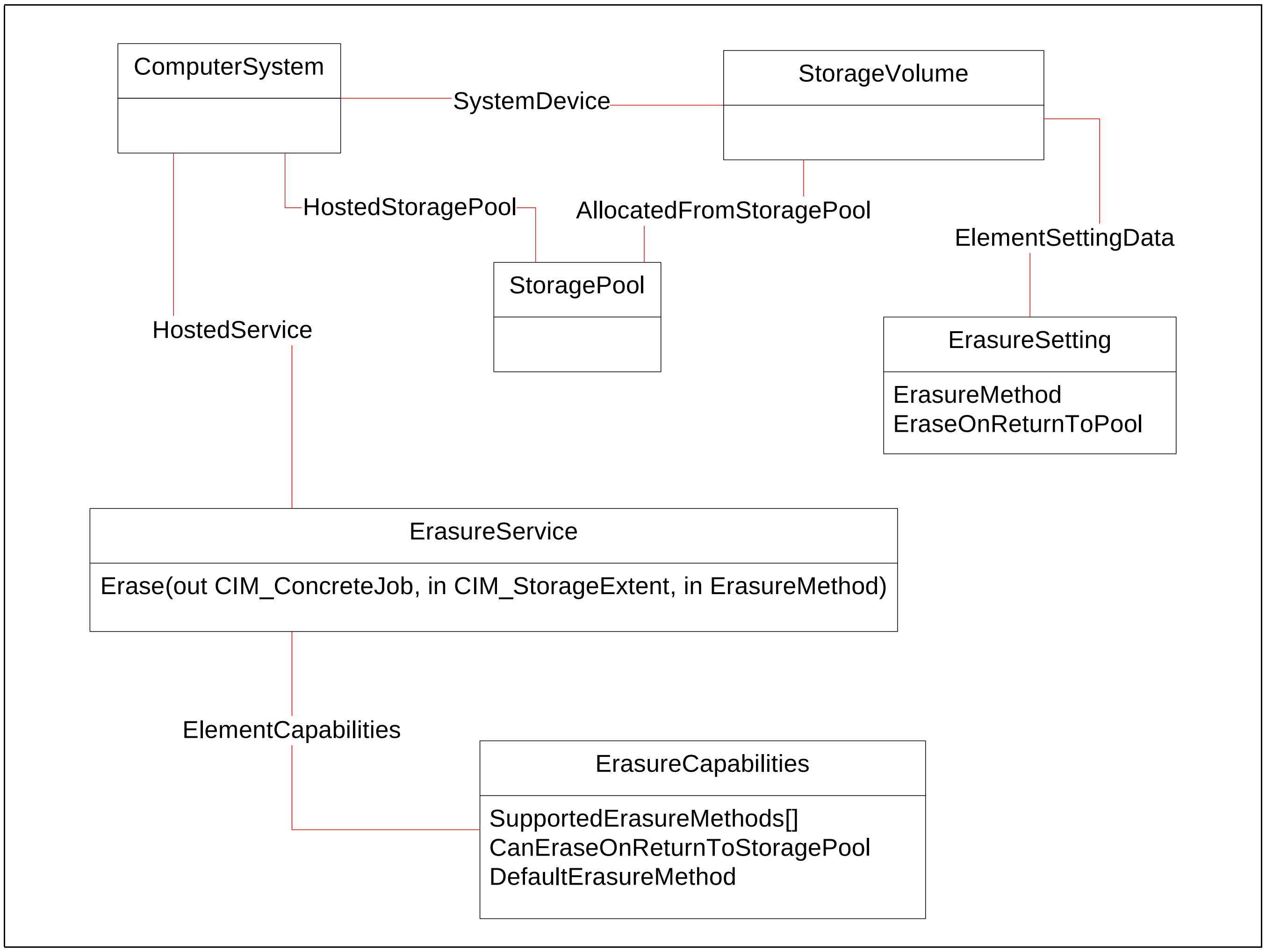

Figure 67 - Model Elements

13.2 Health and Fault Management Considerations

Not defined in this standard.

13.3 Cascading Considerations

Not applicable

13.4 Supported Profiles, Subprofiles, and Packages

Not defined in this standard.

13.5 Methods of the Profile

The Erase method 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 323 - 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	ErasureMethod	uint32	Erasure method to use

13.6 Client Considerations and Recipes

These cases can be generalized into the explicit case of "Volume Erasure" and the implicit case of "Volume Deletion".

13.6.1 Recipe 1: Volume Erasure

This is the case where it is determined that the contents of a storage volume must 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 #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. 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->)
}
```

13.6.2 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.ErasureMethod 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_ErasureSetting”,
                  null, null,
                  false, false, null)
if ($Setting[].length == 0) {
  // Create setting
  $TheSetting = newInstance(“SNIA_ErasureSetting”)
  $TheSetting.InstanceID = “SNIA:0001” // create unique ID
  $TheSetting.ErasureMethod = #ErasureMethod
  $TheSetting.EraseOnReturnToStoragePool = true
  $instance-> = CreateInstance($TheSetting)
}
else {
  $Setting[0].ErasureMethod = #ErasureMethod
  $Setting[0].EraseOnReturnToStoragePool = true
  ModifyInstance($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->)
}
```

13.7 Registered Name and Version

Erasure version 1.2.0 (Component Profile)

CIM Schema Version: 2.12.0

13.8 CIM Elements

Table 324 describes the CIM elements for Erasure.

Table 324 - CIM Elements for Erasure

Element Name	Requirement	Description
13.8.1 CIM_AllocatedFromStoragePool	Mandatory	AllocationFromStoragePool as defined in the Array Profile.
13.8.2 CIM_LogicalDisk	Conditional	Conditional requirement: Conditional
13.8.3 CIM_StoragePool	Mandatory	
13.8.4 CIM_StorageVolume	Conditional	Conditional requirement: Conditional
13.8.5 SNIA_ErasureCapabilities	Mandatory	
13.8.6 SNIA_ErasureService	Mandatory	
13.8.7 SNIA_ErasureSetting	Mandatory	

13.8.1 CIM_AllocatedFromStoragePool

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

Table 325 describes class CIM_AllocatedFromStoragePool.

Table 325 - SMI Referenced Properties/Methods for CIM_AllocatedFromStoragePool

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

13.8.2 CIM_LogicalDisk

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: null

Table 326 describes class CIM_LogicalDisk.

Table 326 - SMI Referenced Properties/Methods for CIM_LogicalDisk

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

13.8.3 CIM_StoragePool

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

Table 327 describes class CIM_StoragePool.

Table 327 - SMI Referenced Properties/Methods for CIM_StoragePool

Properties	Flags	Requirement	Description & Notes
Primordial		Mandatory	

Table 327 - SMI Referenced Properties/Methods for CIM_StoragePool

Properties	Flags	Requirement	Description & Notes
TotalManagedSpace		Mandatory	
RemainingManagedSpace		Mandatory	

13.8.4 CIM_StorageVolume

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: null

Table 328 describes class CIM_StorageVolume.

Table 328 - SMI Referenced Properties/Methods for CIM_StorageVolume

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

13.8.5 SNIA_ErasureCapabilities

Created By: Static
Requirement: Mandatory

Table 329 describes class SNIA_ErasureCapabilities.

Table 329 - SMI Referenced Properties/Methods for SNIA_ErasureCapabilities

Properties	Flags	Requirement	Description & Notes
ElementName		Mandatory	User friendly name for this instance of Capabilities.
InstanceID		Mandatory	Unique identifier for the instance.
ErasureMethods		Mandatory	Indicates erasure methods supported.
DefaultErasureMethod		Mandatory	Erasure method to use if none specified in the volume's setting.

Table 329 - SMI Referenced Properties/Methods for SNIA_ErasureCapabilities

Properties	Flags	Requirement	Description & Notes
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.

13.8.6 SNIA_ErasureService

Created By: Static
Requirement: Mandatory

Table 330 describes class SNIA_ErasureService.

Table 330 - SMI Referenced Properties/Methods for SNIA_ErasureService

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	The scoping System CreationClassName.
SystemName		Mandatory	The scoping System Name.
CreationClassName		Mandatory	The name of the concrete subclass.
Name		Mandatory	Unique identifer for the Service.
Erase()		Mandatory	This service contains the Erase method used to erase storage elements.

13.8.7 SNIA_ErasureSetting

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

Table 331 describes class SNIA_ErasureSetting.

Table 331 - SMI Referenced Properties/Methods for SNIA_ErasureSetting

Properties	Flags	Requirement	Description & Notes
ErasureMethod		Mandatory	Erasure method to use. Must 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

Clause 14: Extent Composition Subprofile

14.1 Description

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.

14.1.1 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.

14.1.2 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.

14.1.3 Model Element Summary

This subprofile uses the following CIM Classes:

LogicalDisk & 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 14.1.4.2).

DEPRECATED

AssociatedComponentExtent - Used to associate extents (Pool Component or Composite Pool Component) to their parent StoragePool (See 14.1.4.2).

StoragePool and AllocatedStoragePool are shown in instance diagrams for context but are part of the Block Service package Read Only sub-package.

Refer to 14.8 "CIM Elements" for detailed class descriptions.

14.1.4 Relation to other Packages and Subprofiles

14.1.4.1 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.

14.1.4.2 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 14.1.4.3).

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.

14.1.4.3 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.

14.1.4.4 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.

However, the Extent Composition Subprofile is capable of describing general organizations, such as heterogeneous, multi-layer RAID such as can be create by the Volume Management Profile. An example of this would be a RAID5 mirrored against a RAID0, a RAID(5,0)+1. Another example would be a three layer RAID organization such as a RAID10 where the bottom layer RAID1 members were concatenations of available extents.

14.1.5 Remaining Extents

When a StorageExtent (or CompositeExtent) is based on only part of an underlying storage extent (a partial allocation), the unused part of the underlying StorageExtent is represented by a Remaining StorageExtent. This is illustrated in Figure 68.

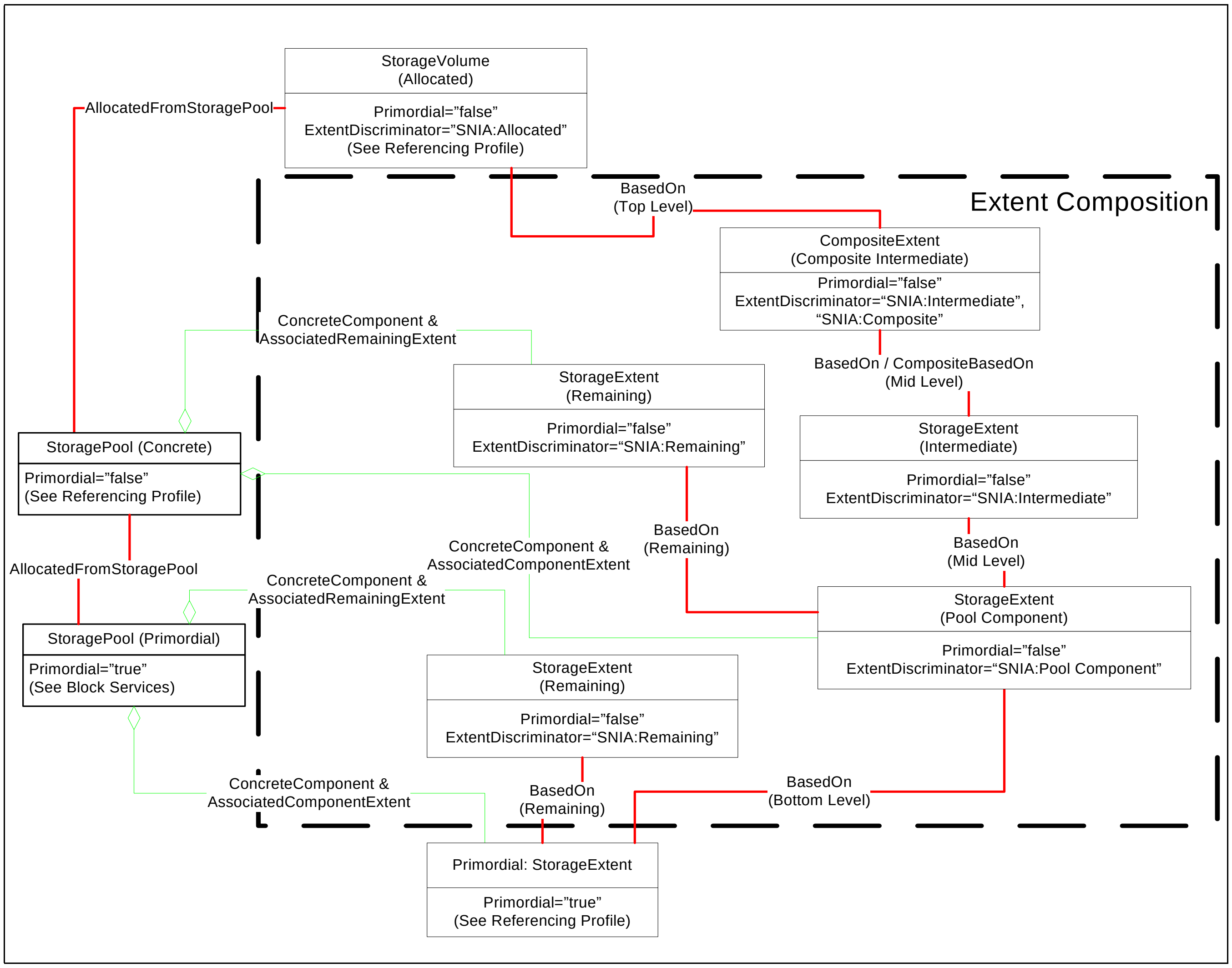
StorageVolume (Allocated)
Primordial="false"
ExtentDiscriminator="SNIA:Allocated"
(See Referencing Profile)
AllocatedFromStoragePool
BasedOn (Top Level)
Extent Composition
CompositeExtent (Composite Intermediate)
Primordial="false"
ExtentDiscriminator="SNIA:Intermediate", "SNIA:Composite"
ConcreteComponent & AssociatedRemainingExtent
StorageExtent (Remaining)
Primordial="false"
ExtentDiscriminator="SNIA:Remaining"
BasedOn / CompositeBasedOn (Mid Level)
StorageExtent (Intermediate)
Primordial="false"
ExtentDiscriminator="SNIA:Intermediate"
StoragePool (Concrete)
Primordial="false"
(See Referencing Profile)
BasedOn (Remaining)
BasedOn (Mid Level)
ConcreteComponent & AssociatedComponentExtent
AllocatedFromStoragePool
ConcreteComponent & AssociatedRemainingExtent
StorageExtent (Pool Component)
Primordial="false"
ExtentDiscriminator="SNIA:Pool Component"
StoragePool (Primordial)
Primordial="true"
(See Block Services)
StorageExtent (Remaining)
Primordial="false"
ExtentDiscriminator="SNIA:Remaining"
ConcreteComponent & AssociatedComponentExtent
BasedOn (Remaining)
BasedOn (Bottom Level)
Primordial: StorageExtent
Primordial="true"
(See Referencing Profile)

Figure 68 - Remaining Extents in Extent Composition

Figure 68 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 5.1.15 for extent conservation provisions.

14.1.6 Scenarios

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.

14.1.6.1 Volume Composition

Figure 69: "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 14.6.1) for this subprofile makes use of the decomposition function (i.e., complete consumption) to make this initial traversal.

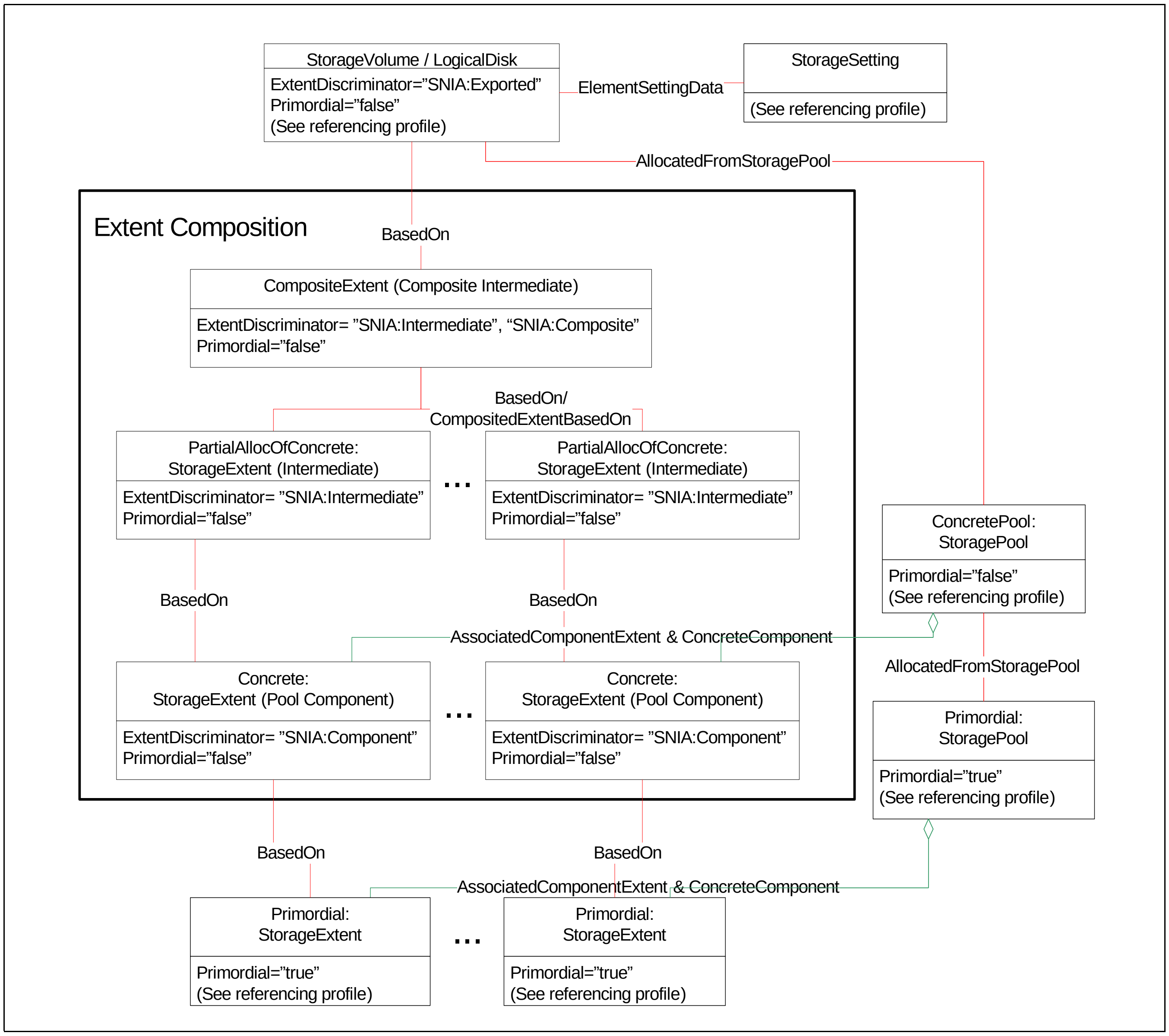
StorageVolume / LogicalDisk
ExtentDiscriminator="SNIA:Exported"
Primordial="false"
(See referencing profile)
ElementSettingData
StorageSetting
(See referencing profile)
AllocatedFromStoragePool
Extent Composition
BasedOn
CompositeExtent (Composite Intermediate)
ExtentDiscriminator= "SNIA:Intermediate", "SNIA:Composite"
Primordial="false"
BasedOn/
CompositedExtentBasedOn
PartialAllocOfConcrete:
StorageExtent (Intermediate)
ExtentDiscriminator= "SNIA:Intermediate"
Primordial="false"
...
PartialAllocOfConcrete:
StorageExtent (Intermediate)
ExtentDiscriminator= "SNIA:Intermediate"
Primordial="false"
ConcretePool:
StoragePool
Primordial="false"
(See referencing profile)
BasedOn
BasedOn
AssociatedComponentExtent & ConcreteComponent
AllocatedFromStoragePool
Concrete:
StorageExtent (Pool Component)
ExtentDiscriminator= "SNIA:Component"
Primordial="false"
...
Concrete:
StorageExtent (Pool Component)
ExtentDiscriminator= "SNIA:Component"
Primordial="false"
Primordial:
StoragePool
Primordial="true"
(See referencing profile)
BasedOn
BasedOn
AssociatedComponentExtent & ConcreteComponent
Primordial:
StorageExtent
Primordial="true"
(See referencing profile)
...
Primordial:
StorageExtent
Primordial="true"
(See referencing profile)

Figure 69 - Volume Composition from General QOS Pool

Figure 70: "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 14.1.6.2).

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.

14.1.6.2 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 70 shows the Single QOS Pool Composition (RAID Groups).

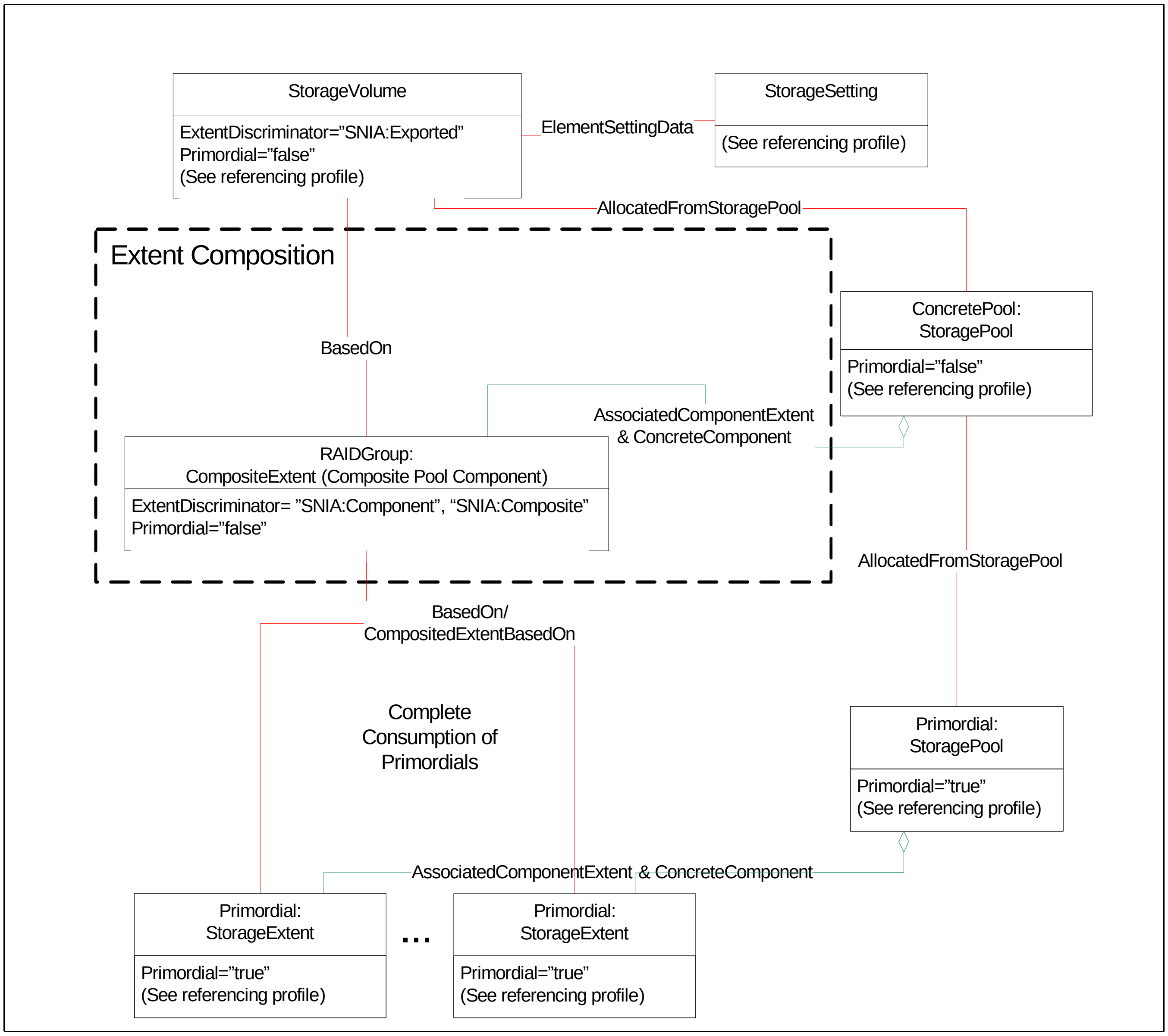

Figure 70 - Single QOS Pool Composition (RAID Groups)

Figure 71: "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.

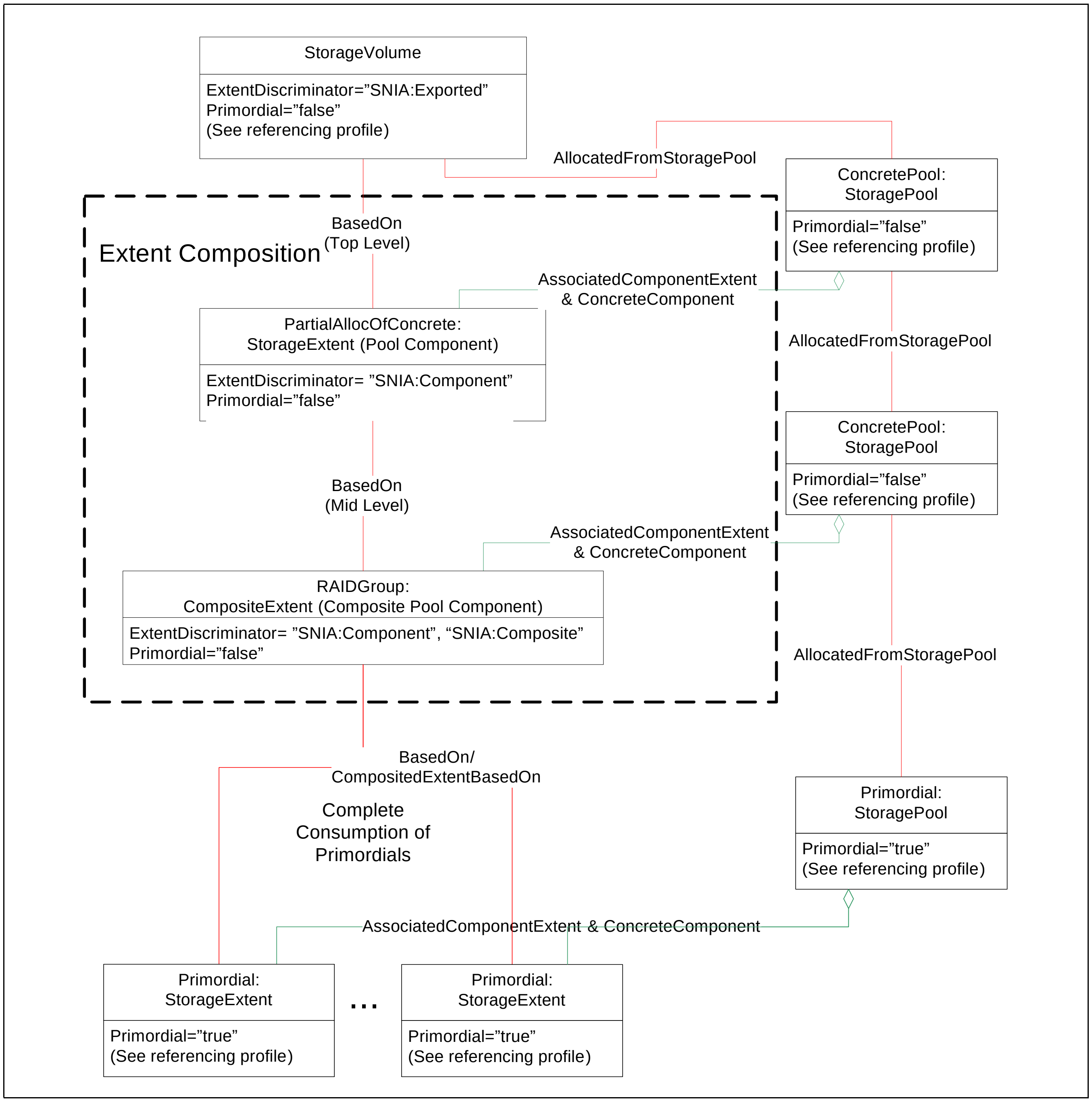
StorageVolume
ExtentDiscriminator=”SNIA:Exported”
Primordial=”false”
(See referencing profile)
AllocatedFromStoragePool
ConcretePool: StoragePool
Primordial=”false”
(See referencing profile)
Extent Composition
BasedOn (Top Level)
AssociatedComponentExtent & ConcreteComponent
PartialAllocOfConcrete: StorageExtent (Pool Component)
ExtentDiscriminator= ”SNIA:Component”
Primordial=”false”
AllocatedFromStoragePool
ConcretePool: StoragePool
Primordial=”false”
(See referencing profile)
BasedOn (Mid Level)
AssociatedComponentExtent & ConcreteComponent
RAIDGroup: CompositeExtent (Composite Pool Component)
ExtentDiscriminator= ”SNIA:Component”, “SNIA:Composite”
Primordial=”false”
AllocatedFromStoragePool
BasedOn/ CompositedExtentBasedOn
Complete Consumption of Primordials
Primordial: StoragePool
Primordial=”true”
(See referencing profile)
AssociatedComponentExtent & ConcreteComponent
Primordial: StorageExtent
Primordial=”true”
(See referencing profile)
...
Primordial: StorageExtent
Primordial=”true”
(See referencing profile)

Figure 71 - SIngle QOS Pool Composition - Two Concretes

14.1.6.3 Example RAID Compositions from Block Services

Table 332 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 332 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 332 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 332 - 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

14.1.6.3.1 JBOD (Concatenation)

Figure 72: "Concatenation Composition" shows a partial instance diagram for a JBOD Volume or Pool, in which the Antecedent Extents are concatenated.

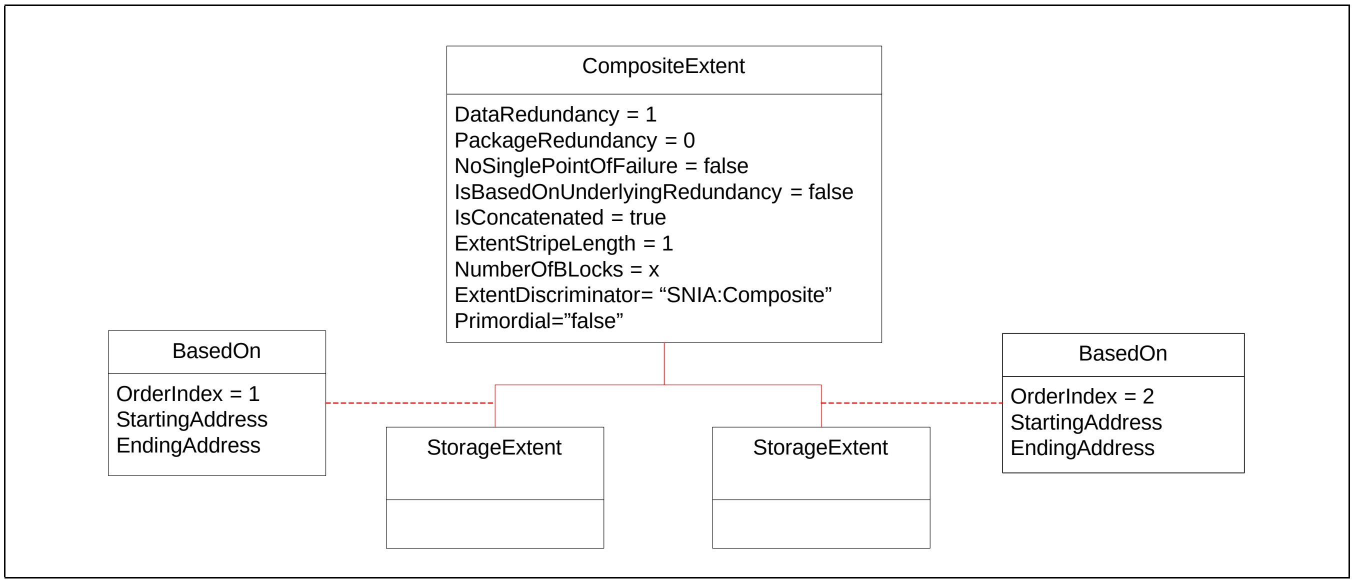

Figure 72 - Concatenation Composition

14.1.6.3.2 RAID0 (Striping)

Figure 73: "RAID0 Composition" shows a partial instance diagram for a RAID0 Volume or Pool.

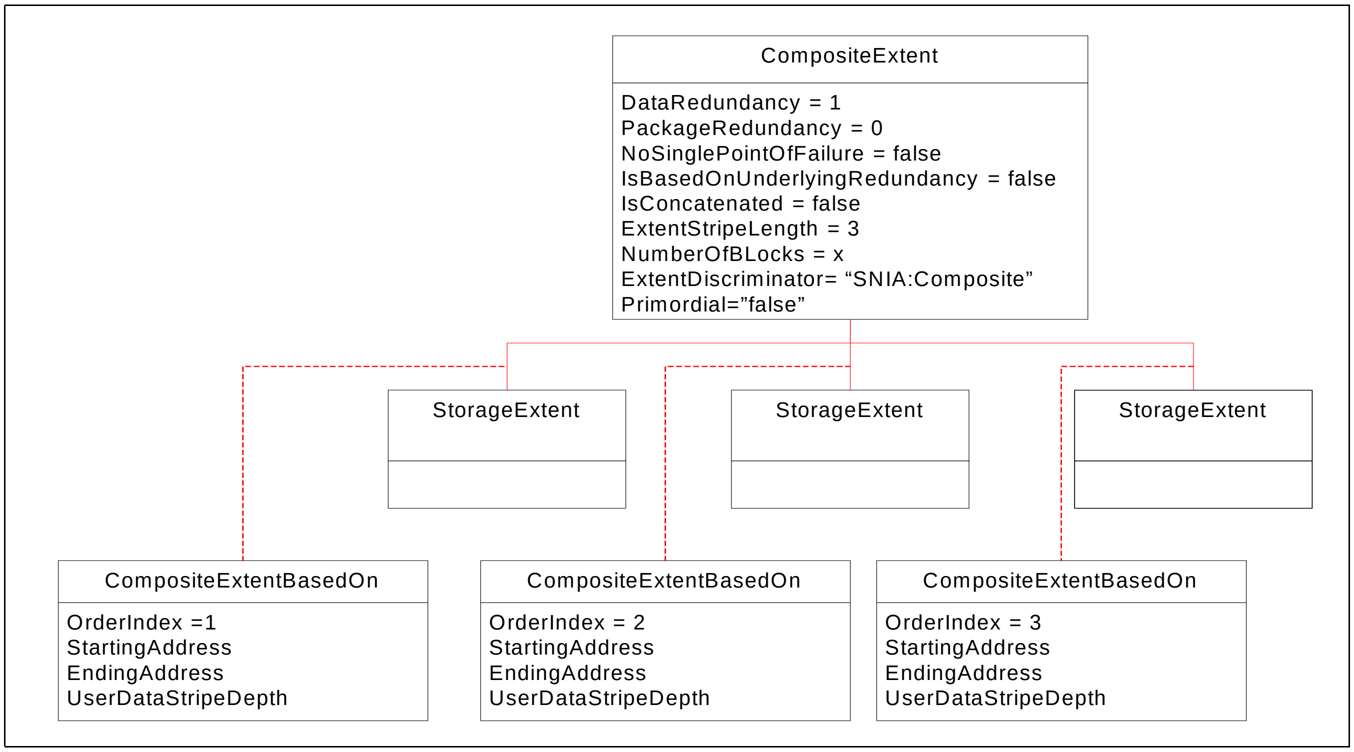

Figure 73 - RAID0 Composition

14.1.6.3.3 RAID1

Figure 74: "RAID1 Composition" shows a partial instance diagram for a RAID1 Volume or Pool.

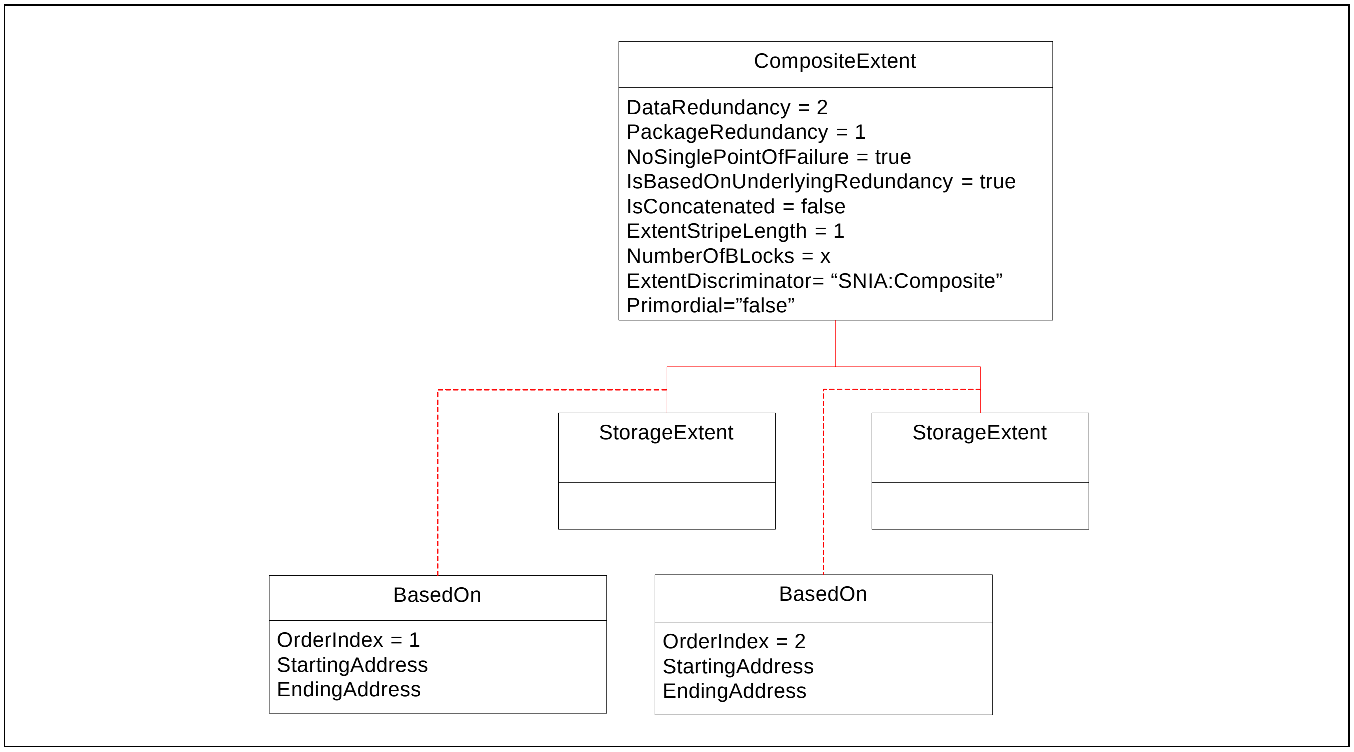

Figure 74 - RAID1 Composition

14.1.6.3.4 RAID10

Figure 75: "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.

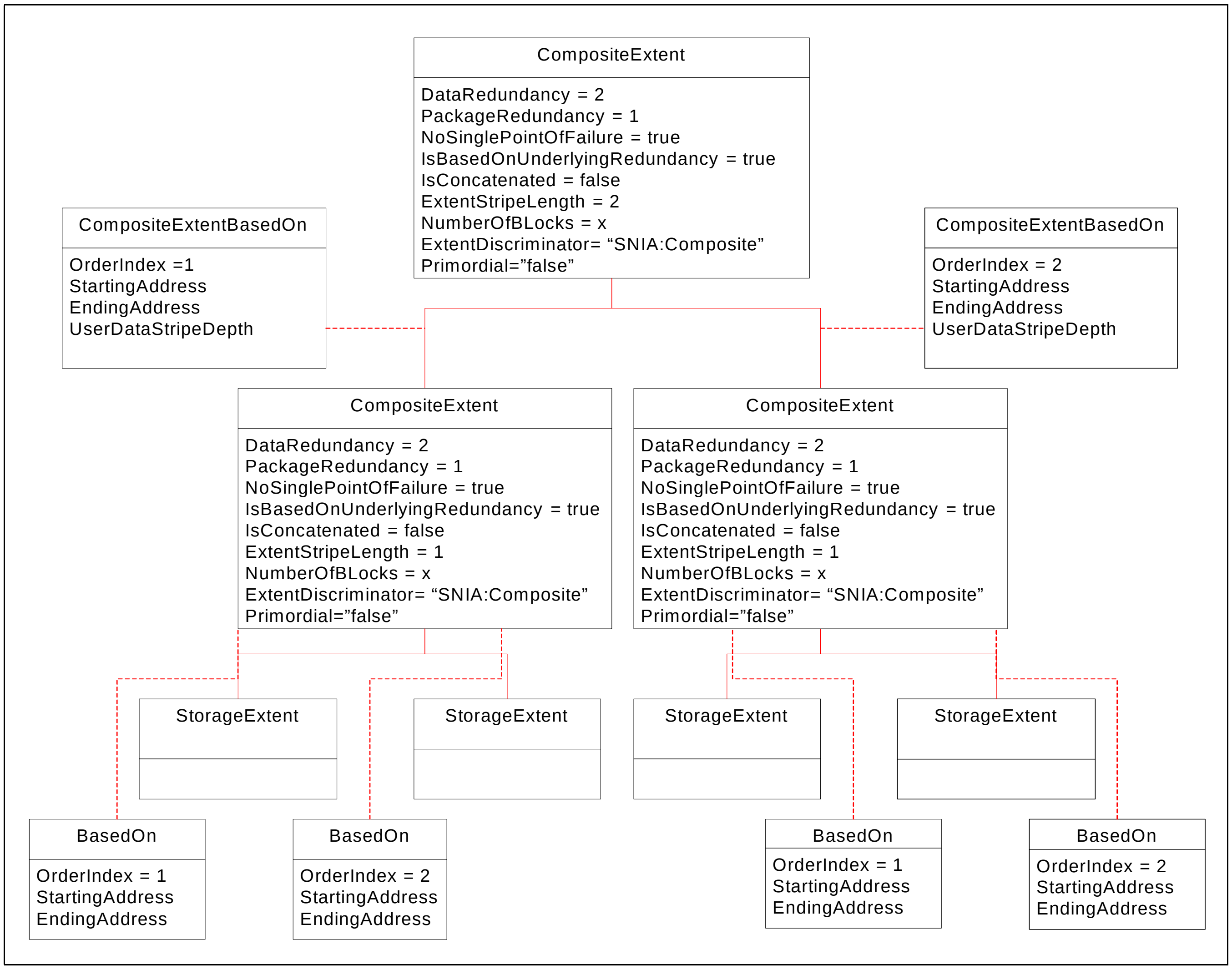

Figure 75 - RAID10 Composition

14.1.6.3.5 RAID0+1

Figure 76: "RAID0+1 Composition" shows a partial instance diagram for a RAID0+1 Volume or Pool

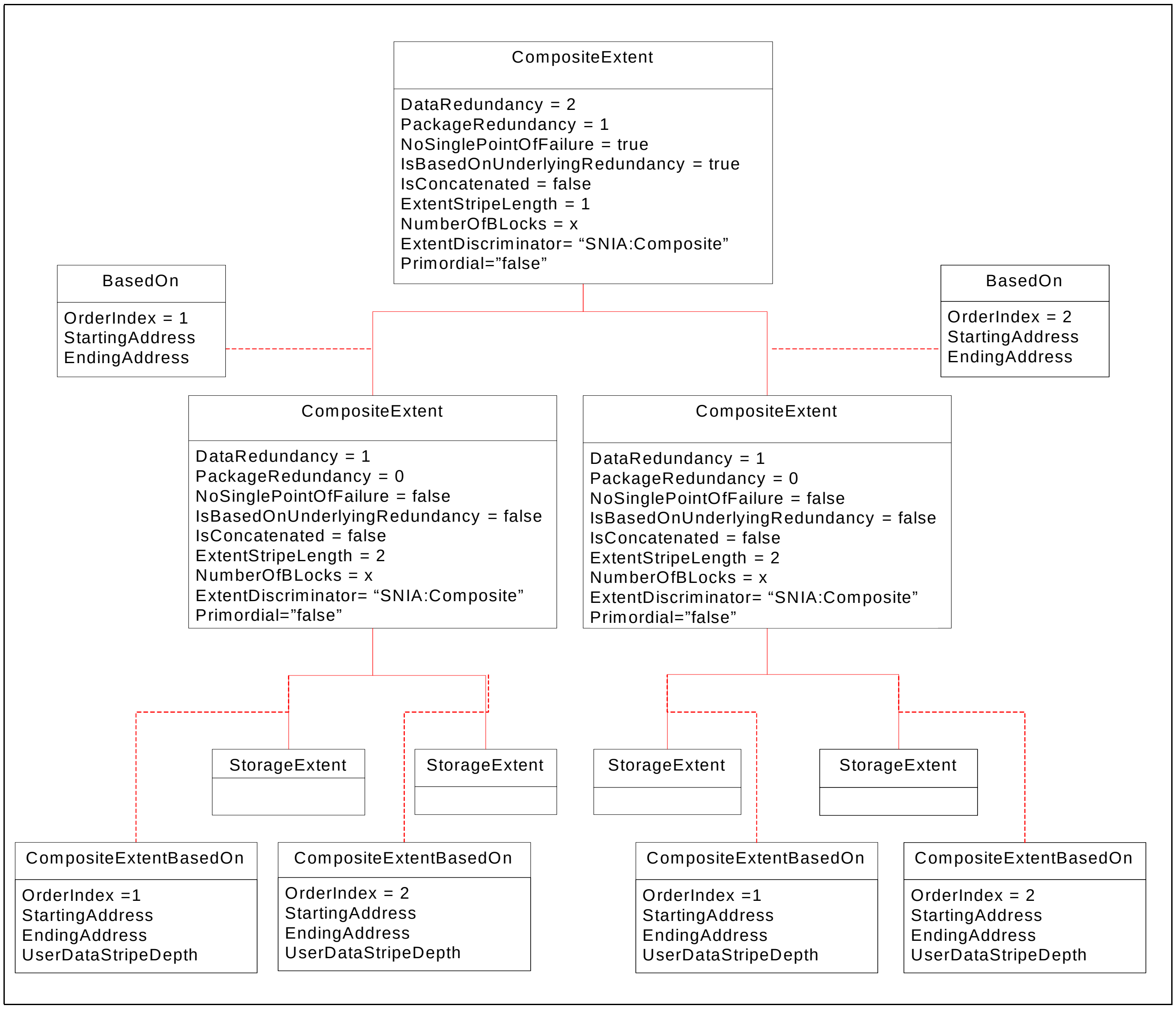

Figure 76 - RAID0+1 Composition

14.1.6.3.6 RAID4 or 5

Figure 77: "RAID4, 5 Composition" shows a partial instance diagram for a RAID4 or 5 Volume or Pool.

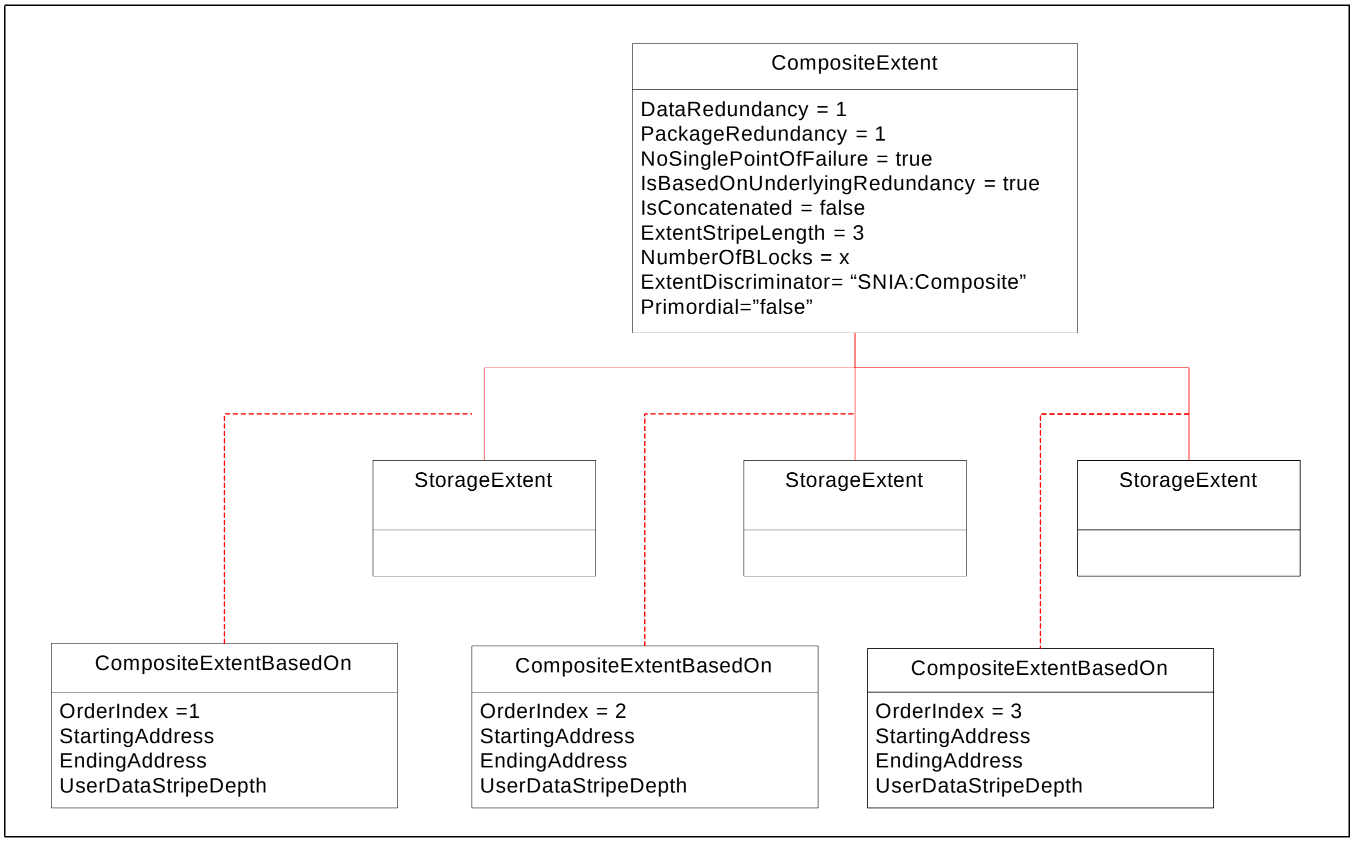

Figure 77 - RAID4, 5 Composition

14.1.6.3.7 RAID6, 5DP, and 4DP

Figure 78: "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.

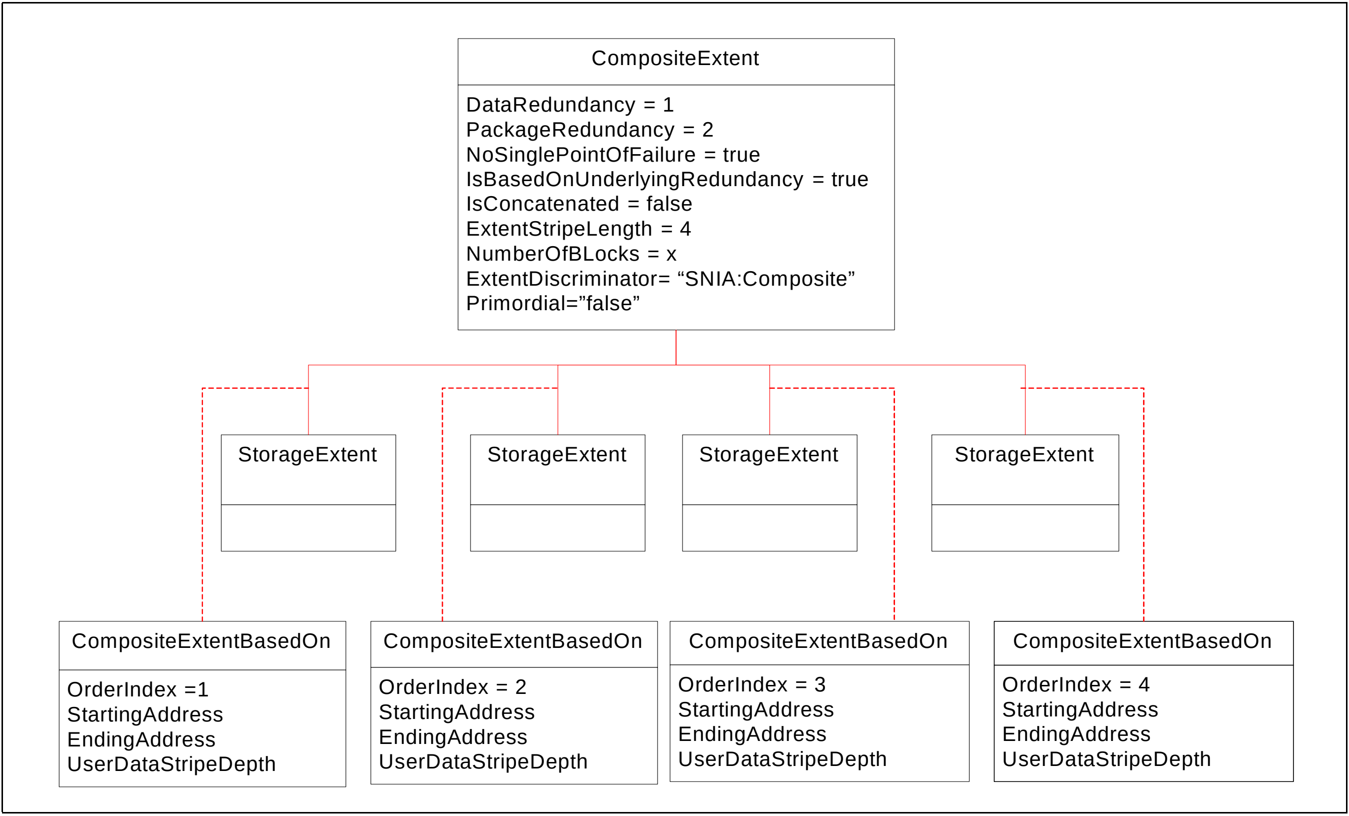

Figure 78 - RAID 6, 5DP, 4DP

14.1.6.3.8 RAID 15

Figure 79: "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.

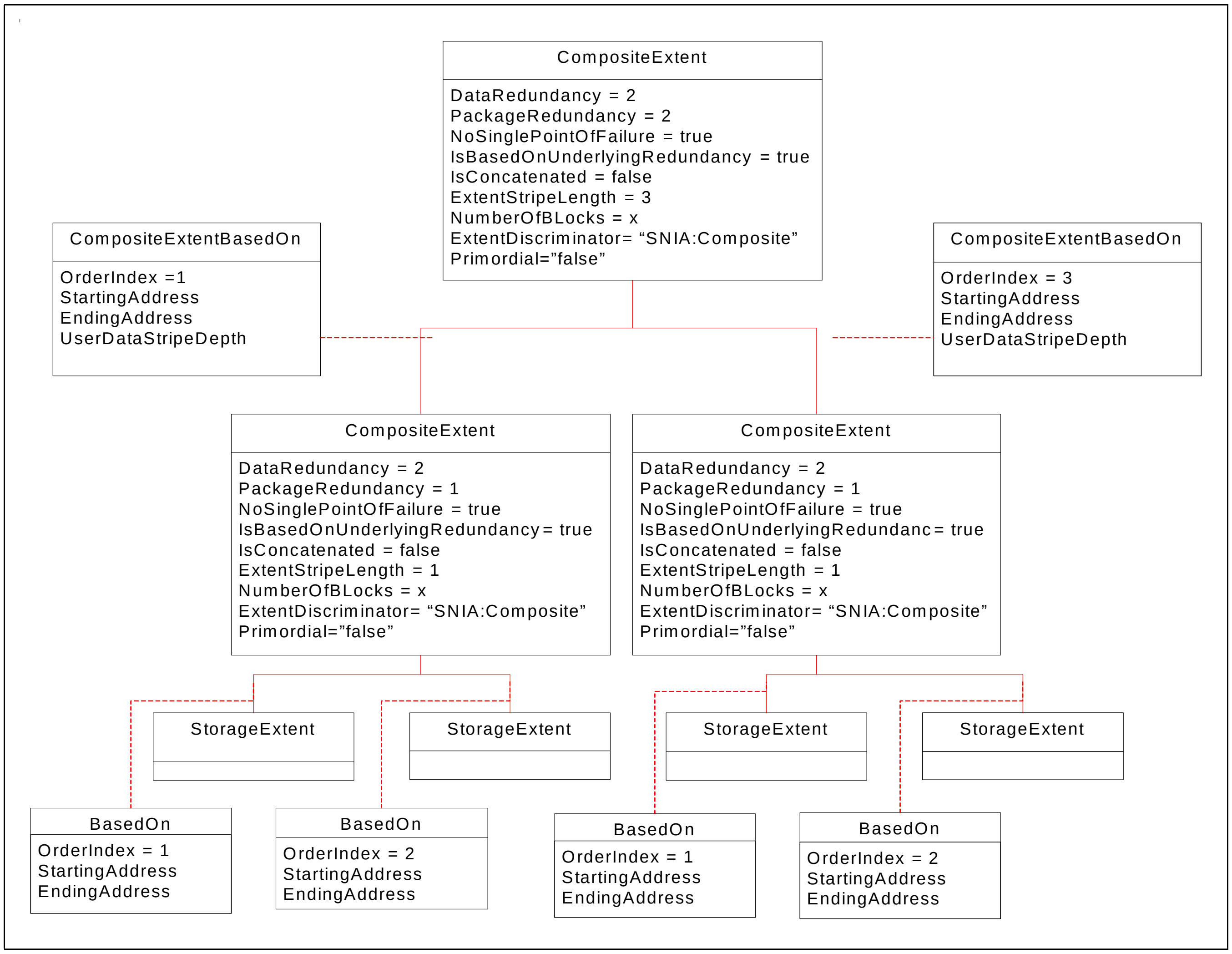

Figure 79 - RAID15 Composition

14.1.6.3.9 RAID50

Figure 80: "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.

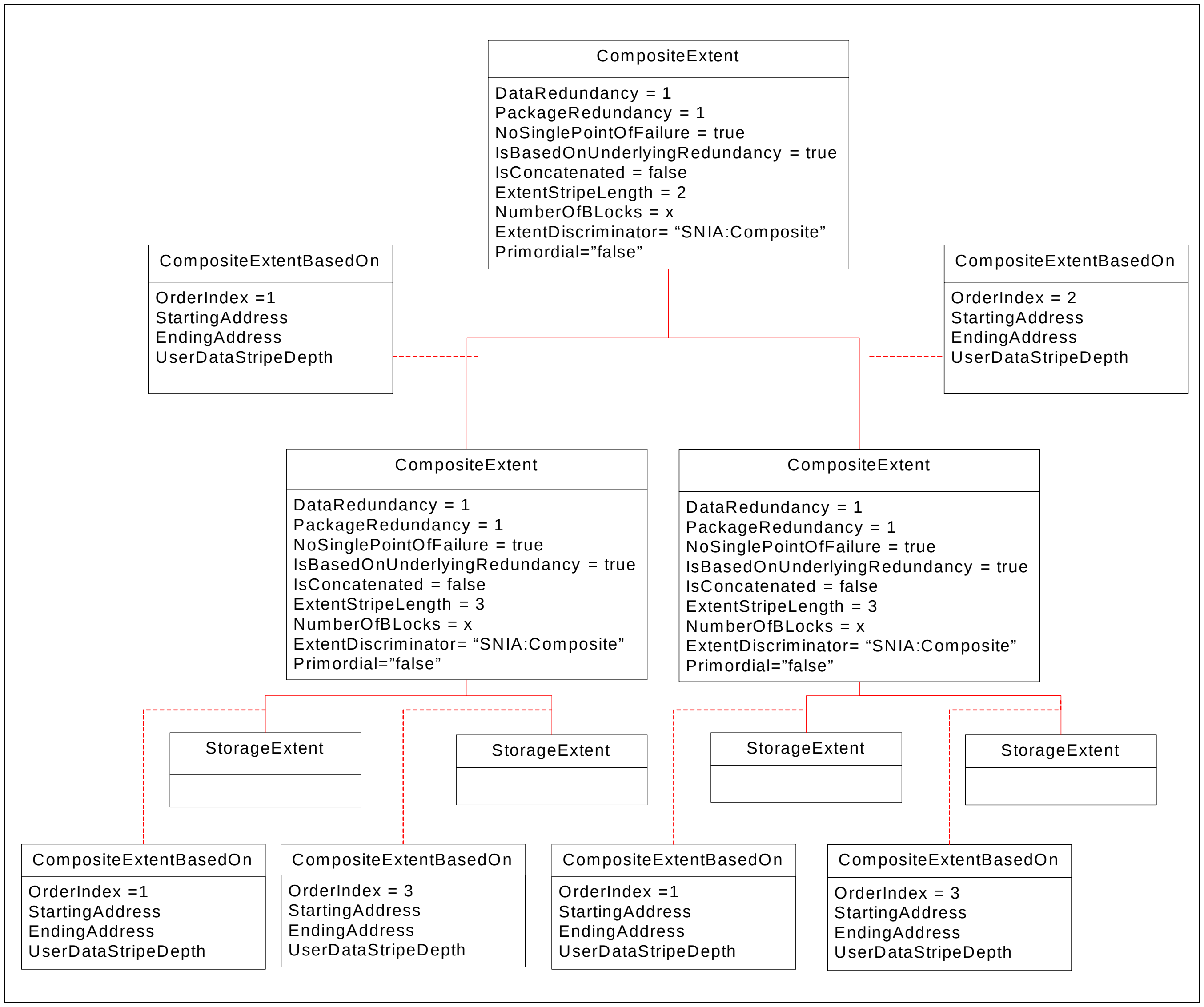

Figure 80 - RAID50 Composition

14.1.6.3.10 RAID51

Figure 81: "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.

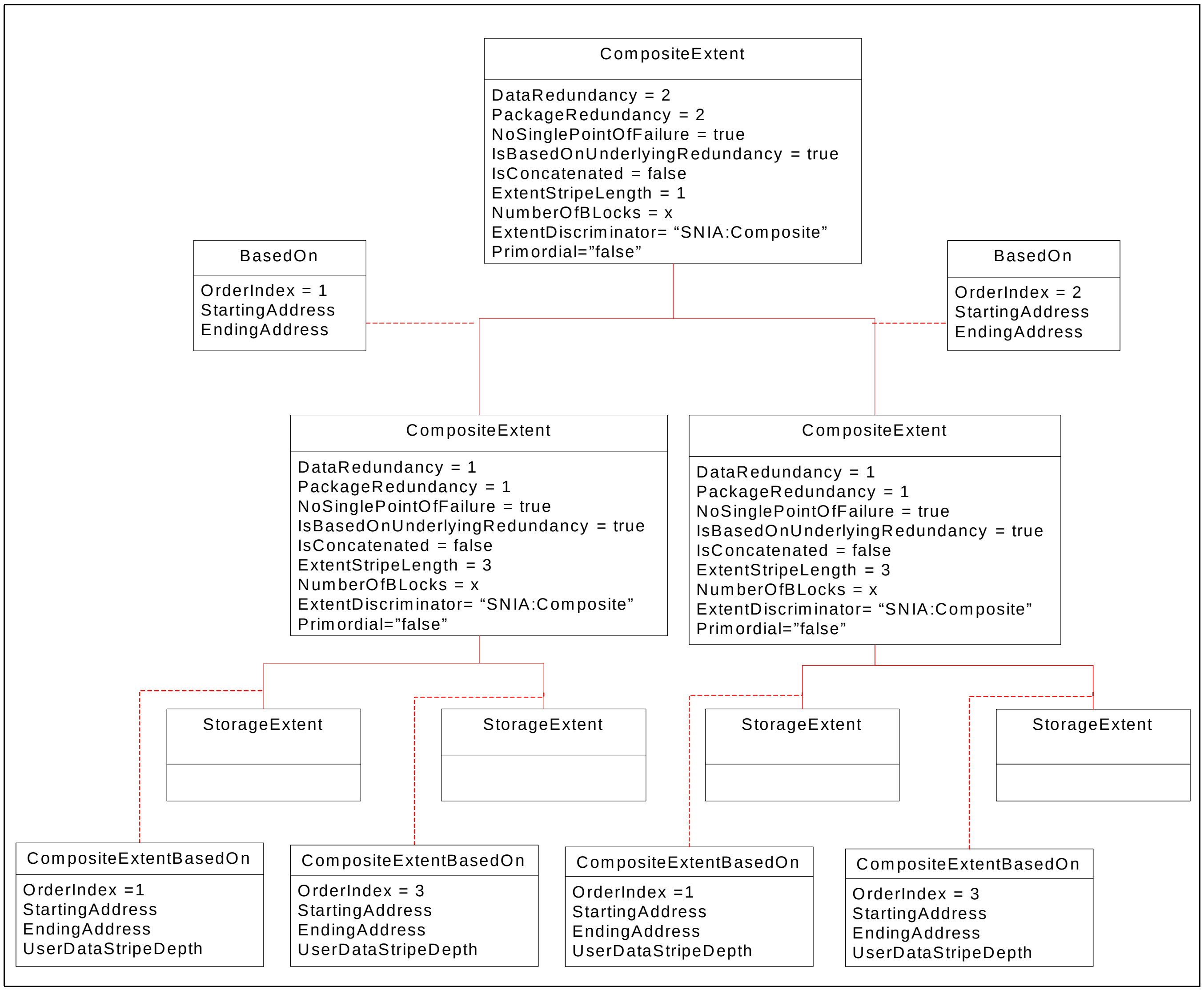

Figure 81 - RAID51 Composition

14.2 Health and Fault Management Considerations

Not defined in this standard.

14.3 Cascading Considerations

None.

14.4 Supported Subprofiles and Packages

Related Profiles for Extent Composition: Not defined in this standard.

14.5 Methods of the Profile

None.

14.6 Client Considerations and Recipes

14.6.1 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 Primoridal 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 Primoridal 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->)
```

14.6.2 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>
}
```

14.7 Registered Name and Version

Extent Composition version 1.6.0 (Component Profile)

CIM Schema Version: 2.29

14.8 CIM Elements

Table 333 describes the CIM elements for Extent Composition.

Table 333 - CIM Elements for Extent Composition

Element Name	Requirement	Description
14.8.1 CIM_AssociatedComponentExtent (Pool Component to Concrete Pool)	Mandatory	
14.8.2 CIM_AssociatedRemainingExtent (Pool to its remaining extents)	Mandatory	
14.8.3 CIM_BasedOn (Mid level BasedOn)	Optional	Associates a Storage Extent (Pool Component or Intermediate) to underlying Storage Extents it is based on.
14.8.4 CIM_BasedOn (Top level BasedOn)	Mandatory	Associates a StorageVolume (or LogicalDisk) to the underlying Storage Extent it is based on.
14.8.5 CIM_CompositeExtent (Composite Intermediate)	Optional	Represents a Concrete StorageExtent that is a composite and does not have an AssociatedComponentExtent association to a Concrete StoragePool.
14.8.6 CIM_CompositeExtent (Composite Pool Component)	Optional	Represents a Concrete StorageExtent that is a composite and has an AssociatedComponentExtent association to a Concrete StoragePool.
14.8.7 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.
14.8.8 CIM_ConcreteComponent (Pool Component to Concrete Pool)	Mandatory	Deprecated. Associate the extents that are playing the Pool Component role to their aggregating StoragePool.
14.8.9 CIM_ConcreteComponent (Remaining Extent to Pool)	Mandatory	Deprecated. Associate a remaining extent to the StoragePool for which it represents unused space.
14.8.10 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.

Table 333 - CIM Elements for Extent Composition

Element Name	Requirement	Description
14.8.11 CIM_HostedCollection (System to predefined Extent Composition IndicationFilters)	Conditional	Experimental. Conditional requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='5' (Predefined Filter Collections).
14.8.12 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).
14.8.13 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.
14.8.14 CIM_StorageExtent (Intermediate)	Optional	Represents a Concrete StorageExtent that is not a composite and does not have an AssociatedComponentExtent association to a Concrete StoragePool.
14.8.15 CIM_StorageExtent (Pool Component)	Optional	Represents a Concrete StorageExtent that is not a composite and has an AssociatedComponentExtent association to a Concrete StoragePool.
14.8.16 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.
14.8.17 CIM_SystemDevice (Composite Extent System)	Optional	Associates a CompositeExtent to a hosting computer system.
14.8.18 CIM_SystemDevice (Storage Extent System)	Optional	Associates a StorageExtent to a hosting computer system.

14.8.1 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 334 describes class CIM_AssociatedComponentExtent (Pool Component to Concrete Pool).

Table 334 - SMI Referenced Properties/Methods for CIM_AssociatedComponentExtent (Pool Component to Concrete Pool)

Properties	Flags	Requirement	Description & 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.

14.8.2 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 335 describes class CIM_AssociatedRemainingExtent (Pool to its remaining extents).

Table 335 - SMI Referenced Properties/Methods for CIM_AssociatedRemainingExtent (Pool to its remaining extents)

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	The (non-empty, Concrete or Primordial) StoragePool.
PartComponent		Mandatory	The storage extent that represents free space in the concrete storage pool.

14.8.3 CIM_BasedOn (Mid level BasedOn)

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

Table 336 describes class CIM_BasedOn (Mid level BasedOn).

Table 336 - SMI Referenced Properties/Methods for CIM_BasedOn (Mid level BasedOn)

Properties	Flags	Requirement	Description & 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.

14.8.4 CIM_BasedOn (Top level BasedOn)

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

Table 337 describes class CIM_BasedOn (Top level BasedOn).

Table 337 - SMI Referenced Properties/Methods for CIM_BasedOn (Top level BasedOn)

Properties	Flags	Requirement	Description & Notes
StartingAddress		Optional	
EndingAddress		Optional	
Dependent		Mandatory	The Storage Volume or Logical Disk that dependents on the associated extent.
Antecedent		Mandatory	The extent on which the storage volume or logical disk is based.

14.8.5 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 338 describes class CIM_CompositeExtent (Composite Intermediate).

Table 338 - SMI Referenced Properties/Methods for CIM_CompositeExtent (Composite Intermediate)

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

Table 338 - SMI Referenced Properties/Methods for CIM_CompositeExtent (Composite Intermediate)

Properties	Flags	Requirement	Description & Notes
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'.

14.8.6 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 339 describes class CIM_CompositeExtent (Composite Pool Component).

Table 339 - SMI Referenced Properties/Methods for CIM_CompositeExtent (Composite Pool Component)

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

Table 339 - SMI Referenced Properties/Methods for CIM_CompositeExtent (Composite Pool Component)

Properties	Flags	Requirement	Description & Notes
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'.

14.8.7 CIM_CompositeExtentBasedOn

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

Table 340 describes class CIM_CompositeExtentBasedOn.

Table 340 - SMI Referenced Properties/Methods for CIM_CompositeExtentBasedOn

Properties	Flags	Requirement	Description & 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.

Table 340 - SMI Referenced Properties/Methods for CIM_CompositeExtentBasedOn

Properties	Flags	Requirement	Description & Notes
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.

14.8.8 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 341 describes class CIM_ConcreteComponent (Pool Component to Concrete Pool).

Table 341 - SMI Referenced Properties/Methods for CIM_ConcreteComponent (Pool Component to Concrete Pool)

Properties	Flags	Requirement	Description & 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.

14.8.9 CIM_ConcreteComponent (Remaining Extent to Pool)

Deprecated.

Created By: External

Modified By: External

Deleted By: External

Requirement: Mandatory

Table 342 describes class CIM_ConcreteComponent (Remaining Extent to Pool).

Table 342 - SMI Referenced Properties/Methods for CIM_ConcreteComponent (Remaining Extent to Pool)

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	The (non-empty) StoragePool.
PartComponent		Mandatory	The storage extent that represents unused space in the storage pool.

14.8.10 CIM_FilterCollection (Extent Composition Predefined FilterCollection)

Experimental. This is a collection of predefined IndicationFilters to which a client may subscribe. A 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 343 describes class CIM_FilterCollection (Extent Composition Predefined FilterCollection).

Table 343 - SMI Referenced Properties/Methods for CIM_FilterCollection (Extent Composition Predefined FilterCollection)

Properties	Flags	Requirement	Description & 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:Predefined'.

14.8.11 CIM_HostedCollection (System to predefined Extent Composition IndicationFilters)

Experimental.

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

Table 344 describes class CIM_HostedCollection (System to predefined Extent Composition IndicationFilters).

Table 344 - SMI Referenced Properties/Methods for CIM_HostedCollection (System to predefined Extent Composition IndicationFilters)

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	Reference to the predefined FilterCollection for Extent Composition.
Antecedent		Mandatory	Reference to the 'Top level' System.

14.8.12 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 345 describes class CIM_MemberOfCollection (Extent Composition Filter Collection to FilterCollection).

Table 345 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Extent Composition Filter Collection to FilterCollection)

Properties	Flags	Requirement	Description & Notes
Collection		Mandatory	Reference to the Extent Composition predefined FilterCollection.
Member		Mandatory	Reference to the Extent Composition predefined FilterCollection.

14.8.13 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 346 describes class CIM_MemberOfCollection (Predefined Filter Collection to Extent Composition Filters).

Table 346 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Predefined Filter Collection to Extent Composition Filters)

Properties	Flags	Requirement	Description & Notes
Collection		Mandatory	Reference to the Extent Composition predefined FilterCollection.
Member		Mandatory	Reference to the predefined IndicationFilters of the Extent Composition implementation.

14.8.14 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 347 describes class CIM_StorageExtent (Intermediate).

Table 347 - SMI Referenced Properties/Methods for CIM_StorageExtent (Intermediate)

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

Table 347 - SMI Referenced Properties/Methods for CIM_StorageExtent (Intermediate)

Properties	Flags	Requirement	Description & Notes
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'.

14.8.15 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 348 describes class CIM_StorageExtent (Pool Component).

Table 348 - SMI Referenced Properties/Methods for CIM_StorageExtent (Pool Component)

Properties	Flags	Requirement	Description & 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:Pool Component'.

14.8.16 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 349 describes class CIM_StorageExtent (Remaining).

Table 349 - SMI Referenced Properties/Methods for CIM_StorageExtent (Remaining)

Properties	Flags	Requirement	Description & 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:Remaining'.

14.8.17 CIM_SystemDevice (Composite Extent System)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 350 describes class CIM_SystemDevice (Composite Extent System).

Table 350 - SMI Referenced Properties/Methods for CIM_SystemDevice (Composite Extent System)

Properties	Flags	Requirement	Description & 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.

14.8.18 CIM_SystemDevice (Storage Extent System)

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

Table 351 describes class CIM_SystemDevice (Storage Extent System).

Table 351 - SMI Referenced Properties/Methods for CIM_SystemDevice (Storage Extent System)

Properties	Flags	Requirement	Description & 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

Clause 15: LUN Creation Subprofile

The functionality of the LUN Creation and Pool Manipulation Capabilities, and Settings Subprofiles have been subsumed by the Clause 5: Block Services Package.

The LUN Creation Subprofile is defined in section 7.3.3.11 of SMI-S 1.0.2.

DEPRECATED

DEPRECATED

Clause 16: Extent Mapping Subprofile

The functionality of the Extent Mapping Subprofile (Section 7.3.3.5 of SMI-S 1.0.2) has been subsumed by the Extent Composition Subprofile (8.3.1.15).

DEPRECATED

DEPRECATED

Clause 17: LUN Mapping and Masking Subprofile

The LUN Mapping and Masking Subprofile (section 7.3.3.14 in SMI-S 1.0.2) has been replaced by Clause 18: Masking and Mapping Subprofile.

17.1 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 it's 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 17: LUN Mapping and Masking Subprofile in *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4*.)
- 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

Clause 18: Masking and Mapping Subprofile

18.1 Description

Note: See 17.1 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 disks 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.

18.1.1 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 each logical unit, initiator port, and target port - the concept of path is independent from a CIM model, but a view expresses a combinations 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.

18.1.2 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). SCSIProtocolEnpoint 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 82: "Generic System with no Configuration Service" and Figure 83: "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 82: "Generic System with no Configuration Service" depicts a configuration with no LUN Masking capabilities.

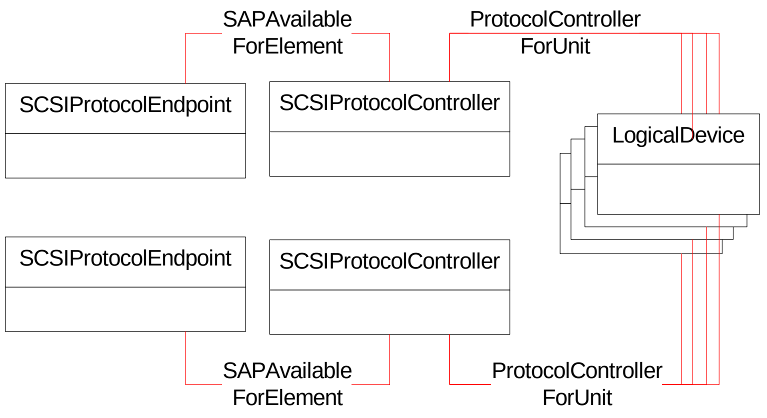

Figure 82 - Generic System with no Configuration Service

Figure 83: "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.

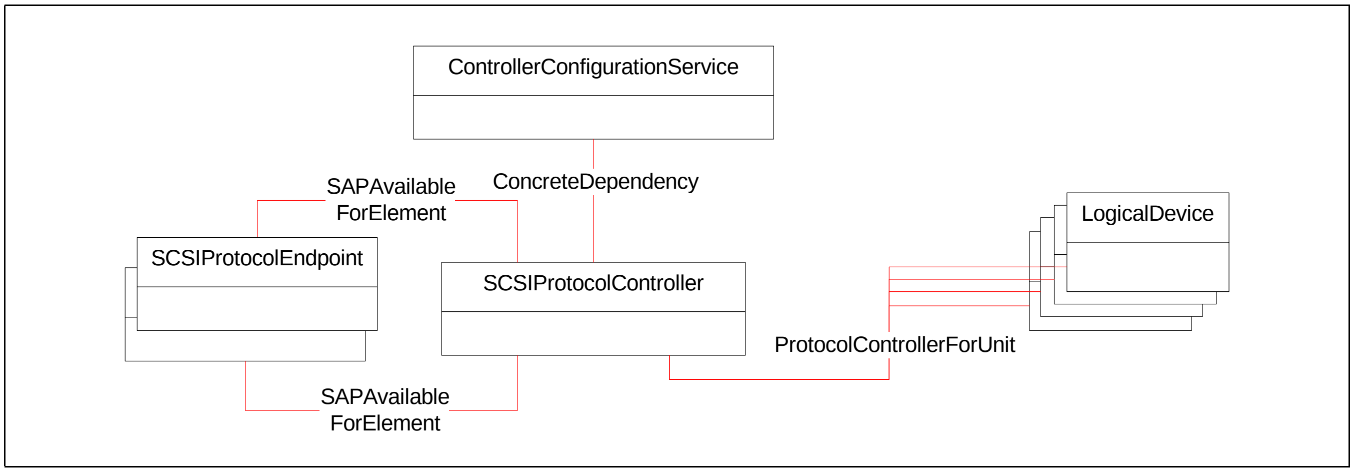

Figure 83 - Generic System with ControllerConfigurationService

The means to grant access is discussed in 18.5.1 "ExposePaths".

18.1.3 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

For each StorageHardwareID relationship to a SCSIProtocolController there shall also be an instance of the AssociatedPrivilege association. The AssociatedPrivilege association is in addition to the instances of AuthorizedPrivilege, AuthorizedSubject and AuthorizedTarget. AuthorizedPrivilege, AuthorizedSubject and AuthorizedTarget are deprecated and will be removed in a future version of the specification. To maintain backward compatibility with the previous versions of SMI-S, the implementation shall continue to provide instances of these classes.

See Figure 84, “Relationship of Initiator IDs, Endpoints, and Logical Units” for the relationship between these classes.

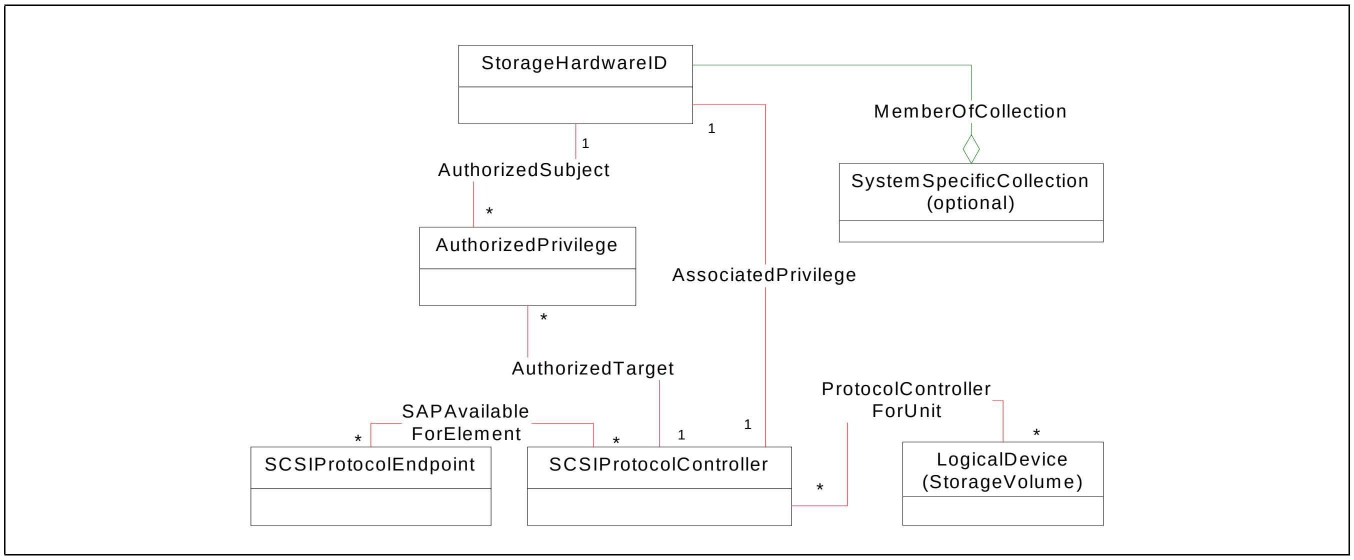

Figure 84 - Relationship of Initiator IDs, Endpoints, and Logical Units

18.1.4 Initiator ID Collections

An implementation may optionally model collections of Initiator IDs. This is modeled as depicted in Figure 84: "Relationship of Initiator IDs, Endpoints, and Logical Units". If the implementation supports collection of initiator IDs, the instrumentation shall set ProtocolControllerMaskingCapabilities.ProtocolControllerSupportsCollections to True

18.1.5 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 or 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.

For the one null-name StorageHardwareID that is related to each default SCSIProtocolController, there shall be one instance of the AssociatedPrivilege association.

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.

18.1.6 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.

18.1.7 Read-only verses Read-Write access

ExposePaths includes a DeviceAccess parameter that is used to set the DeviceAccess property of ProtocolControllerForUnit association.

18.1.8 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).

18.1.9 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.

18.1.10 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.

18.1.11 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 SCSI's). 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.

18.1.12 SCSIProtocolController Properties

Table 352 - 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 depends 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	In 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 InitatorPortIDs, 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 352. 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.

18.1.13 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 previous configured with the setting behavior.

A client can associate StorageHardwareIDs to a StorageClientSettingData instance (when a customer or host agent maps an initiator to an OS type). This is done by specifying the Setting parameter to CreateStorageHardwareID). A client can also associate an StorageClientSettingData instance to a storage 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 85: "StorageClientSettingData Model" provides an example.

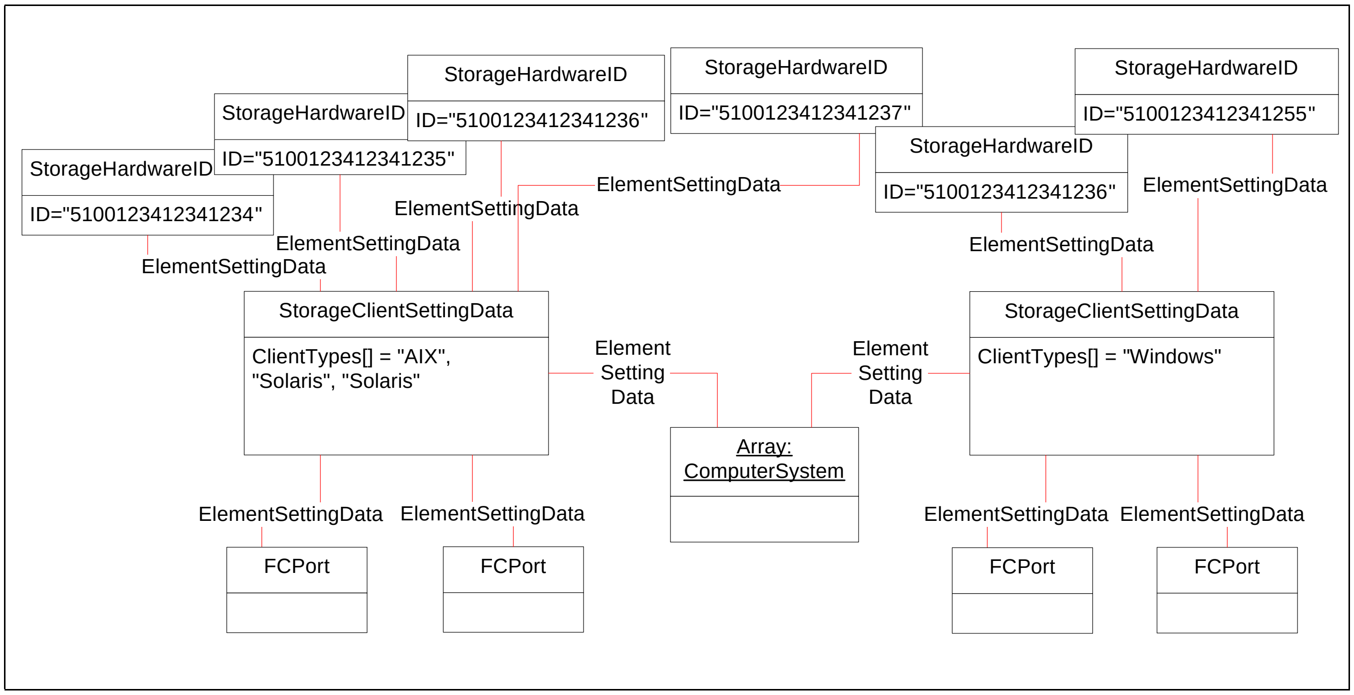

Figure 85 - StorageClientSettingData Model

Figure 86: "Entire Model" depicts the entire model.

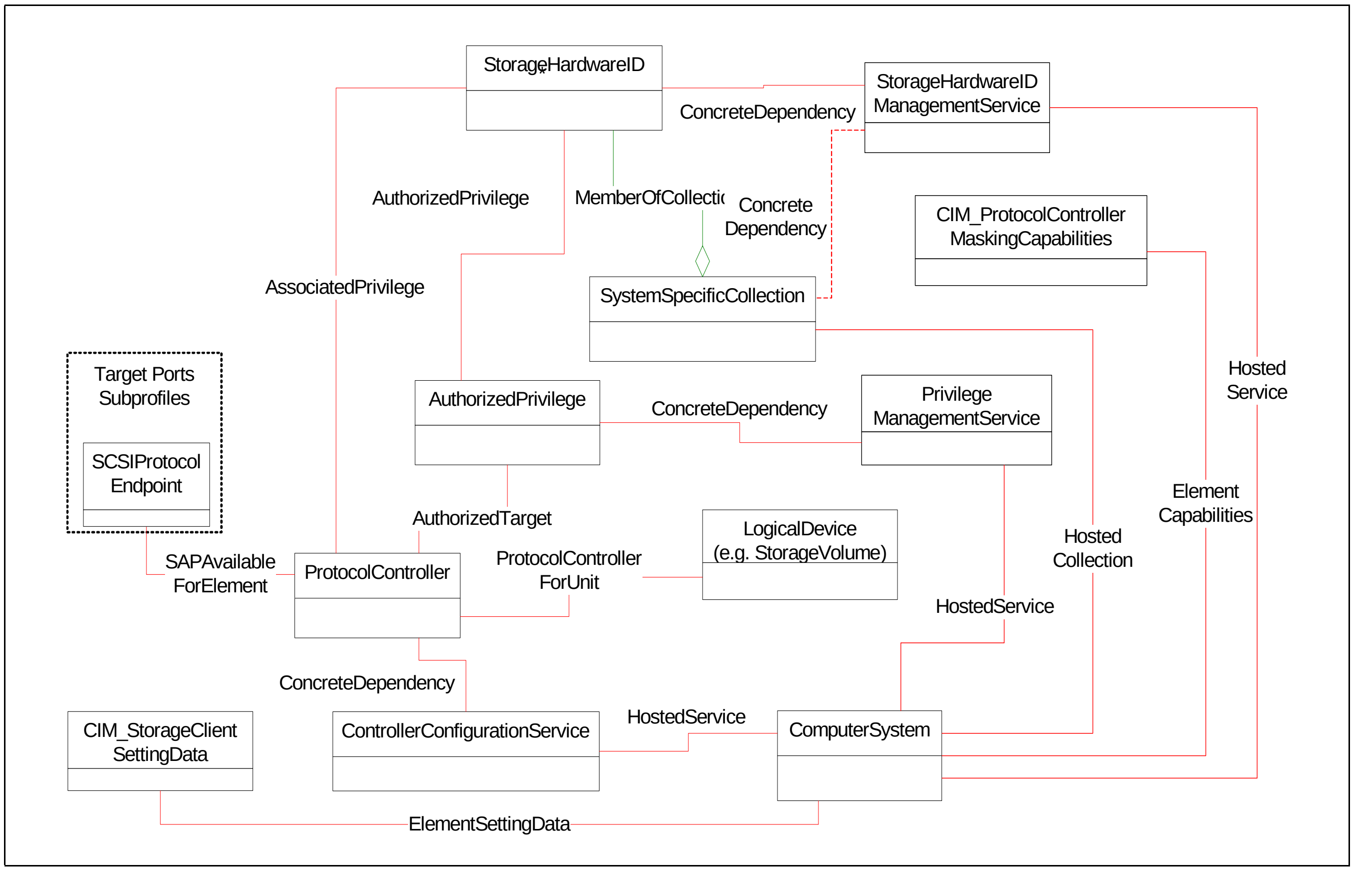

Figure 86 - Entire Model

18.1.14 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.

18.1.15 Instrumentation Requirements

If a PrivilegeManagementService is not present, then all access is provided through the ControllerConfigurationService. 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.

18.1.16 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 18.8.19 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 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 353.

Table 353 - 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 354.

Table 354 - 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

18.2 Health and Fault Management Considerations

None.

18.3 Cascading Considerations

None.

18.4 Supported Subprofiles, and Packages

None.

18.5 Methods of the Profile

18.5.1 ExposePaths

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 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 is only possible is 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 355.

Table 355 - ExposePath Use Cases

parameters/use cases	LUNames	InitiatorPortIDs	TargetPortIDs	DeviceNumbers	DeviceAccesses	ProtocolControllers (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.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 (18.8.27 "CIM_ProtocolControllerForUnit").

18.5.1.1 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 355, “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 355, “ExposePath Use Cases” for situations where this parameter may be null.

IN string TargetPortIDs[]

IDs of target ports. See Table 355, “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.

18.5.2 HidePaths

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 356.

Table 356 - 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.

18.5.2.1 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 356, “HidePaths Use Cases” for situations where this parameter may be null.

IN string InitiatorPortIDs[]

IDs of initiator ports. See Table 356, “HidePaths Use Cases” for situations where this parameter may be null.

IN string TargetPortIDs[]

IDs of target ports. See Table 356, “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.

18.5.3 ExposeDefaultLUs

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 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 a 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 357.

Table 357 - 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.

18.5.3.1 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 357, “Use Cases for ExposeDefaultLUs” for situations where this parameter may be null.

IN string TargetPortIDs[]

> IDs of target ports. See Table 357, “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.

18.5.4 HideDefaultLUs

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 358

Table 358 - 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 ProtocolControllerMaskingCapabilites 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.

18.5.4.1 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 358, “Use Cases for HideDefaultLUs” for situations where this parameter may be null.

IN string TargetPortIDs[]

IDs of target ports. See Table 358, “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.

18.5.5 CreateStorageHardwareID

CreateStorageHardwareID creates a StorageHardwareID and the ConcreteDependency association between this service and the new StorageHardwareID.

18.5.5.1 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.

18.5.6 DeleteStorageHardwareID

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.

18.5.6.1 Uint32 DeleteStorageHardwareID

IN CIM_StorageHardwareID REF HardwareID

REF to the StorageHardwareID to delete

18.5.7 CreateHardwareIDCollection

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.

18.5.7.1 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.

18.5.8 AddHardwareIDsToCollection

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.

18.5.8.1 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

18.5.9 DeleteProtocolController

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.

18.5.9.1 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

18.6 Client Considerations and Recipes

18.6.1 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[“LUNames”]          = 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>
      }
    }
```

18.6.2 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)
```

18.6.3 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)
```

18.7 Registered Name and Version

Masking and Mapping version 1.6.0 (Component Profile)

18.8 CIM Elements

Table 359 describes the CIM elements for Masking and Mapping.

Table 359 - CIM Elements for Masking and Mapping

Element Name	Requirement	Description
18.8.1 CIM_AssociatedPrivilege	Mandatory	
18.8.2 CIM_AuthorizedPrivilege	Mandatory	Deprecated.
18.8.3 CIM_AuthorizedSubject	Mandatory	Deprecated.
18.8.4 CIM_AuthorizedTarget	Mandatory	Deprecated.
18.8.5 CIM_ConcreteDependency (Associates ControllerConfiguirationService and ProtocolController)	Mandatory	
18.8.6 CIM_ConcreteDependency (Associates PrivilegeManagementService and AuthorizedPrivilege)	Mandatory	Deprecated.
18.8.7 CIM_ConcreteDependency (Associates StorageHardwareIDManagementService and StorageHardwareID)	Mandatory	
18.8.8 CIM_ConcreteDependency (Associates StorageHardwareIDManagementService and SystemSpecificCollection)	Conditional	Conditional requirement: Implementation support for collections of StorageHardwareIDs.
18.8.9 CIM_ControllerConfigurationService	Mandatory	
18.8.10 CIM_ElementCapabilities (EnabledLogicalElementCapabilities to ControllerConfigurationService)	Optional	Associates EnabledLogicalElementCapabilities with ControllerConfigurationService.
18.8.11 CIM_ElementCapabilities (EnabledLogicalElementCapabilities to ProtocolController)	Optional	Expressed the ability for the element to be named or have its state changed.
18.8.12 CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StorageHardwareID)	Optional	Associates EnabledLogicalElementCapabilities to StorageHardwareID.

Table 359 - CIM Elements for Masking and Mapping

Element Name	Requirement	Description
18.8.13 CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StorageHardwareIDManagementService)	Optional	Associates EnabledLogicalElementCapabilities with StorageHardwareIDManagementService.
18.8.14 CIM_ElementCapabilities (EnabledLogicalElementCapabilities to SystemSpecificCollection)	Conditional	Conditional requirement: Implementation support for collections of StorageHardwareIDs. Associates EnabledLogicalElementCapabilities and SystemSpecificCollection.
18.8.15 CIM_ElementCapabilities (System to ProtocolControllerMaskingCapabilities)	Mandatory	
18.8.16 CIM_ElementSettingData (Associates ComputerSystem and StorageClientSettingData)	Mandatory	
18.8.17 CIM_ElementSettingData (Associates Port and StorageClientSettingData)	Optional	
18.8.18 CIM_ElementSettingData (Associates ProtocolController and StorageClientSettingData)	Optional	
18.8.19 CIM_ElementSettingData (Associates StorageHardwareID and StorageClientSettingData)	Optional	
18.8.20 CIM_EnabledLogicalElementCapabilities	Optional	This class is used to express the naming and possible requested state change possibilities for storage elements.
18.8.21 CIM_HostedCollection	Conditional	Conditional requirement: Implementation support for collections of StorageHardwareIDs.
18.8.22 CIM_HostedService (Associates ComputerSystem and ControllerConfigurationService)	Mandatory	
18.8.23 CIM_HostedService (Associates ComputerSystem and PrivilegeManagementService)	Mandatory	Deprecated.
18.8.24 CIM_HostedService (Associates ComputerSystem and StorageHardwareIDManagementService)	Mandatory	
18.8.25 CIM_MemberOfCollection	Conditional	Conditional requirement: Implementation support for collections of StorageHardwareIDs.
18.8.26 CIM_PrivilegeManagementService	Mandatory	Deprecated.
18.8.27 CIM_ProtocolController	Mandatory	

Table 359 - CIM Elements for Masking and Mapping

Element Name	Requirement	Description
18.8.28 CIM_ProtocolControllerForUnit	Mandatory	
18.8.29 CIM_ProtocolControllerMaskingCapabilities	Mandatory	
18.8.30 CIM_SAPAvailableForElement	Mandatory	
18.8.31 CIM_StorageClientSettingData	Mandatory	
18.8.32 CIM_StorageHardwareID	Mandatory	
18.8.33 CIM_StorageHardwareIDManagementService	Mandatory	
18.8.34 CIM_SystemSpecificCollection	Conditional	Conditional requirement: Implementation support for collections of StorageHardwareIDs.
18.8.35 SNIA_ProtocolControllerMaskingCapabilities	Optional	An experimental subclass of CIM_ProtocolControllerMaskingCapabilities.
18.8.36 SNIA_StorageHardwareID	Optional	Experimental SNIA class adding SAS Address IDs.
18.8.37 SNIA_StorageHardwareIDManagementService	Optional	Experimental subclass with support for SAS StorageHardwareIDs.
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_AssociatedPrivilege	Mandatory	Creation of an AssociatedPrivilege association.
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_InstDeletion WHERE SourceInstance ISA CIM_AssociatedPrivilege	Mandatory	Deletion of an AssociatedPrivilege association.
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_AuthorizedSubject	Mandatory	Deprecated. Creation of an AuthorizedSubject association.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_AuthorizedSubject	Mandatory	Deprecated. Deletion of an AuthorizedSubject association.

18.8.1 CIM_AssociatedPrivilege

Created By: Extrinsics: CIM_ControllerConfigurationService.ExposePaths, CIM_ControllerConfigurationService.HidePaths

Modified By: Extrinsics: CIM_ControllerConfigurationService.ExposePaths, CIM_ControllerConfigurationService.HidePaths

Deleted By: Extrinsics: CIM_ControllerConfigurationService.ExposePaths, CIM_ControllerConfigurationService.HidePaths

Requirement: Mandatory

Table 360 describes class CIM_AssociatedPrivilege.

Table 360 - SMI Referenced Properties/Methods for CIM_AssociatedPrivilege

Properties	Flags	Requirement	Description & Notes
UseKey		Mandatory	Opaque and unique identifer.
PrivilegeGranted		Mandatory	Indicates if the privilege is granted or not.
Activities		Mandatory	For SMI-S, shall be 5,6 ('Read' and Write').
Subject		Mandatory	The StorageHardwareID having access to a target element.
Target		Mandatory	The ProtocolController accessible to subject element.

18.8.2 CIM_AuthorizedPrivilege

Deprecated.

Created By: Extrinsics: CIM_ControllerConfigurationService.ExposePaths, CIM_ControllerConfigurationService.HidePaths

Modified By: Extrinsics: CIM_ControllerConfigurationService.ExposePaths, CIM_ControllerConfigurationService.HidePaths

Deleted By: Extrinsics: CIM_ControllerConfigurationService.ExposePaths, CIM_ControllerConfigurationService.HidePaths

Requirement: Mandatory

Table 361 describes class CIM_AuthorizedPrivilege.

Table 361 - SMI Referenced Properties/Methods for CIM_AuthorizedPrivilege

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	Opaque and unique identifer.
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').

18.8.3 CIM_AuthorizedSubject

Deprecated.

Created By: Extrinsics: CIM_ControllerConfigurationService.ExposePaths, CIM_ControllerConfigurationService.HidePaths

Modified By: Extrinsics: CIM_ControllerConfigurationService.ExposePaths, CIM_ControllerConfigurationService.HidePaths

Deleted By: Extrinsics: CIM_ControllerConfigurationService.ExposePaths, CIM_ControllerConfigurationService.HidePaths

Requirement: Mandatory

Table 362 describes class CIM_AuthorizedSubject.

Table 362 - SMI Referenced Properties/Methods for CIM_AuthorizedSubject

Properties	Flags	Requirement	Description & 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.

18.8.4 CIM_AuthorizedTarget

Deprecated.

Created By: Extrinsics: CIM_ControllerConfigurationService.ExposePaths, CIM_ControllerConfigurationService.HidePaths

Modified By: Extrinsics: CIM_ControllerConfigurationService.ExposePaths, CIM_ControllerConfigurationService.HidePaths

Deleted By: Extrinsics: CIM_ControllerConfigurationService.ExposePaths, CIM_ControllerConfigurationService.HidePaths

Requirement: Mandatory

Table 363 describes class CIM_AuthorizedTarget.

Table 363 - SMI Referenced Properties/Methods for CIM_AuthorizedTarget

Properties	Flags	Requirement	Description & Notes
TargetElement		Mandatory	The target set of resources to which the Privilege applies.
Privilege		Mandatory	The Privilege affecting the target resource.

18.8.5 CIM_ConcreteDependency (Associates ControllerConfiguirationService and ProtocolController)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 364 describes class CIM_ConcreteDependency (Associates ControllerConfiguirationService and ProtocolController).

Table 364 - SMI Referenced Properties/Methods for CIM_ConcreteDependency (Associates ControllerConfiguirationService and ProtocolController)

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

18.8.6 CIM_ConcreteDependency (Associates PrivilegeManagementService and AuthorizedPrivilege)

Deprecated.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 365 describes class CIM_ConcreteDependency (Associates PrivilegeManagementService and AuthorizedPrivilege).

Table 365 - SMI Referenced Properties/Methods for CIM_ConcreteDependency (Associates PrivilegeManagementService and AuthorizedPrivilege)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

18.8.7 CIM_ConcreteDependency (Associates StorageHardwareIDManagementService and StorageHardwareID)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 366 describes class CIM_ConcreteDependency (Associates StorageHardwareIDManagementService and StorageHardwareID).

Table 366 - SMI Referenced Properties/Methods for CIM_ConcreteDependency (Associates StorageHardwareIDManagementService and StorageHardwareID)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

18.8.8 CIM_ConcreteDependency (Associates StorageHardwareIDManagementService and SystemSpecificCollection)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Implementation support for collections of StorageHardwareIDs.

Table 367 describes class CIM_ConcreteDependency (Associates StorageHardwareIDManagementService and SystemSpecificCollection).

Table 367 - SMI Referenced Properties/Methods for CIM_ConcreteDependency (Associates StorageHardwareIDManagementService and SystemSpecificCollection)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

18.8.9 CIM_ControllerConfigurationService

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 368 describes class CIM_ControllerConfigurationService.

Table 368 - SMI Referenced Properties/Methods for CIM_ControllerConfigurationService

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	The scoping System CreationClassName.
SystemName		Mandatory	The scoping System Name.
CreationClassName		Mandatory	The name of the concrete subclass.
Name		Mandatory	Unique identifer for the Service.
ExposePaths()		Mandatory	
HidePaths()		Mandatory	
ExposeDefaultLUs()		Optional	
HideDefaultLUs()		Optional	
DeleteProtocolController()		Optional	

18.8.10 CIM_ElementCapabilities (EnabledLogicalElementCapabilities to ControllerConfigurationService)

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

Table 369 describes class CIM_ElementCapabilities (EnabledLogicalElementCapabilities to ControllerConfigurationService).

Table 369 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (EnabledLogicalElementCapabilities to ControllerConfigurationService)

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

18.8.11 CIM_ElementCapabilities (EnabledLogicalElementCapabilities to ProtocolController)

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

Table 370 describes class CIM_ElementCapabilities (EnabledLogicalElementCapabilities to ProtocolController).

Table 370 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (EnabledLogicalElementCapabilities to ProtocolController)

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

18.8.12 CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StorageHardwareID)

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

Table 371 describes class CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StorageHardwareID).

Table 371 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StorageHardwareID)

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

18.8.13 CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StorageHardwareIDManagementService)

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

Table 372 describes class CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StorageHardwareIDManagementService).

Table 372 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StorageHardwareIDManagementService)

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

18.8.14 CIM_ElementCapabilities (EnabledLogicalElementCapabilities to SystemSpecificCollection)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Implementation support for collections of StorageHardwareIDs.

Table 373 describes class CIM_ElementCapabilities (EnabledLogicalElementCapabilities to SystemSpecificCollection).

Table 373 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (EnabledLogicalElementCapabilities to SystemSpecificCollection)

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

18.8.15 CIM_ElementCapabilities (System to ProtocolControllerMaskingCapabilities)

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

Table 374 describes class CIM_ElementCapabilities (System to ProtocolControllerMaskingCapabilities).

Table 374 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (System to ProtocolControllerMaskingCapabilities)

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	
Capabilities		Mandatory	

18.8.16 CIM_ElementSettingData (Associates ComputerSystem and StorageClientSettingData)

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

Table 375 describes class CIM_ElementSettingData (Associates ComputerSystem and StorageClientSettingData).

Table 375 - SMI Referenced Properties/Methods for CIM_ElementSettingData (Associates ComputerSystem and StorageClientSettingData)

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	
SettingData		Mandatory	

18.8.17 CIM_ElementSettingData (Associates Port and StorageClientSettingData)

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

Table 376 describes class CIM_ElementSettingData (Associates Port and StorageClientSettingData).

Table 376 - SMI Referenced Properties/Methods for CIM_ElementSettingData (Associates Port and StorageClientSettingData)

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	
SettingData		Mandatory	

18.8.18 CIM_ElementSettingData (Associates ProtocolController and StorageClientSettingData)

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

Table 377 describes class CIM_ElementSettingData (Associates ProtocolController and StorageClientSettingData).

Table 377 - SMI Referenced Properties/Methods for CIM_ElementSettingData (Associates ProtocolController and StorageClientSettingData)

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	
SettingData		Mandatory	

18.8.19 CIM_ElementSettingData (Associates StorageHardwareID and StorageClientSettingData)

Created By: Extrinsic: CIM_StorageHardwareIDManagementService.CreateStorageHardwareID
Modified By: Static
Deleted By: Extrinsic: CIM_StorageHardwareIDManagementService.DeleteStorageHardwareID
Requirement: Optional

Table 378 describes class CIM_ElementSettingData (Associates StorageHardwareID and StorageClientSettingData).

Table 378 - SMI Referenced Properties/Methods for CIM_ElementSettingData (Associates StorageHardwareID and StorageClientSettingData)

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	
SettingData		Mandatory	

18.8.20 CIM_EnabledLogicalElementCapabilities

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

Table 379 describes class CIM_EnabledLogicalElementCapabilities.

Table 379 - SMI Referenced Properties/Methods for CIM_EnabledLogicalElementCapabilities

Properties	Flags	Requirement	Description & 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.

18.8.21 CIM_HostedCollection

Created By: Extrinsic: CIM_StorageHardwareIDManagementService.CreateHardwareIDCollection
Modified By: Static
Deleted By: Static
Requirement: Implementation support for collections of StorageHardwareIDs.

Table 380 describes class CIM_HostedCollection.

Table 380 - SMI Referenced Properties/Methods for CIM_HostedCollection

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

18.8.22 CIM_HostedService (Associates ComputerSystem and ControllerConfigurationService)

Created By: Static
Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 381 describes class CIM_HostedService (Associates ComputerSystem and ControllerConfigurationService).

Table 381 - SMI Referenced Properties/Methods for CIM_HostedService (Associates ComputerSystem and ControllerConfigurationService)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

18.8.23 CIM_HostedService (Associates ComputerSystem and PrivilegeManagementService)

Deprecated.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 382 describes class CIM_HostedService (Associates ComputerSystem and PrivilegeManagementService).

Table 382 - SMI Referenced Properties/Methods for CIM_HostedService (Associates ComputerSystem and PrivilegeManagementService)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

18.8.24 CIM_HostedService (Associates ComputerSystem and StorageHardwareIDManagementService)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 383 describes class CIM_HostedService (Associates ComputerSystem and StorageHardwareIDManagementService).

Table 383 - SMI Referenced Properties/Methods for CIM_HostedService (Associates ComputerSystem and StorageHardwareIDManagementService)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

18.8.25 CIM_MemberOfCollection

Created By: Extrinsics: CIM_StorageHardwareIDManagementService.CreateHardwareIDCollection, CIM_StorageHardwareIDManagementService.AddHardwareIDsToCollection

Modified By: Static

Deleted By: Static

Requirement: Implementation support for collections of StorageHardwareIDs.

Table 384 describes class CIM_MemberOfCollection.

Table 384 - SMI Referenced Properties/Methods for CIM_MemberOfCollection

Properties	Flags	Requirement	Description & Notes
Collection		Mandatory	
Member		Mandatory	

18.8.26 CIM_PrivilegeManagementService

Deprecated.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 385 describes class CIM_PrivilegeManagementService.

Table 385 - SMI Referenced Properties/Methods for CIM_PrivilegeManagementService

Properties	Flags	Requirement	Description & 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	

18.8.27 CIM_ProtocolController

Created By: Extrinsics: CIM_ControllerConfigurationService.ExposePaths, CIM_ControllerConfigurationService.HidePaths

Modified By: Extrinsics: CIM_ControllerConfigurationService.ExposePaths, CIM_ControllerConfigurationService.HidePaths

Deleted By: Extrinsics: CIM_ControllerConfigurationService.ExposePaths, CIM_ControllerConfigurationService.HidePaths

Requirement: Mandatory

Table 386 describes class CIM_ProtocolController.

Table 386 - SMI Referenced Properties/Methods for CIM_ProtocolController

Properties	Flags	Requirement	Description & 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.

18.8.28 CIM_ProtocolControllerForUnit

Created By: Extrinsics: CIM_ControllerConfigurationService.ExposePaths, CIM_ControllerConfigurationService.HidePaths, CIM_ControllerConfigurationService.ExposeDefaultLUs, CIM_ControllerConfigurationService.HideDefaultLUs

Modified By: Extrinsics: CIM_ControllerConfigurationService.ExposePaths, CIM_ControllerConfigurationService.HidePaths, CIM_ControllerConfigurationService.ExposeDefaultLUs, CIM_ControllerConfigurationService.HideDefaultLUs

Deleted By: Extrinsics: CIM_ControllerConfigurationService.ExposePaths, CIM_ControllerConfigurationService.HidePaths, CIM_ControllerConfigurationService.ExposeDefaultLUs, CIM_ControllerConfigurationService.HideDefaultLUs

Requirement: Mandatory

Table 387 describes class CIM_ProtocolControllerForUnit.

Table 387 - SMI Referenced Properties/Methods for CIM_ProtocolControllerForUnit

Properties	Flags	Requirement	Description & Notes
DeviceNumber		Mandatory	Address (e.g. LUN) of the associated Device. Shall be formatted as unseparated uppercase hexadecimal digits, with no leading 0x.
DeviceAccess		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).

18.8.29 CIM_ProtocolControllerMaskingCapabilities

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

Table 388 describes class CIM_ProtocolControllerMaskingCapabilities.

Table 388 - SMI Referenced Properties/Methods for CIM_ProtocolControllerMaskingCapabilities

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	Opaque and unique identifer.
ElementName		Mandatory	User-friendly name.
ValidHardwareIdTypes		Mandatory	A list of the valid values for StrorageHardwareID.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 ProtocolContollers attached to a SCSIProtocolEndpoint can expose the same unit numbers (e.g. multiple LUN 0s) or if the numbers must 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.

Table 388 - SMI Referenced Properties/Methods for CIM_ProtocolControllerMaskingCapabilities

Properties	Flags	Requirement	Description & Notes
SPCAllowsNoInitiators		Mandatory	Set to true if a client can create an SPC with no StorageHardwareIDs.
SPCSupportsDefaultViews		Mandatory	Set to true if it the instrumentation supports default view SPCs that exposes logical units to all initiators.

18.8.30 CIM_SAPAvailableForElement

Created By: Extrinsics: CIM_ControllerConfigurationService.ExposePaths, CIM_ControllerConfigurationService.HidePaths, CIM_ControllerConfigurationService.ExposeDefaultLUs, CIM_ControllerConfigurationService.HideDefaultLUs

Modified By: Extrinsics: CIM_ControllerConfigurationService.ExposePaths, CIM_ControllerConfigurationService.HidePaths, CIM_ControllerConfigurationService.ExposeDefaultLUs, CIM_ControllerConfigurationService.HideDefaultLUs

Deleted By: Extrinsics: CIM_ControllerConfigurationService.ExposePaths, CIM_ControllerConfigurationService.HidePaths, CIM_ControllerConfigurationService.ExposeDefaultLUs, CIM_ControllerConfigurationService.HideDefaultLUs

Requirement: Mandatory

Table 389 describes class CIM_SAPAvailableForElement.

Table 389 - SMI Referenced Properties/Methods for CIM_SAPAvailableForElement

Properties	Flags	Requirement	Description & Notes
AvailableSAP		Mandatory	
ManagedElement		Mandatory	

18.8.31 CIM_StorageClientSettingData

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 390 describes class CIM_StorageClientSettingData.

Table 390 - SMI Referenced Properties/Methods for CIM_StorageClientSettingData

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	Opaque and unique identifier.
ElementName		Mandatory	A user-friendly name.
ClientTypes		Mandatory	Array of OS names.

18.8.32 CIM_StorageHardwareID

Created By: Extrinsics: CIM_StorageHardwareIDManagementService.CreateStorageHardwareID, CIM_ControllerConfigurationService.ExposePaths

Modified By: Static

Deleted By: Extrinsic: CIM_StorageHardwareIDManagementService.DeleteStorageHardwareID

Requirement: Mandatory

Table 391 describes class CIM_StorageHardwareID.

Table 391 - SMI Referenced Properties/Methods for CIM_StorageHardwareID

Properties	Flags	Requirement	Description & 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).

18.8.33 CIM_StorageHardwareIDManagementService

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 392 describes class CIM_StorageHardwareIDManagementService.

Table 392 - SMI Referenced Properties/Methods for CIM_StorageHardwareIDManagementService

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	The scoping System CreationClassName.
SystemName		Mandatory	The scoping System Name.
CreationClassName		Mandatory	The name of the concrete subclass.
Name		Mandatory	Uniquely identifies the Service.
CreateStorageHardwareID()		Mandatory	
DeleteStorageHardwareID()		Mandatory	

Table 392 - SMI Referenced Properties/Methods for CIM_StorageHardwareIDManagementService

Properties	Flags	Requirement	Description & Notes
CreateHardwareIDCollection()		Optional	
AddHardwareIDsToCollection()		Optional	

18.8.34 CIM_SystemSpecificCollection

Created By: Extrinsic: CIM_StorageHardwareIDManagementService.CreateHardwareIDCollection

Modified By: Static

Deleted By: Static

Requirement: Implementation support for collections of StorageHardwareIDs.

Table 393 describes class CIM_SystemSpecificCollection.

Table 393 - SMI Referenced Properties/Methods for CIM_SystemSpecificCollection

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	Opaque and unique identifier.
ElementName		Mandatory	A user-friendly name.

18.8.35 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 394 describes class SNIA_ProtocolControllerMaskingCapabilities.

Table 394 - SMI Referenced Properties/Methods for SNIA_ProtocolControllerMaskingCapabilities

Properties	Flags	Requirement	Description & 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.

18.8.36 SNIA_StorageHardwareID

Created By: Extrinsics: CIM_StorageHardwareIDManagementService.CreateStorageHardwareID, CIM_ControllerConfigurationService.ExposePaths

Modified By: Static

Deleted By: Extrinsic: CIM_StorageHardwareIDManagementService.DeleteStorageHardwareID

Requirement: Optional

Table 395 describes class SNIA_StorageHardwareID.

Table 395 - SMI Referenced Properties/Methods for SNIA_StorageHardwareID

Properties	Flags	Requirement	Description & 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).

18.8.37 SNIA_StorageHardwareIDManagementService

Experimental subclass with support for SAS StorageHardwareIDs.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 396 describes class SNIA_StorageHardwareIDManagementService.

Table 396 - SMI Referenced Properties/Methods for SNIA_StorageHardwareIDManagementService

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	The scoping System CreationClassName.
SystemName		Mandatory	The scoping System Name.
CreationClassName		Mandatory	The name of the concrete subclass.
Name		Mandatory	Uniquely identifies the Service.
CreateStorageHardwareID()		Mandatory	Experimental: may use SAS Address IDType.
DeleteStorageHardwareID()		Mandatory	
CreateHardwareIDCollection()		Optional	
AddHardwareIDsToCollection()		Optional	

STABLE

DEPRECATED

Clause 19: Pool Manipulation Capabilities, and Settings Subprofile

The functionality of the LUN Creation and Pool Manipulation Capabilities, and Settings Subprofiles has been subsumed by the Clause 5: Block Services Package.

The Pool Manipulation Capabilities, and Settings Subprofile is defined in section 7.3.3.10 of SMI-S 1.0.2.

DEPRECATED

EXPERIMENTAL

Clause 20: Storage Server Asymmetry Profile

20.1 Description

20.1.1 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 as affinity provide full bandwidth access to the resource. The ports on the “other” storage processors provide full, limited, or standby access, depending on implementation

20.1.2 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 Multiple Computer System.

20.1.3 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 is does not formally extend Masking and Mapping but augments it’s 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.

20.1.4 Relationship to T10

This subprofile supports the passive management of the functionality defined in the Target Port Group Access States clause of the T10 SPC-4 specification.

20.1.5 Behavior, Characteristics, and Capabilities

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.

20.1.5.1 Port Failover

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

20.1.5.1.1 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.

20.1.5.1.2 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.

20.1.5.2 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 20.1.5.3 "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 the 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 affinitied to “other” through ports on “this”.

20.1.5.3 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.

20.1.6 Model

20.1.6.1 Classes

This profile introduces five new classes. These include one capabilities class, two collections, and two associations, shown in Figure 87.

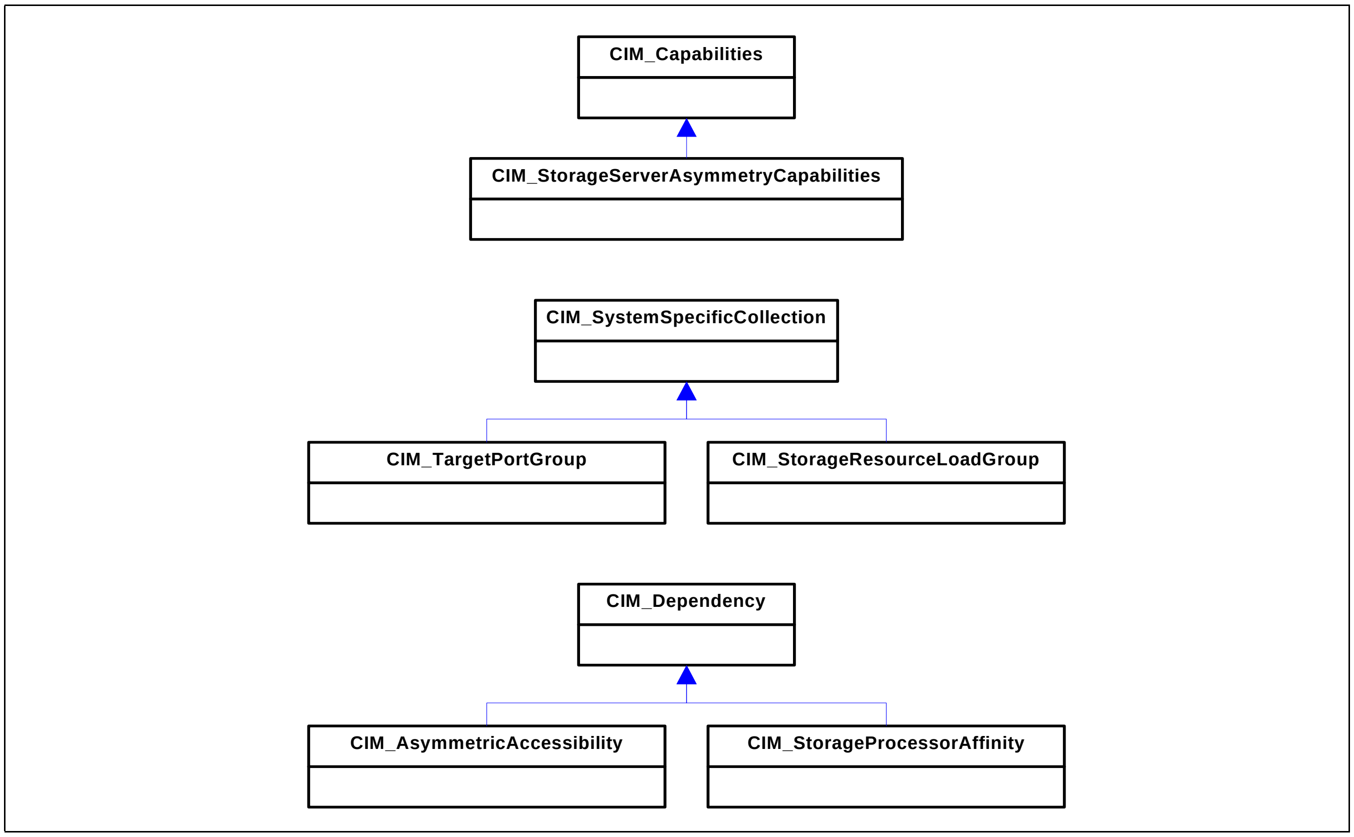
CIM_Capabilities
CIM_StorageServerAsymmetryCapabilities
CIM_SystemSpecificCollection
CIM_TargetPortGroup
CIM_StorageResourceLoadGroup
CIM_Dependency
CIM_AsymmetricAccessibility
CIM_StorageProcessorAffinity

Figure 87 - Storage Asymmetry Class Hierarchy

20.1.6.1.1 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 20.8 "CIM Elements".

20.1.6.1.2 TargetPortGroup

This sub-class 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 ISA CIM_SystemSpecificCollection there must 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.

20.1.6.1.2.1 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.

20.1.6.1.3 StorageResourceLoadGroup

This sub-class 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 20.1.6.1.3.1.

Because CIM_StorageResourceLoadGroup ISA CIM_SystemSpecificCollection there must 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.

20.1.6.1.3.1 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.

20.1.6.1.4 StorageProcessorAffinity

This sub-class 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.

20.1.6.1.5 Asymmetric Accessibility

This sub-class of CIM_Dependency associates instances of StorageResourceLoadGroup in a Redundancy Set to each instance of CIM_CIM_TargetPortGroup in the same RedundancySet. The AccessiblityState surfaces both the current and normal (healthy) accessibility of volumes in the LoadGroup from ports in the Port Group.

20.1.6.2 Instance Diagrams

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 88 shows the Asymmetry instances in context of the Multiple Computer System Profile for a dual redundant storage server.

Figure 88, Figure 89, Figure 90, and Figure 91 do not show the RedundancySet-related classes.

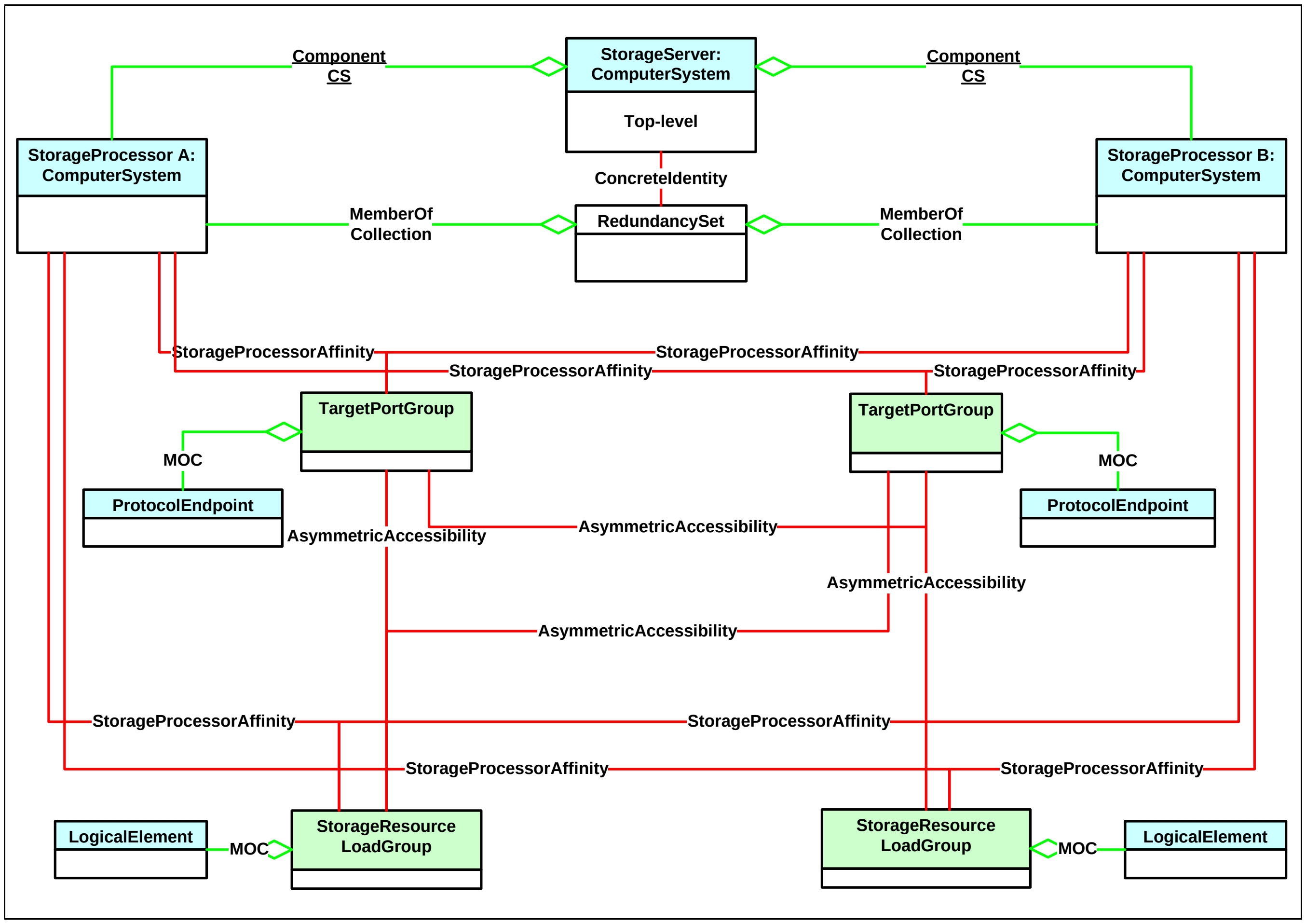

Figure 88 - Asymmetry with MCS

20.1.6.2.1 Multiple Tiers of Systems

Not shown is a system that has three tiers (see Clause 30: Multiple Computer System Subprofile in *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4*). This type of system may aggregate storage processors into more than one redundant-failover sub-system. 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.

20.1.6.2.2 Non-Transparent Asymmetry Cases

Figure 89: "Ports Do Not Failover, Healthy" and Figure 90: "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 failover, there is no need for a StorageResourceAffinity association from the Target Port Group on the storage processor to which the ports belong to the “Other” storage processor.

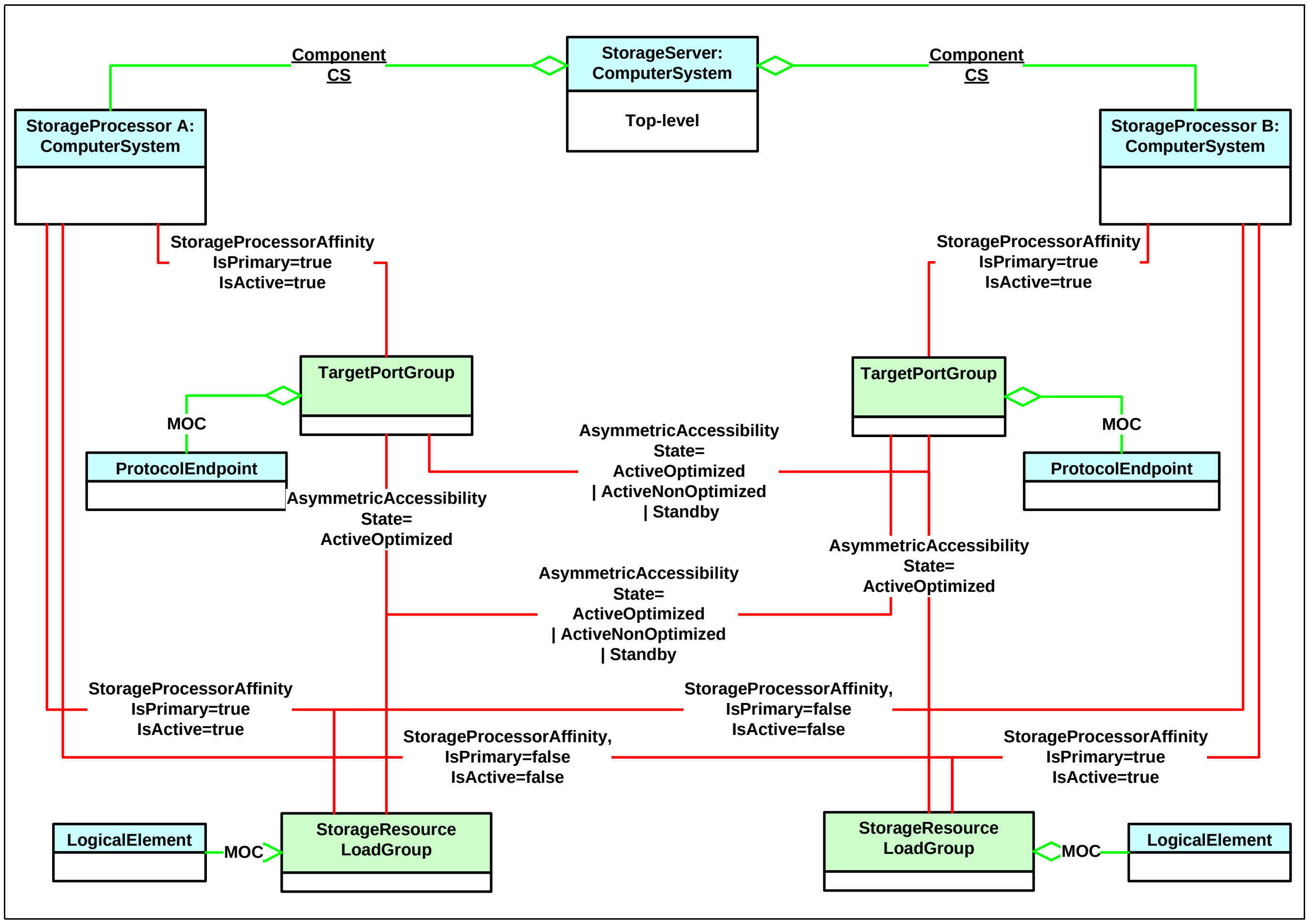
Component
CS
StorageServer: ComputerSystem
Top-level
Component
CS
StorageProcessor A: ComputerSystem
StorageProcessor B: ComputerSystem
StorageProcessorAffinity IsPrimary=true IsActive=true
StorageProcessorAffinity IsPrimary=true IsActive=true
TargetPortGroup
TargetPortGroup
MOC
MOC
ProtocolEndpoint
ProtocolEndpoint
AsymmetricAccessibility State= ActiveOptimized | ActiveNonOptimized | Standby
AsymmetricAccessibility State= ActiveOptimized
AsymmetricAccessibility State= ActiveOptimized
AsymmetricAccessibility State= ActiveOptimized | ActiveNonOptimized | Standby
StorageProcessorAffinity IsPrimary=true IsActive=true
StorageProcessorAffinity, IsPrimary=false IsActive=false
StorageProcessorAffinity, IsPrimary=false IsActive=false
StorageProcessorAffinity IsPrimary=true IsActive=true
LogicalElement
MOC
StorageResource LoadGroup
StorageResource LoadGroup
MOC
LogicalElement

Figure 89 - Ports Do Not Failover, Healthy

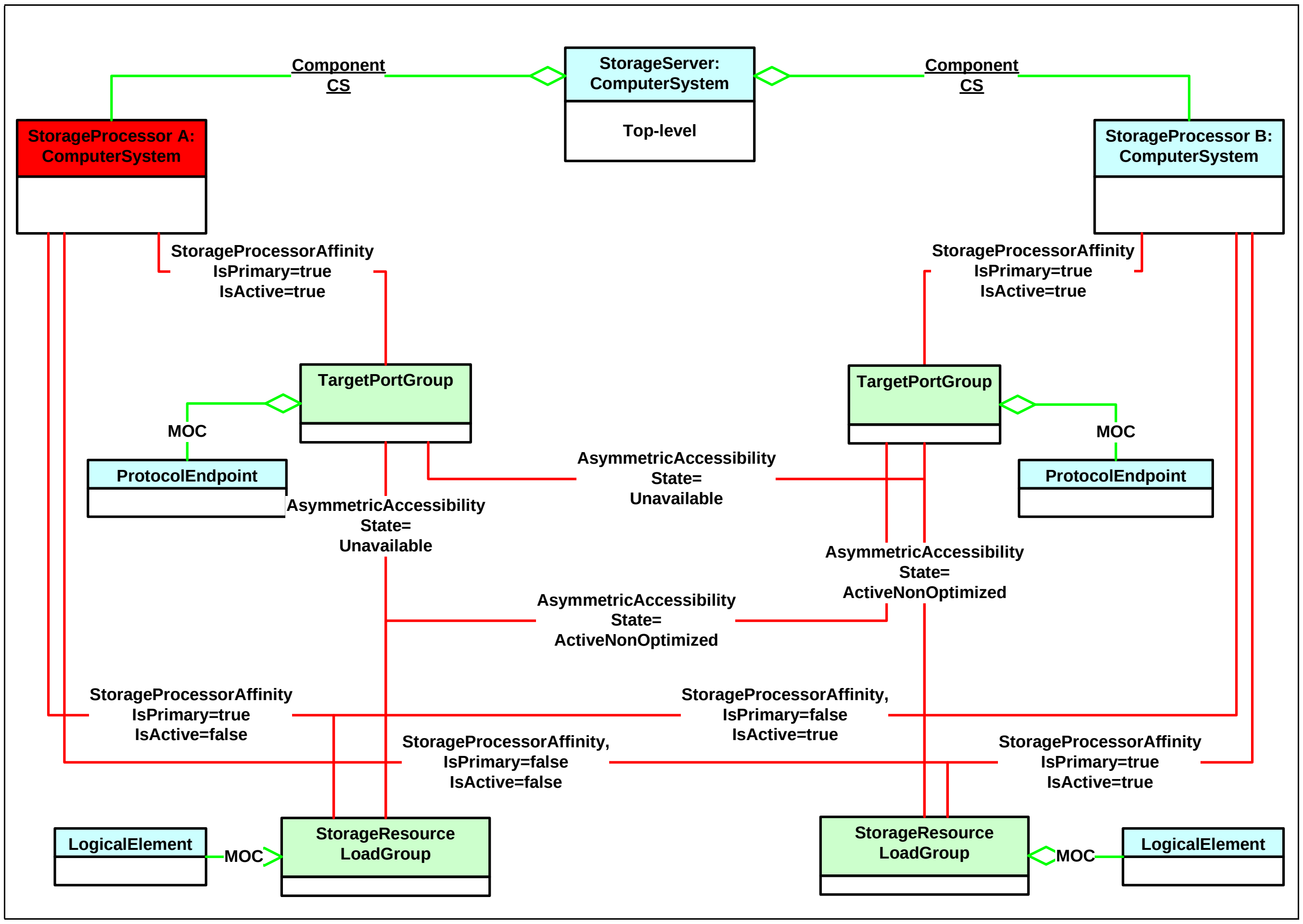

Figure 90 - Ports Do Not Failover, Failed Controller

20.1.6.2.3 Transparent Asymmetry Cases

Figure 91: "Ports Failover, Healthy" and Figure 92: "Ports Failover, Failed Controller" are instance diagrams that show the model for healthy and failed situations in a transparent failover port implementation.

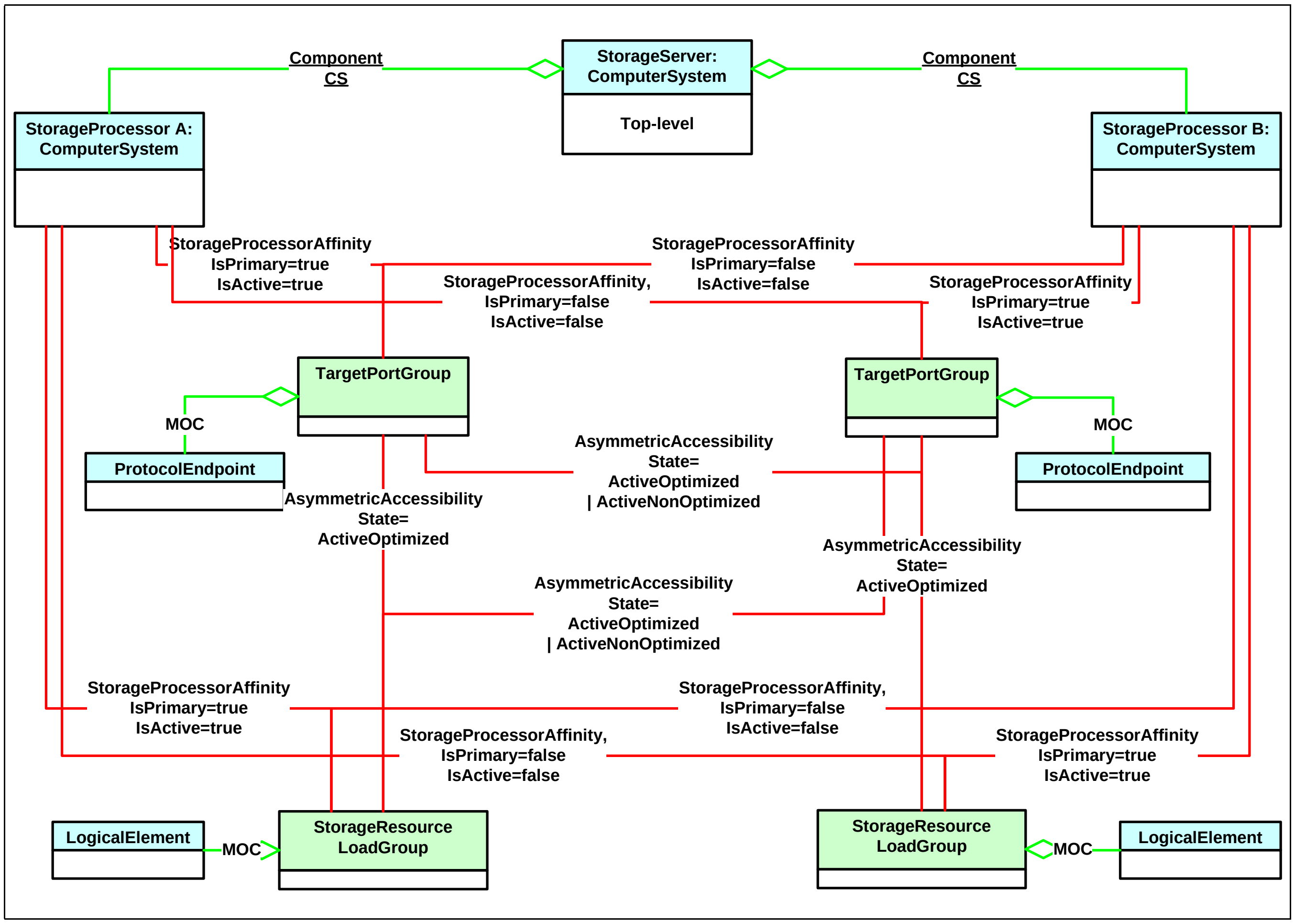
Component
CS
StorageServer: ComputerSystem
Top-level
Component
CS
StorageProcessor A: ComputerSystem
StorageProcessor B: ComputerSystem
StorageProcessorAffinity IsPrimary=true IsActive=true
StorageProcessorAffinity IsPrimary=false IsActive=false
StorageProcessorAffinity, IsPrimary=false IsActive=false
StorageProcessorAffinity IsPrimary=true IsActive=true
TargetPortGroup
TargetPortGroup
MOC
MOC
ProtocolEndpoint
ProtocolEndpoint
AsymmetricAccessibility State= ActiveOptimized | ActiveNonOptimized
AsymmetricAccessibility State= ActiveOptimized
AsymmetricAccessibility State= ActiveOptimized
AsymmetricAccessibility State= ActiveOptimized | ActiveNonOptimized
StorageProcessorAffinity IsPrimary=true IsActive=true
StorageProcessorAffinity, IsPrimary=false IsActive=false
StorageProcessorAffinity, IsPrimary=false IsActive=false
StorageProcessorAffinity IsPrimary=true IsActive=true
LogicalElement
MOC
StorageResource LoadGroup
StorageResource LoadGroup
MOC
LogicalElement

Figure 91 - Ports Failover, Healthy

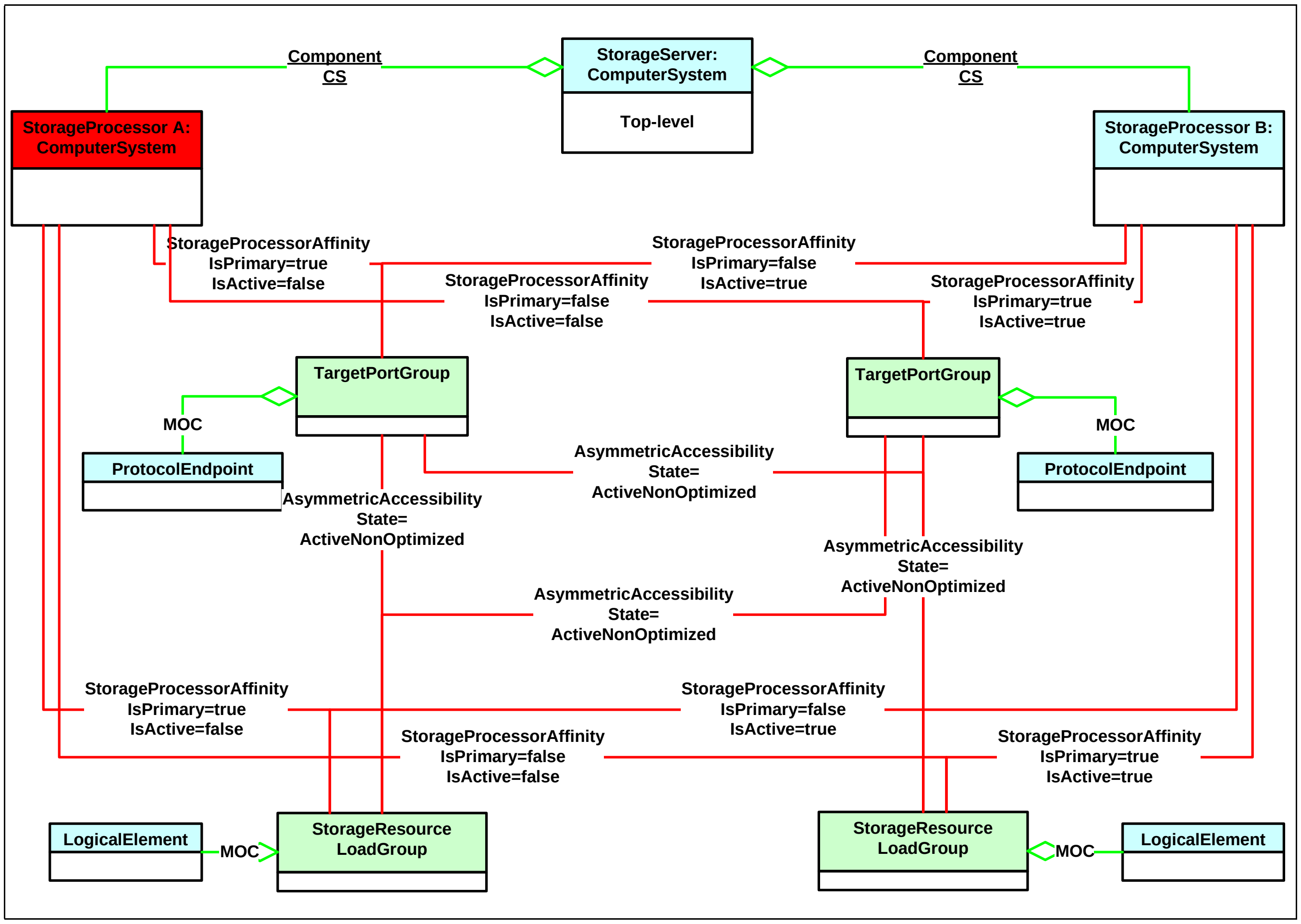
Component
CS
StorageServer: ComputerSystem
Top-level
Component
CS
StorageProcessor A: ComputerSystem
StorageProcessor B: ComputerSystem
StorageProcessorAffinity IsPrimary=true IsActive=false
StorageProcessorAffinity IsPrimary=false IsActive=true
StorageProcessorAffinity IsPrimary=false IsActive=false
StorageProcessorAffinity IsPrimary=true IsActive=true
TargetPortGroup
TargetPortGroup
MOC
MOC
ProtocolEndpoint
ProtocolEndpoint
AsymmetricAccessibility State= ActiveNonOptimized
AsymmetricAccessibility State= ActiveNonOptimized
AsymmetricAccessibility State= ActiveNonOptimized
AsymmetricAccessibility State= ActiveNonOptimized
StorageProcessorAffinity IsPrimary=true IsActive=false
StorageProcessorAffinity IsPrimary=false IsActive=true
StorageProcessorAffinity IsPrimary=false IsActive=false
StorageProcessorAffinity IsPrimary=true IsActive=true
LogicalElement
MOC
StorageResource LoadGroup
StorageResource LoadGroup
MOC
LogicalElement

Figure 92 - Ports Failover, Failed Controller

20.2 Health and Fault Management Consideration

None

20.3 Cascading Considerations

None.

20.4 Supported Profiles, Subprofiles, and Packages

None.

20.5 Methods of the Profile

20.5.1 Assign Storage Resource Affinity

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 a 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.

20.5.1.1 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

20.6 Client Considerations and Recipes

20.6.1 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].StorageResourceSymmetryCapability == 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.>
```

20.7 Registered Name and Version

Storage Server Asymmetry version 1.4.0 (Component Profile)

20.8 CIM Elements

Table 397 describes the CIM elements for Storage Server Asymmetry.

Table 397 - CIM Elements for Storage Server Asymmetry

Element Name	Requirement	Description
20.8.1 CIM_AsymmetricAccessibility	Mandatory	This association indicates the accessibility of StorageVolumes in the StorageResourceLoadGroup through ports in the associated TargetPortGroup.
20.8.2 CIM_ElementCapabilities (To Top-level ComputerSystem)	Mandatory	

Table 397 - CIM Elements for Storage Server Asymmetry

Element Name	Requirement	Description
20.8.3 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.
20.8.4 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.
20.8.5 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.
20.8.6 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.
20.8.7 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.
20.8.8 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.
20.8.9 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.
20.8.10 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.
20.8.11 CIM_StorageConfigurationService	Optional	

Table 397 - CIM Elements for Storage Server Asymmetry

Element Name	Requirement	Description
20.8.12 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.
20.8.13 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.
20.8.14 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 failback/ 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.
20.8.15 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.

Table 397 - CIM Elements for Storage Server Asymmetry

Element Name	Requirement	Description
20.8.16 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.

Table 397 - CIM Elements for Storage Server Asymmetry

Element Name	Requirement	Description
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.

20.8.1 CIM_AsymmetricAccessibility

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

Table 398 describes class CIM_AsymmetricAccessibility.

Table 398 - SMI Referenced Properties/Methods for CIM_AsymmetricAccessibility

Properties	Flags	Requirement	Description & 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 by the T10 SPC-4 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 by the T10 SPC-4 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.

20.8.2 CIM_ElementCapabilities (To Top-level ComputerSystem)

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

Table 399 describes class CIM_ElementCapabilities (To Top-level ComputerSystem).

Table 399 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (To Top-level ComputerSystem)

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	The Top-level Storage Sever ComputerSystem.
Capabilities		Mandatory	StorageServerAsymmetryCapabilities.

20.8.3 CIM_HostedCollection (Top-Level System to Load Group)

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

Table 400 describes class CIM_HostedCollection (Top-Level System to Load Group).

Table 400 - SMI Referenced Properties/Methods for CIM_HostedCollection (Top-Level System to Load Group)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

20.8.4 CIM_HostedCollection (Top-Level System to Port Group)

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

Table 401 describes class CIM_HostedCollection (Top-Level System to Port Group).

Table 401 - SMI Referenced Properties/Methods for CIM_HostedCollection (Top-Level System to Port Group)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

20.8.5 CIM_MemberOfCollection (SATA Target Port Group)

Created By: Static
Modified By: External
Deleted By: Static
Requirement: Requires TargetPortGroup to aggregate CIM_ProtocolEndpoint.

Table 402 describes class CIM_MemberOfCollection (SATA Target Port Group).

Table 402 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (SATA Target Port Group)

Properties	Flags	Requirement	Description & Notes
Collection		Mandatory	The Target Port Group.
Member		Mandatory	The SATA Target Ports.

20.8.6 CIM_MemberOfCollection (SB Target Port Group)

Created By: Static

Modified By: External

Deleted By: Static

Requirement: Requires TargetPortGroup to aggregate SNIA_SBProtocolEndpoint.

Table 403 describes class CIM_MemberOfCollection (SB Target Port Group).

Table 403 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (SB Target Port Group)

Properties	Flags	Requirement	Description & Notes
Collection		Mandatory	The Target Port Group.
Member		Mandatory	The The SB Target Ports.

20.8.7 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 404 describes class CIM_MemberOfCollection (SCSI Target Port Group).

Table 404 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (SCSI Target Port Group)

Properties	Flags	Requirement	Description & Notes
Collection		Mandatory	The Target Port Group.
Member		Mandatory	The DA, FC, SPI, or SAS Target Ports.

20.8.8 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 405 describes class CIM_MemberOfCollection (Storage Resource Load Group aggregating Storage Pools).

Table 405 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Storage Resource Load Group aggregating Storage Pools)

Properties	Flags	Requirement	Description & Notes
Collection		Mandatory	The Storage Resource Load Group.
Member		Mandatory	The StoragePools.

20.8.9 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 406 describes class CIM_MemberOfCollection (Storage Resource Load Group aggregating Storage Volumes).

Table 406 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Storage Resource Load Group aggregating Storage Volumes)

Properties	Flags	Requirement	Description & Notes
Collection		Mandatory	The Storage Resource Load Group.
Member		Mandatory	The Storage Volumes.

20.8.10 CIM_MemberOfCollection (iSCSI Target Port Group)

Created By: Static
Modified By: External
Deleted By: Static
Requirement: Requires TargetPortGroup to aggregate CIM_iSCSIProtocolEndpoint.

Table 407 describes class CIM_MemberOfCollection (iSCSI Target Port Group).

Table 407 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (iSCSI Target Port Group)

Properties	Flags	Requirement	Description & Notes
Collection		Mandatory	The Target Port Group.
Member		Mandatory	The iSCSI Target Ports.

20.8.11 CIM_StorageConfigurationService

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

Table 408 describes class CIM_StorageConfigurationService.

Table 408 - SMI Referenced Properties/Methods for CIM_StorageConfigurationService

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
CreationClassName		Mandatory	
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 StorageServerAsymmetryCapabilites 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.

20.8.12 CIM_StorageProcessorAffinity (StorageResourceLoadGroup)

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

Table 409 describes class CIM_StorageProcessorAffinity (StorageResourceLoadGroup).

Table 409 - SMI Referenced Properties/Methods for CIM_StorageProcessorAffinity (StorageResourceLoadGroup)

Properties	Flags	Requirement	Description & 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.

20.8.13 CIM_StorageProcessorAffinity (Target Port Group)

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

Table 410 describes class CIM_StorageProcessorAffinity (Target Port Group).

Table 410 - SMI Referenced Properties/Methods for CIM_StorageProcessorAffinity (Target Port Group)

Properties	Flags	Requirement	Description & 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.

20.8.14 CIM_StorageResourceLoadGroup (Load Groups)

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

20.8.15 CIM_StorageServerAsymmetryCapabilities

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

Table 411 describes class CIM_StorageServerAsymmetryCapabilities.

Table 411 - SMI Referenced Properties/Methods for CIM_StorageServerAsymmetryCapabilities

Properties	Flags	Requirement	Description & 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 are 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.

Table 411 - SMI Referenced Properties/Methods for CIM_StorageServerAsymmetryCapabilities

Properties	Flags	Requirement	Description & Notes
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. If this property has a value of 3, the TargetPortGroup will have a StorageProcessorAffinity association to each storage processor that can host it's 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 values 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.

20.8.16 CIM_TargetPortGroup (Port Groups)

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

EXPERIMENTAL

DEPRECATED

Clause 21: Block Services Resource Ownership Subprofile

Note: The Block Services Resource Ownership Subprofile is scheduled for removal for SMI-S 2.0. The functionality of this profile will not be replaced in SMI-S 2.0. The Storage Network Industry Association (SNIA) is not aware of any implementations of this profile. The SNIA would like to hear from anyone that has implemented the Block Services Resource Ownership Subprofile. If your company or organization has implemented this subprofile and is a member of the SNIA, please contact the DRM Technical Working Group or indicate your preference to keep this subprofile in SMI-S 2.0 during member reviews and ballots. If your company or organization has implemented this subprofile and is not a member of the SNIA, please indicate your preference to keep this subprofile as part of SMI-S using the SNIA feedback portal: http://www.snia.org/tech_activities/feedback/ .

21.1 Description

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 93. 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.

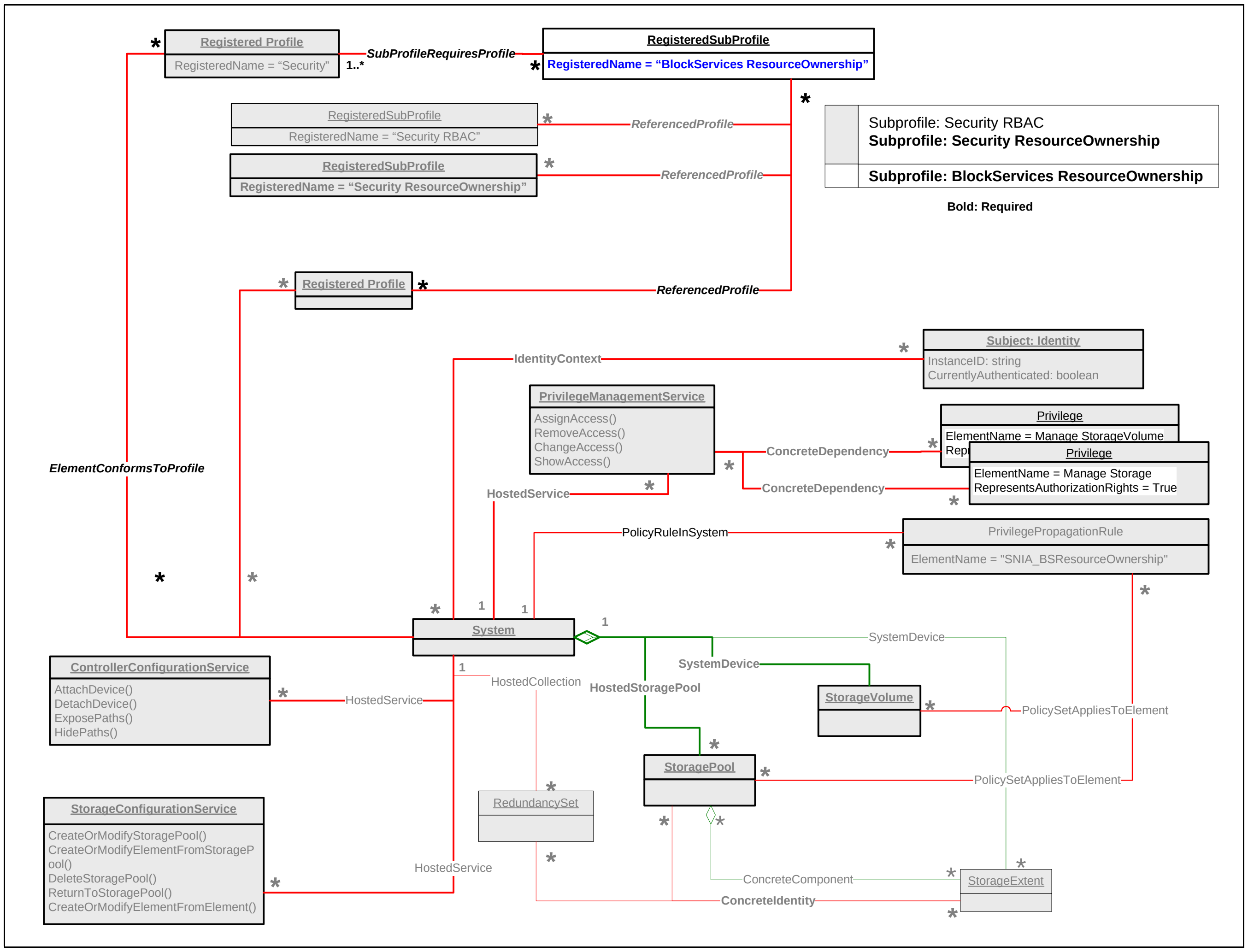
Registered Profile
RegisteredName = "Security"
SubProfileRequiresProfile
1..*
RegisteredSubProfile
RegisteredName = "BlockServices ResourceOwnership"
RegisteredSubProfile
RegisteredName = "Security RBAC"
RegisteredSubProfile
RegisteredName = "Security ResourceOwnership"
ReferencedProfile
Subprofile: Security RBAC
Subprofile: Security ResourceOwnership
Subprofile: BlockServices ResourceOwnership
Bold: Required
Registered Profile
ElementConformsToProfile
Subject: Identity
InstanceID: string
CurrentlyAuthenticated: boolean
IdentityContext
PrivilegeManagementService
AssignAccess()
RemoveAccess()
ChangeAccess()
ShowAccess()
Privilege
ElementName = Manage StorageVolume
Privilege
ElementName = Manage Storage
RepresentsAuthorizationRights = True
ConcreteDependency
HostedService
PolicyRuleInSystem
PrivilegePropagationRule
ElementName = "SNIA_BSResourceOwnership"
System
SystemDevice
HostedCollection
HostedStoragePool
StorageVolume
PolicySetAppliesToElement
ControllerConfigurationService
AttachDevice()
DetachDevice()
ExposePaths()
HidePaths()
StoragePool
RedundancySet
StorageConfigurationService
CreateOrModifyStoragePool()
CreateOrModifyElementFromStoragePool()
DeleteStoragePool()
ReturnToStoragePool()
CreateOrModifyElementFromElement()
ConcreteComponent
ConcreteIdentity
StorageExtent

Figure 93 - 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 412.

Table 412 - Block Service Management Rights

ElementName	Property	Index	Value
Manage StorageVolume	Activities	0	Execute
	QualifiersFormats	0	<class>.method

Table 412 - Block Service Management Rights

ElementName	Property	Index	Value
	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 affect. 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.

21.1.1 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 94: "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.

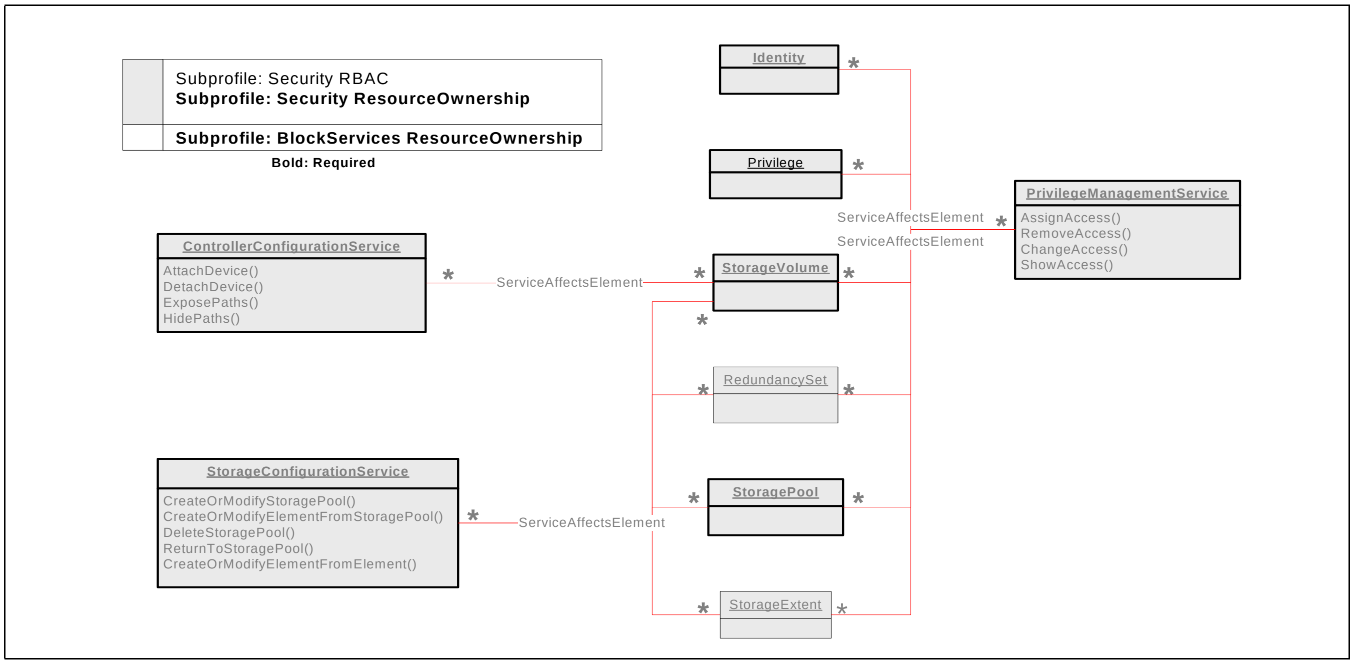

Figure 94 - ServiceAffectsElement Associations for ResourceOwnership

- AuthorizedPrivilege instances are assumed when a Privilege is granted to a subject or assigned to a target. (See Figure 95: "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

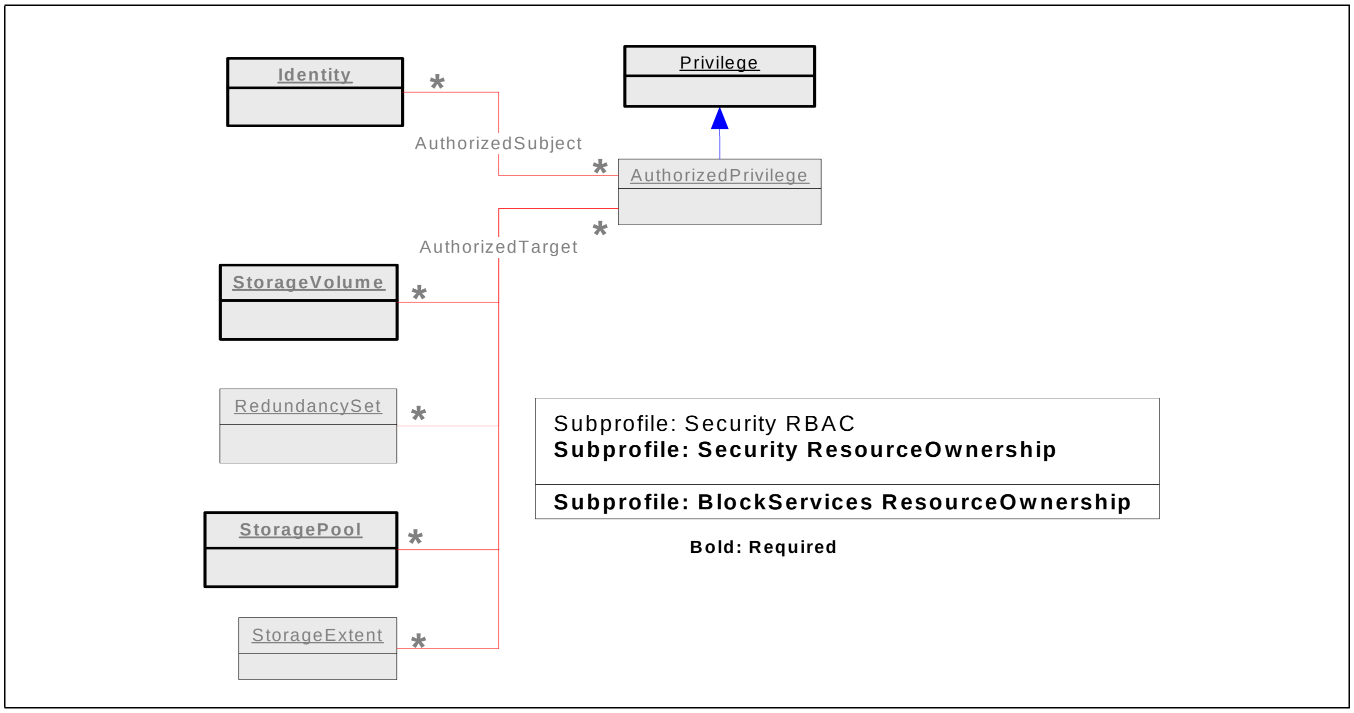

Figure 95 - AuthorizedPrivilege Associations for ResourceOwnership

- 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.

21.1.2 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.

21.2 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's 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

IMPLEMENTED

Clause 22: Storage Virtualizer Profile

22.1 Description

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 96: "Storage Virtualizer Package Diagram" extend this description and also enable configuration. Refer to 22.4 for more information on these optional extensions. This profile also includes the mandatory Clause 31: Physical Package Package (in *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4)* that describes the physical layout of the system and includes product identification information. The modeling in this document is split into various sections that describe how to model particular elements of an storage virtualizer system.

Figure 96: "Storage Virtualizer Package Diagram" illustrates the relationship between the packages related to the Storage Virtualizer Profile.

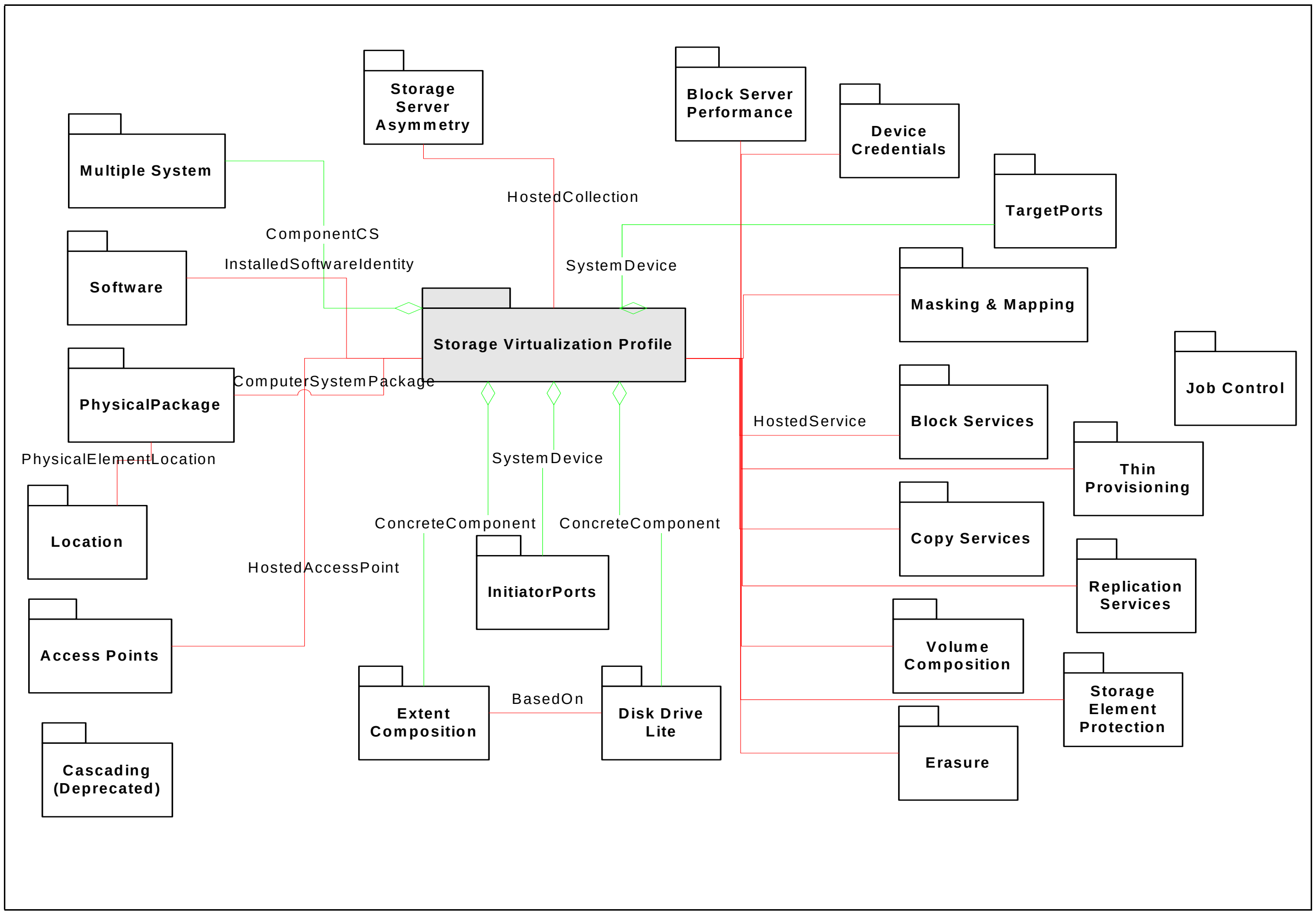

Figure 96 - Storage Virtualizer Package Diagram

22.1.1 Instance Diagram

The diagrams used in this document are 'Instance' diagrams implying the actual classes that you implement rather than the class hierarchy diagrams often used to show CIM models. This is felt to be easier to understand. Refer to the DMTF MOF files for information on class inheritance information and full information on the properties and methods used.

Figure 97: "Storage Virtualizer System Instance" is an instance diagram of a simple Storage Virtualization system.

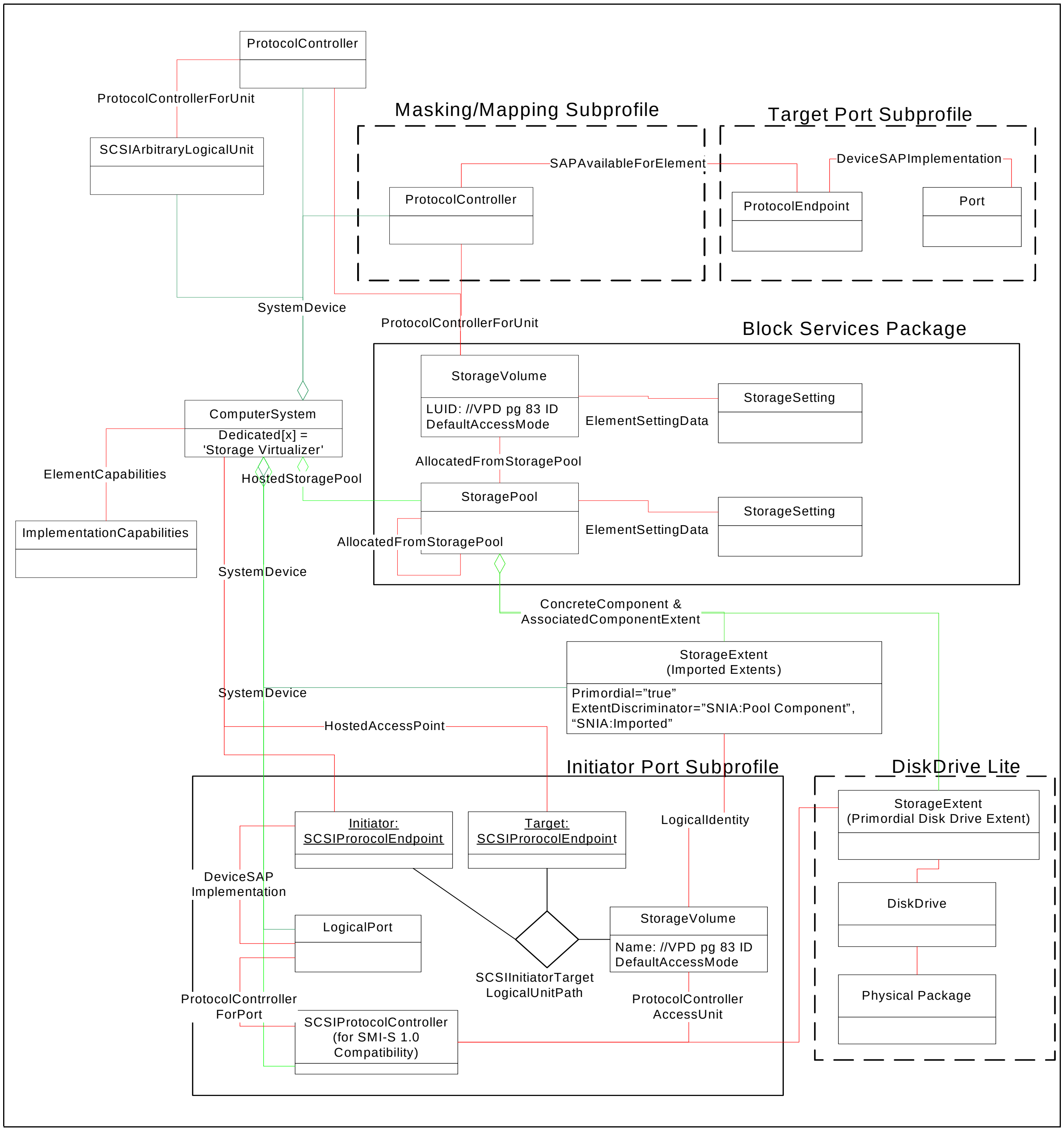

Figure 97 - Storage Virtualizer System Instance

22.1.2 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.

EXPERIMENTAL

The capabilities of the Storage Virtualizer implementation are identified in an instance of CIM_ImplementationCapabilities, which is associated to the top level Storage Virtualizer ComputerSystem via ElementCapabilities. This includes information on the capacity optimization techniques supported by the Sorage Virtualizer.

EXPERIMENTAL

The StoragePool classes in the center of the diagram represents the mapping from array storage to volumes for host access. The pool is hosted on the ComputerSystem and services to control it are host on the same controller. The StorageExtent at the bottom of the screen represents the storage from external arrays used by the mapping. These StorageExtents are connected to the pool using the ConcreteComponent association. 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.

22.1.3 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.

22.1.4 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.

22.1.5 Device Management Access

Most devices now have a web GUI to allow device specific configuration. This is modeled using the common subprofile “Access Point”.

22.1.6 Physical Modeling

The physical aspects of the storage virtualizer ComputerSystem are represented by the *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* Package Clause 31: "Physical Package Package" and the optional *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* Clause 27: "Location Subprofile", which provide more details.

22.1.7 Services

The system hosts services used to control the configuration of the system’s resources. These services are optional and modeled by Clause 5: "Block Services Package", Clause 9: "Copy Services Subprofile", and Clause 26: "Job Control Subprofile".

22.1.8 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 specification does not specify any particular port type be supported. In either target or initiator cases, the ports could be FC or iSCSI. All port subprofiles are documented in *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4*.

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

22.1.9 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

22.2 Health and Fault Management

Defined in the included subprofiles.

EXPERIMENTAL

22.3 Storage Virtualizer Support for Cascading

The Cascading Profile (see Clause 24: Cascading Subprofile in the *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4*) has been deprecated in favor of embedding the cascading related classes in the Storage Virtualizer Profile. The classes identified in this section identify the elements of Storage Virtualizer support for the cascading function.

Figure 98: "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 98: "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.

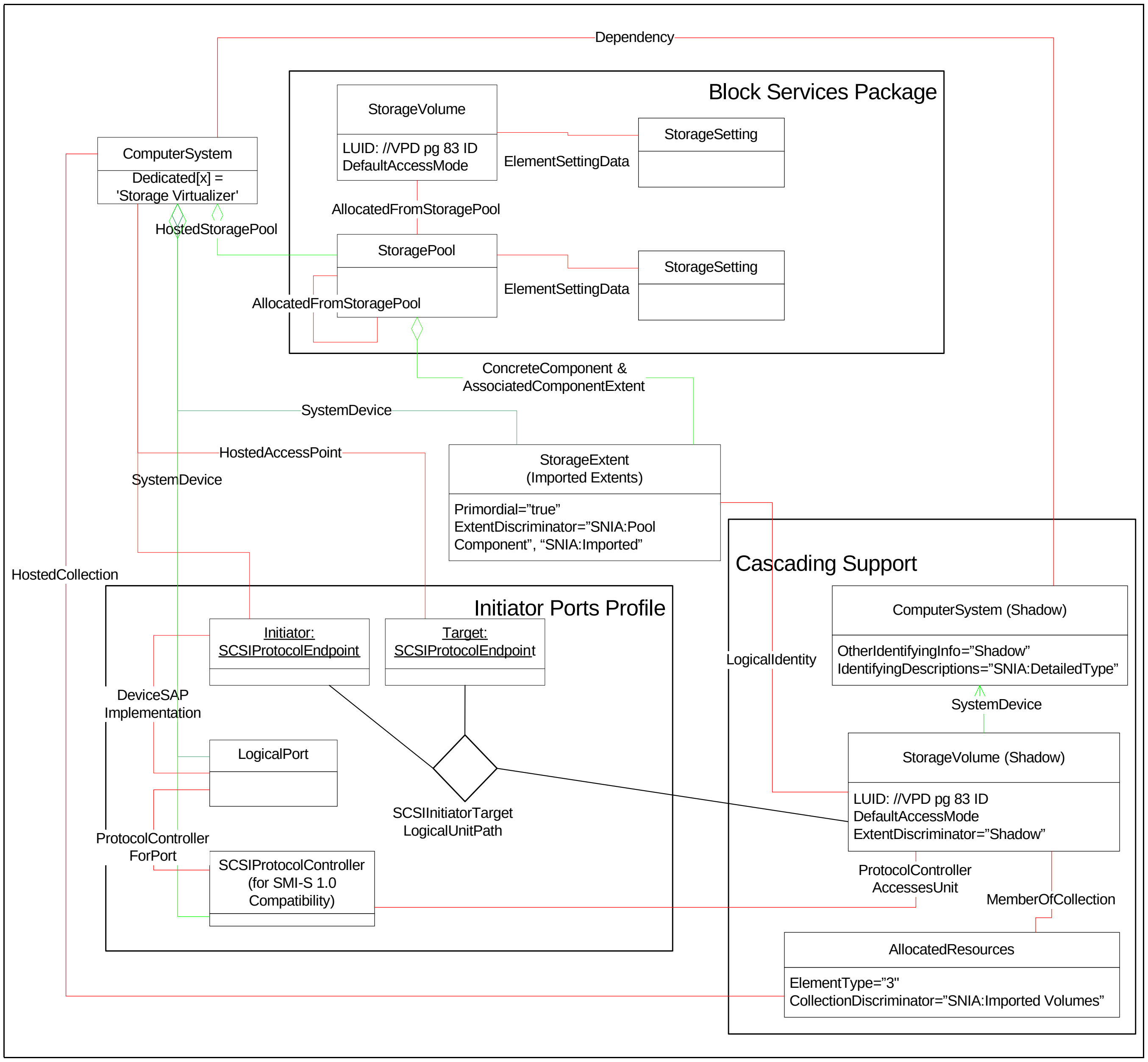

Figure 98 - Virtualizer, Cascading and Initiator Ports

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

22.4 Supported Subprofiles and Packages

Table 413 describes the supported profiles for Storage Virtualizer.

Table 413 - 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	1.6.0	Optional	Experimental.
CKD Block Services	SNIA	1.3.0	Optional	Experimental.
Cluster	SNIA	1.0.2	Optional	Deprecated. See Multiple Computer System
Extra Capacity Set	SNIA	1.0.2	Optional	Deprecated. See Multiple Computer System
Disk Drive	SNIA	1.0.2	Optional	Deprecated. See Disk Drive Lite
Disk Drive Lite	SNIA	1.6.0	Optional	
Extent Mapping	SNIA	1.0.2	Optional	Deprecated. See Extent Composition
Erasure	SNIA	1.2.0	Optional	Experimental.
Storage Server Asymmetry	SNIA	1.4.0	Optional	Experimental.
Volume Composition	SNIA	1.5.0	Optional	Experimental.
Storage Element Protection	SNIA	1.4.0	Optional	Experimental.
Copy Services	SNIA	1.5.0	Optional	
Pool Manipulation Capabilities and Settings	SNIA	1.0.2	Optional	Deprecated. See Block Services
LUN Creation	SNIA	1.0.2	Optional	Deprecated. See Block Services
Device Credentials	SNIA	1.3.0	Optional	
LUN Mapping and Masking	SNIA	1.0.2	Optional	Deprecated. See Masking and Mapping
Job Control	SNIA	1.5.0	Optional	
Location	SNIA	1.4.0	Optional	
Masking and Mapping	SNIA	1.6.0	Optional	
Software	SNIA	1.4.0	Optional	
Multiple Computer System	SNIA	1.2.0	Optional	

Table 413 - Supported Profiles for Storage Virtualizer

Profile Name	Organization	Version	Requirement	Description
Backend Ports	SNIA	1.0.2	Optional	Deprecated. See specific Initiator Ports
Disk Sparing	SNIA	1.5.0	Optional	
Extent Composition	SNIA	1.6.0	Optional	
Cascading	SNIA	1.3.0	Mandatory	Deprecated. This is deprecated in favor of embedding cascading elements in the Storage Virtualizer profile.
Block Services	SNIA	1.6.0	Mandatory	
Physical Package	SNIA	1.5.0	Mandatory	
Health	SNIA	1.2.0	Mandatory	
Thin Provisioning	SNIA	1.4.0	Optional	Experimental.
Replication Services	SNIA	1.6.0	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
iSCSI Target Ports	SNIA	1.6.0	Support for at least one is mandatory.	
FC Target Ports	SNIA	1.4.0		
SAS Target Ports	SNIA	1.4.0		Experimental.
SB Target Ports	SNIA	1.2.0		
FC Initiator Ports	SNIA	1.6.0		
iSCSI Initiator Ports	SNIA	1.2.0		Experimental.
SAS Initiator Ports	SNIA	1.4.0		Experimental.
ATA Initiator Ports	SNIA	1.4.0		Experimental.
SB Initiator Ports	SNIA	1.4.0		Experimental.
Indication	SNIA	1.5.0	Support for at least one is mandatory.	Deprecated. See the SNIA Indications Profile
Indications	SNIA	1.6.0		Experimental.
Indications	DMTF	1.2.0		Experimental. See DSP1054, version 1.2.0

22.5 Methods of the Profile

None.

22.6 Use Cases

EXPERIMENTAL

22.6.1 Discover the Capacity Optimization Support in an Storage Virtualizer

22.6.1.1 Summary

From a list of available Storage Virtualizer devices, determine which devices support any capacity optimization techniques.

22.6.1.2 Basic Course of Events

1) Administrator identifies an available virtualizer device.
2) Administrator determines if the virtualizer advertises implementation capabilities.
3) System responds with an implementation capabilities.
4) Administrator inspects the capacity optimization techniques supported by the virtualizer

22.6.1.3 Alternative Paths

none

22.6.1.4 Exception Paths

FAILED:

- The Storage Virtualizer System does not report implementation capabilities
- The Storage Virtualizer System reports implementation capabilities, but reports “none” for supported capacity optimizations.

22.6.1.5 Triggers

Device selection for provisioning storage for an application.

22.6.1.6 Assumptions

The administrator has a list of candidate storage virtualizer system names for doing provisioning.

22.6.1.7 Preconditions

The systems are available.

EXPERIMENTAL

22.7 Registered Name and Version

Storage Virtualizer version 1.6.0 (Autonomous Profile)

CIM Schema Version: 2.25

22.8 CIM Elements

Table 414 describes the CIM elements for Storage Virtualizer.

Table 414 - CIM Elements for Storage Virtualizer

Element Name	Requirement	Description
22.8.1 CIM_AssociatedComponentExtent (Pool Component to Primordial Pool)	Conditional	Conditional requirement: Implementation of the Extent Composition profile.
22.8.2 CIM_ComputerSystem (Shadow)	Mandatory	Experimental. 'Top level' system that represents a block storage device (e.g., an Array).
22.8.3 CIM_ComputerSystem (Top Level System)	Mandatory	'Top-level' system that represents the whole virtualizer. Associated to RegisteredProfile.
22.8.4 CIM_ConcreteComponent (Imported Extents to Primordial Pool)	Mandatory	Used to associate StorageExtents that are playing the Pool Component role to a Primordial StoragePool.
22.8.5 CIM_Dependency (Systems)	Mandatory	Experimental. This associates the block storage (e.g., Array) System to the Storage Virtualizer System.
22.8.6 CIM_ElementCapabilities (ImplementationCapabilities to System)	Optional	Experimental. Associates the conformant Storage Virtualizer ComputerSystem to the CIM_ImplementationCapabilities supported by the implementation.
22.8.7 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.
22.8.8 CIM_FilterCollection (Storage Virtualizer ProfileSpecificLifecycleIndicationFilterCollection)	Optional	Experimental. This is a collection of Lifecycle IndicationFilters defined by the Storage Virtualizer Profile.
22.8.9 CIM_HostedCollection (Allocated Resources)	Mandatory	Experimental. This would associate the AllocatedResources collection to the top level system for the Storage Virtualizer.
22.8.10 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.
22.8.11 CIM_HostedCollection (Storage Virtualizer to ProfileSpecificLifecycleIndicationFilterCollection)	Optional	Experimental.

Table 414 - CIM Elements for Storage Virtualizer

Element Name	Requirement	Description
22.8.12 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).
22.8.13 CIM_ImplementationCapabilities (ImplementationCapabilities)	Optional	Experimental. The capabilities of the profile implementation.
22.8.14 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.
22.8.15 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.
22.8.16 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.
22.8.17 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.

Table 414 - CIM Elements for Storage Virtualizer

Element Name	Requirement	Description
22.8.18 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.
22.8.19 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.
22.8.20 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.
22.8.21 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.
22.8.22 CIM_LogicalIdentity (Shadow Storage Volume)	Mandatory	Experimental. Associates a Storage Virtualizer StorageExtent to a shadow instance of an (imported) StorageVolume.
22.8.23 CIM_MemberOfCollection (Allocated Resources)	Mandatory	Experimental. This supports collecting StorageVolumes. This is required to support the AllocatedResources collection.
22.8.24 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.
22.8.25 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).
22.8.26 CIM_MemberOfCollection (Storage Virtualizer ProfileSpecificLifecycleIndicationFilterCollection to Storage Virtualizer Filters)	Optional	Experimental. This associates the Storage Virtualizer ProfileSpecificLifecycleIndicationFilterCollection to the Filters defined by the Storage Virtualizer Profile.

Table 414 - CIM Elements for Storage Virtualizer

Element Name	Requirement	Description
22.8.27 CIM_ProtocolControllerForUnit (Arbitrary LU for All LUNs View)	Conditional	Conditional requirement: Elements that are mandatory if Masking and Mapping is not implemented.
22.8.28 CIM_ProtocolControllerForUnit (Storage volumes for All LUNs View)	Conditional	Conditional requirement: Elements that are mandatory if Masking and Mapping is not implemented.
22.8.29 CIM_RemoteServiceAccessPoint (Shadow)	Optional	Experimental. CIM_RemoteServiceAccessPoint represents the management interface to a Shadow system.
22.8.30 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.
22.8.31 CIM_SCSIArbitraryLogicalUnit (Arbitrary LU)	Optional	A SCSI Logical Unit that exists only for management of the virtualizer.
22.8.32 CIM_SCSIProtocolController (All LUNs View)	Conditional	Conditional requirement: Elements that are mandatory if Masking and Mapping is not implemented.
22.8.33 CIM_StorageExtent (Imported Extents)	Mandatory	Used to represent the storage imported from external arrays and used as ConcreteComponents of Primordial StoragePools.
22.8.34 CIM_StorageVolume (Shadow)	Mandatory	Experimental. A shadow copy of a remote StorageVolume that is imported to the Storage Virtualizer.
22.8.35 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 StorageVolunmes with the System that manages them.
22.8.36 CIM_SystemDevice (System to SCSIArbitraryLogicalUnit)	Conditional	Conditional requirement: Elements that are mandatory if SCSIArbitraryLogicalUnit is instantiated. This association links SCSIArbitraryLogicalUnit to the scoping system.
22.8.37 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.

Table 414 - CIM Elements for Storage Virtualizer

Element Name	Requirement	Description
22.8.38 CIM_SystemDevice (System to StorageExtent)	Mandatory	This association links the primordial imported StorageExtent to the scoping system.
22.8.39 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).
22.8.40 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 section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 22.8.16 CIM_IndicationFilter (Storage Virtualizer System Creation).
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_ComputerSystem	Mandatory	Deletion of a ComputerSystem instance. See section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 22.8.17 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 section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 22.8.20 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 section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 22.8.15 CIM_IndicationFilter (Storage Virtualizer Storage Volume OperationalStatus).

Table 414 - CIM Elements for Storage Virtualizer

Element Name	Requirement	Description
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 section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 22.8.19 CIM_IndicationFilter (WQL Storage Virtualizer FCPort OperationalStatus).
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 section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 22.8.14 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 section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 22.8.21 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 section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 22.8.18 CIM_IndicationFilter (Storage Virtualizer System OperationalStatus).

22.8.1 CIM_AssociatedComponentExtent (Pool Component to Primordial Pool)

The referenced primordial imported 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 415 describes class CIM_AssociatedComponentExtent (Pool Component to Primordial Pool).

Table 415 - SMI Referenced Properties/Methods for CIM_AssociatedComponentExtent (Pool Component to Primordial Pool)

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	The Primordial StoragePool.
PartComponent		Mandatory	The imported storage extent that is a component of the primordial storage pool.

22.8.2 CIM_ComputerSystem (Shadow)

Experimental.

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

Table 416 describes class CIM_ComputerSystem (Shadow).

Table 416 - SMI Referenced Properties/Methods for CIM_ComputerSystem (Shadow)

Properties	Flags	Requirement	Description & Notes
CreationClassName		Mandatory	
Name		Mandatory	Unique identifier for the shadow system. E.g., IP address.
ElementName		Mandatory	User friendly name.
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' if the OtherIdentifyInfo is an IPv4 address). In the index for the OItherIdentifyingInfo 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.

22.8.3 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 '1.6.0'.

Table 417 describes class CIM_ComputerSystem (Top Level System).

Table 417 - SMI Referenced Properties/Methods for CIM_ComputerSystem (Top Level System)

Properties	Flags	Requirement	Description & Notes
CreationClassName		Mandatory	
Name	C	Mandatory	Unique identifier for the storage virtualizer. Eg 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.

22.8.4 CIM_ConcreteComponent (Imported Extents to Primordial Pool)

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

Table 418 describes class CIM_ConcreteComponent (Imported Extents to Primordial Pool).

Table 418 - SMI Referenced Properties/Methods for CIM_ConcreteComponent (Imported Extents to Primordial Pool)

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	A Primordial StoragePool.
PartComponent		Mandatory	The imported StorageExtent.

22.8.5 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 419 describes class CIM_Dependency (Systems).

Table 419 - SMI Referenced Properties/Methods for CIM_Dependency (Systems)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	The Storage Virtualizer top level System.
Dependent		Mandatory	The shadow System (e.g., system of the Array device).

22.8.6 CIM_ElementCapabilities (ImplementationCapabilities to System)

Experimental. Associates the conformant Storage Virtualizer ComputerSystem to the CIM_ImplementationCapabilities supported by the implementation.

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

Table 420 describes class CIM_ElementCapabilities (ImplementationCapabilities to System).

Table 420 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (ImplementationCapabilities to System)

Properties	Flags	Requirement	Description & Notes
Capabilities		Mandatory	The ImplementationCapabilities.
ManagedElement		Mandatory	The conformant Storage Virtualizer ComputerSystem that has ImplementationCapabilities.

22.8.7 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 421 describes class CIM_FilterCollection (Storage Virtualizer Predefined FilterCollection).

Table 421 - SMI Referenced Properties/Methods for CIM_FilterCollection (Storage Virtualizer Predefined FilterCollection)

Properties	Flags	Requirement	Description & 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:Predefined'.

22.8.8 CIM_FilterCollection (Storage Virtualizer ProfileSpecificLifecycleIndicationFilterCollection)

Experimental. This is a collection of Lifecycle IndicationFilters defined by the Storage Virtualizer Profile.

Requirement: Optional

Table 422 describes class CIM_FilterCollection (Storage Virtualizer ProfileSpecificLifecycleIndicationFilterCollection).

Table 422 - SMI Referenced Properties/Methods for CIM_FilterCollection (Storage Virtualizer ProfileSpecificLifecycleIndicationFilterCollection)

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	See the InstanceID definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.13 CIM_FilterCollection (StaticFilterCollection).
CollectionName		Mandatory	The value of CollectionName shall be 'SNIA:Storage Virtualizer:ProfileSpecificLifecycleIndicationFilterCollection'.

22.8.9 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 423 describes class CIM_HostedCollection (Allocated Resources).

Table 423 - SMI Referenced Properties/Methods for CIM_HostedCollection (Allocated Resources)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	The Top Level System of the Storage Virtualizer.
Dependent		Mandatory	The AllocatedResources collection of shadow storage volumes.

22.8.10 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 424 describes class CIM_HostedCollection (Remote Resources).

Table 424 - SMI Referenced Properties/Methods for CIM_HostedCollection (Remote Resources)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	The Top Level System of the Storage Virtualizer.
Dependent		Mandatory	The RemoteResources collection of shadow storage volumes.

22.8.11 CIM_HostedCollection (Storage Virtualizer to ProfileSpecificLifecycleIndicationFilterCollection)

Experimental.

Requirement: Optional

Table 425 describes class CIM_HostedCollection (Storage Virtualizer to ProfileSpecificLifecycleIndicationFilterCollection).

Table 425 - SMI Referenced Properties/Methods for CIM_HostedCollection (Storage Virtualizer to ProfileSpecificLifecycleIndicationFilterCollection)

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	Reference to the ProfileSpecificLifecycleIndicationFilterCollection for the Storage Virtualizer.
Antecedent		Mandatory	Reference to the 'Top level' Storage Virtualizer System.

22.8.12 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 426 describes class CIM_HostedCollection (Storage Virtualizer to predefined FilterCollection).

Table 426 - SMI Referenced Properties/Methods for CIM_HostedCollection (Storage Virtualizer to predefined FilterCollection)

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	Reference to the predefined FilterCollection for the Storage Virtualizer.
Antecedent		Mandatory	Reference to the 'Top level' Storage Virtualizer System.

22.8.13 CIM_ImplementationCapabilities (ImplementationCapabilities)

Experimental. The capabilities (features) of the profile implementation.

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

Table 427 describes class CIM_ImplementationCapabilities (ImplementationCapabilities).

Table 427 - SMI Referenced Properties/Methods for CIM_ImplementationCapabilities (ImplementationCapabilities)

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	An opaque, unique id for the implementation capability of an implementation.
ElementName		Optional	A provider supplied user-friendly name for this CIM_ImplementationCapabilities element.
SupportedCapacityOptimizations		Mandatory	This array of strings lists the capacity optimization techniques that are supported by the implementation. Valid string values are "none" | "SNIA:Thin Provisioning" | "SNIA:Data Compression" | "SNIA:Data Deduplication".
SupportedViews		Mandatory	This array of strings lists the view classes that are supported by the implementation. Valid string values are "none" | "SNIA:VolumeView" | "SNIA:DiskDriveView" | "SNIA:ExposedView" | "SNIA:MaskingMappingView" | "SNIA:MappingProtocolControllerView" | "SNIA:StoragePoolView" | "SNIA:ReplicaPairView" .

22.8.14 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 428 describes class CIM_IndicationFilter (Storage Virtualizer LogicalPort OperationalStatus).

Table 428 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Storage Virtualizer LogicalPort OperationalStatus)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Name		Mandatory	This shall be 'SNIA:Storage Virtualizer:LogicalPortOperationalStatus'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
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 section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).

22.8.15 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 429 describes class CIM_IndicationFilter (Storage Virtualizer Storage Volume OperationalStatus).

Table 429 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Storage Virtualizer Storage Volume OperationalStatus)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Name		Mandatory	This shall be 'SNIA:Storage Virtualizer:StorageVolumeOperationalStatus'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
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 section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).

22.8.16 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 430 describes class CIM_IndicationFilter (Storage Virtualizer System Creation).

Table 430 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Storage Virtualizer System Creation)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Name		Mandatory	This shall be 'SNIA:Storage Virtualizer:SystemCreation'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
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 *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).

22.8.17 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 431 describes class CIM_IndicationFilter (Storage Virtualizer System Deletion).

Table 431 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Storage Virtualizer System Deletion)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Name		Mandatory	This shall be 'SNIA:Storage Virtualizer:SystemDeletion'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
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 *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).

22.8.18 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 432 describes class CIM_IndicationFilter (Storage Virtualizer System OperationalStatus).

Table 432 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Storage Virtualizer System OperationalStatus)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Name		Mandatory	This shall be 'SNIA:Storage Virtualizer:SystemOperationalStatus'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Query		Mandatory	SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_ComputerSystem AND SourceInstance.CIM_ComputerSystem::OperationalStatus <> PreviousInstance.CIM_ComputerSystem::OperationalStatus.

Table 432 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Storage Virtualizer System OperationalStatus)

Properties	Flags	Requirement	Description & Notes
QueryLanguage		Mandatory	This shall be 'DMTF:CQL'.
ElementName	N	Optional	See the ElementName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).

22.8.19 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 433 describes class CIM_IndicationFilter (WQL Storage Virtualizer FCPort OperationalStatus).

Table 433 - SMI Referenced Properties/Methods for CIM_IndicationFilter (WQL Storage Virtualizer FCPort OperationalStatus)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Name		Mandatory	This shall be 'SNIA:Storage Virtualizer:FCPortOperationalStatusWQL'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).

Table 433 - SMI Referenced Properties/Methods for CIM_IndicationFilter (WQL Storage Virtualizer FCPort OperationalStatus)

Properties	Flags	Requirement	Description & 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 section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).

22.8.20 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 434 describes class CIM_IndicationFilter (WQL Storage Virtualizer Storage Volume OperationalStatus).

Table 434 - SMI Referenced Properties/Methods for CIM_IndicationFilter (WQL Storage Virtualizer Storage Volume OperationalStatus)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Name		Mandatory	This shall be 'SNIA:Storage Virtualizer:StorageVolumeOperationalStatusWQL'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).

Table 434 - SMI Referenced Properties/Methods for CIM_IndicationFilter (WQL Storage Virtualizer Storage Volume OperationalStatus)

Properties	Flags	Requirement	Description & Notes
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
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 section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).

22.8.21 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 435 describes class CIM_IndicationFilter (WQL Storage Virtualizer System OperationalStatus).

Table 435 - SMI Referenced Properties/Methods for CIM_IndicationFilter (WQL Storage Virtualizer System OperationalStatus)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Name		Mandatory	This shall be 'SNIA:Storage Virtualizer:SystemOperationalStatusWQL'.

Table 435 - SMI Referenced Properties/Methods for CIM_IndicationFilter (WQL Storage Virtualizer System OperationalStatus)

Properties	Flags	Requirement	Description & Notes
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
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 section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).

22.8.22 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 436 describes class CIM_LogicalIdentity (Shadow Storage Volume).

Table 436 - SMI Referenced Properties/Methods for CIM_LogicalIdentity (Shadow Storage Volume)

Properties	Flags	Requirement	Description & 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.

22.8.23 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 437 describes class CIM_MemberOfCollection (Allocated Resources).

Table 437 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Allocated Resources)

Properties	Flags	Requirement	Description & Notes
Member		Mandatory	A shadow storage volume (one with ExtentDiscriminator='SNIA:Shadow').
Collection		Mandatory	The AllocatedResources collection of shadow storage volumes.

22.8.24 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 438 describes class CIM_MemberOfCollection (Predefined Filter Collection to Storage Virtualizer Filters).

Table 438 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Predefined Filter Collection to Storage Virtualizer Filters)

Properties	Flags	Requirement	Description & Notes
Collection		Mandatory	Reference to the Storage Virtualizer predefined FilterCollection.
Member		Mandatory	Reference to the predefined IndicationFilters of the Storage Virtualizer.

22.8.25 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 439 describes class CIM_MemberOfCollection (Remote Resources).

Table 439 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Remote Resources)

Properties	Flags	Requirement	Description & Notes
Member		Mandatory	A shadow storage volume (one with ExtentDiscriminator='SNIA:Shadow').
Collection		Mandatory	The RemoteResources collection of shadow storage volumes.

22.8.26 CIM_MemberOfCollection (Storage Virtualizer ProfileSpecificLifecycleIndicationFilterCollection to Storage Virtualizer Filters)

Experimental. This associates the Storage Virtualizer ProfileSpecificLifecycleIndicationFilterCollection to the Filters defined by the Storage Virtualizer Profile.

Requirement: Optional

Table 440 describes class CIM_MemberOfCollection (Storage Virtualizer ProfileSpecificLifecycleIndicationFilterCollection to Storage Virtualizer Filters).

Table 440 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Storage Virtualizer ProfileSpecificLifecycleIndicationFilterCollection to Storage Virtualizer Filters)

Properties	Flags	Requirement	Description & Notes
Collection		Mandatory	Reference to the Storage Virtualizer ProfileSpecificLifecycleIndicationFilterCollection.
Member		Mandatory	Reference to the IndicationFilters defined by the Storage Virtualizer Profile.

22.8.27 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 441 describes class CIM_ProtocolControllerForUnit (Arbitrary LU for All LUNs View).

Table 441 - SMI Referenced Properties/Methods for CIM_ProtocolControllerForUnit (Arbitrary LU for All LUNs View)

Properties	Flags	Requirement	Description & Notes
DeviceNumber		Mandatory	Address (e.g. LUN) of the associated Device. Shall be formatted as unseparated uppercase hexadecimal digits, with no leading 0x.
DeviceAccess		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.

22.8.28 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 442 describes class CIM_ProtocolControllerForUnit (Storage volumes for All LUNs View).

Table 442 - SMI Referenced Properties/Methods for CIM_ProtocolControllerForUnit (Storage volumes for All LUNs View)

Properties	Flags	Requirement	Description & Notes
DeviceNumber		Mandatory	Address (e.g. LUN) of the associated Device. Shall be formatted as unseparated uppercase hexadecimal digits, with no leading 0x.
DeviceAccess		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).

22.8.29 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 443 describes class CIM_RemoteServiceAccessPoint (Shadow).

Table 443 - SMI Referenced Properties/Methods for CIM_RemoteServiceAccessPoint (Shadow)

Properties	Flags	Requirement	Description & 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.

22.8.30 CIM_SAPAvailableForElement

Experimental.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: This is required if CIM_RemoteServiceAccessPoint is modeled.

Table 444 describes class CIM_SAPAvailableForElement.

Table 444 - SMI Referenced Properties/Methods for CIM_SAPAvailableForElement

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	Shadow System.
AvailableSAP		Mandatory	The service access point of the shadow system.

22.8.31 CIM_SCSIArbitraryLogicalUnit (Arbitrary LU)

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

Table 445 describes class CIM_SCSIArbitraryLogicalUnit (Arbitrary LU).

Table 445 - SMI Referenced Properties/Methods for CIM_SCSIArbitraryLogicalUnit (Arbitrary LU)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	

Table 445 - SMI Referenced Properties/Methods for CIM_SCSIArbitraryLogicalUnit (Arbitrary LU)

Properties	Flags	Requirement	Description & Notes
CreationClassName		Mandatory	
DeviceID		Mandatory	Opaque identifer.
ElementName		Mandatory	User-friendly name.
Name		Mandatory	
OperationalStatus		Mandatory	

22.8.32 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 446 describes class CIM_SCSIProtocolController (All LUNs View).

Table 446 - SMI Referenced Properties/Methods for CIM_SCSIProtocolController (All LUNs View)

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

22.8.33 CIM_StorageExtent (Imported Extents)

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

Table 447 describes class CIM_StorageExtent (Imported Extents).

Table 447 - SMI Referenced Properties/Methods for CIM_StorageExtent (Imported Extents)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	

Table 447 - SMI Referenced Properties/Methods for CIM_StorageExtent (Imported Extents)

Properties	Flags	Requirement	Description & Notes
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'.

22.8.34 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 448 describes class CIM_StorageVolume (Shadow).

Table 448 - SMI Referenced Properties/Methods for CIM_StorageVolume (Shadow)

Properties	Flags	Requirement	Description & 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.

Table 448 - SMI Referenced Properties/Methods for CIM_StorageVolume (Shadow)

Properties	Flags	Requirement	Description & Notes
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'.
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.

Table 448 - SMI Referenced Properties/Methods for CIM_StorageVolume (Shadow)

Properties	Flags	Requirement	Description & Notes
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.

22.8.35 CIM_SystemDevice (Shadow StorageVolumes)

Experimental.

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

Table 449 describes class CIM_SystemDevice (Shadow StorageVolumes).

Table 449 - SMI Referenced Properties/Methods for CIM_SystemDevice (Shadow StorageVolumes)

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	The Shadow Computer System that contains this StorageVolume.
PartComponent		Mandatory	The storage volume that is managed by a computer system.

22.8.36 CIM_SystemDevice (System to SCSIArbitraryLogicalUnit)

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Elements that are mandatory if SCSIArbitraryLogicalUnit is instantiated.

Table 450 describes class CIM_SystemDevice (System to SCSIArbitraryLogicalUnit).

Table 450 - SMI Referenced Properties/Methods for CIM_SystemDevice (System to SCSIArbitraryLogicalUnit)

Properties	Flags	Requirement	Description & Notes
PartComponent		Mandatory	
GroupComponent		Mandatory	

22.8.37 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 451 describes class CIM_SystemDevice (System to SCSIProtocolController).

Table 451 - SMI Referenced Properties/Methods for CIM_SystemDevice (System to SCSIProtocol-Controller)

Properties	Flags	Requirement	Description & Notes
PartComponent		Mandatory	
GroupComponent		Mandatory	

22.8.38 CIM_SystemDevice (System to StorageExtent)

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

Table 452 describes class CIM_SystemDevice (System to StorageExtent).

Table 452 - SMI Referenced Properties/Methods for CIM_SystemDevice (System to StorageEx-tent)

Properties	Flags	Requirement	Description & Notes
PartComponent		Mandatory	The imported StorageExtent.
GroupComponent		Mandatory	The scoping ComputerSystem.

22.8.39 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 453 describes class SNIA_AllocatedResources.

Table 453 - SMI Referenced Properties/Methods for SNIA_AllocatedResources

Properties	Flags	Requirement	Description & 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'.

22.8.40 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 454 describes class SNIA_RemoteResources.

Table 454 - SMI Referenced Properties/Methods for SNIA_RemoteResources

Properties	Flags	Requirement	Description & 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

Clause 23: Volume Composition Profile

23.1 Description

23.1.1 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 this version of the specification. 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 version of the specification, 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.

23.1.2 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.

23.1.3 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.

23.1.4 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 99: "Volume Composition Class Mode".

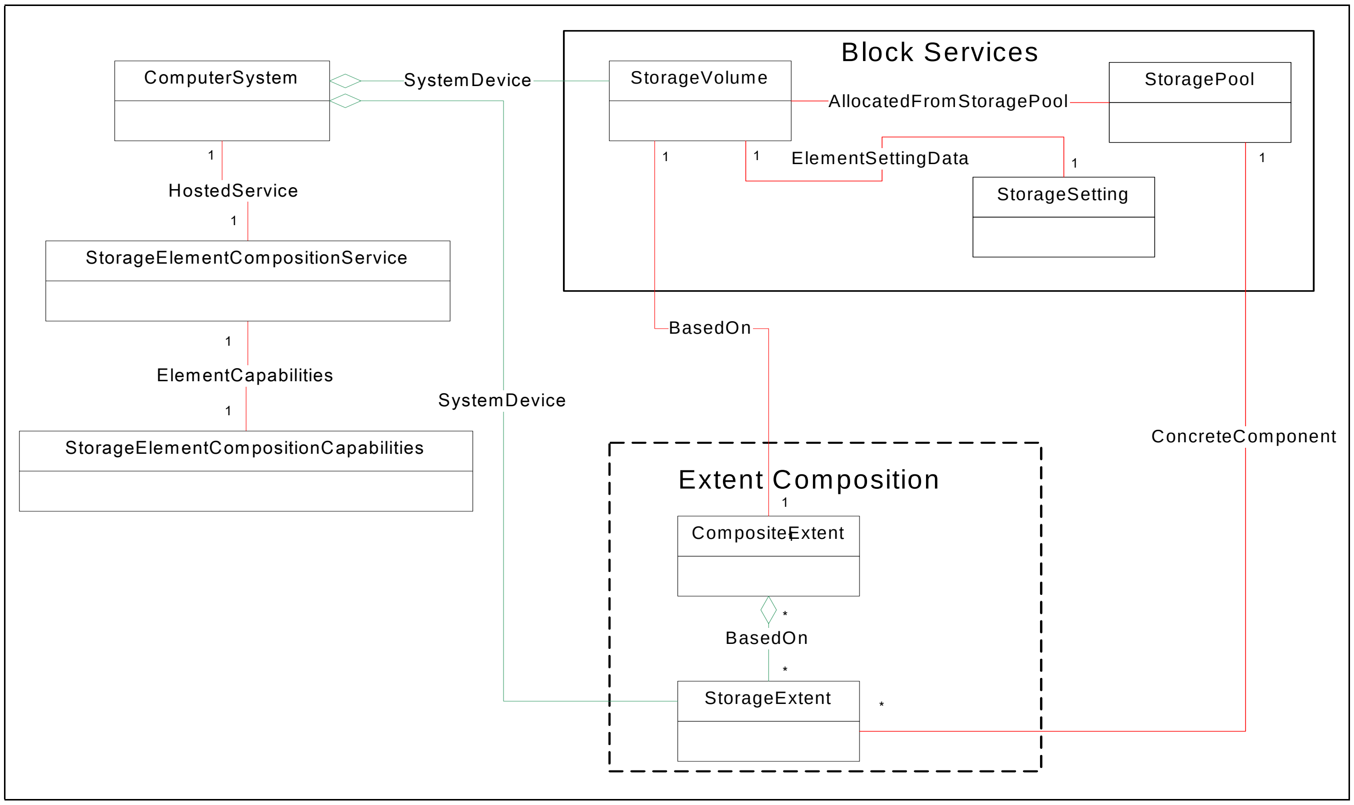

Figure 99 - Volume Composition Class Mode

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 version of the specification, 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 455 for a summary of those possible values.

Table 455 - 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
CompositeElementsMustBeSame Size	All elements used to create/modify a composite shall be the same size
CompositeElementsMustBeSame RAID/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

23.1.5 Quality of Service (QoS) Considerations

It is a requirement of Block Services that each StorageVolumes 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).

23.1.6 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.

23.1.7 Examples

23.1.7.1 Example 1

Figure 100 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.

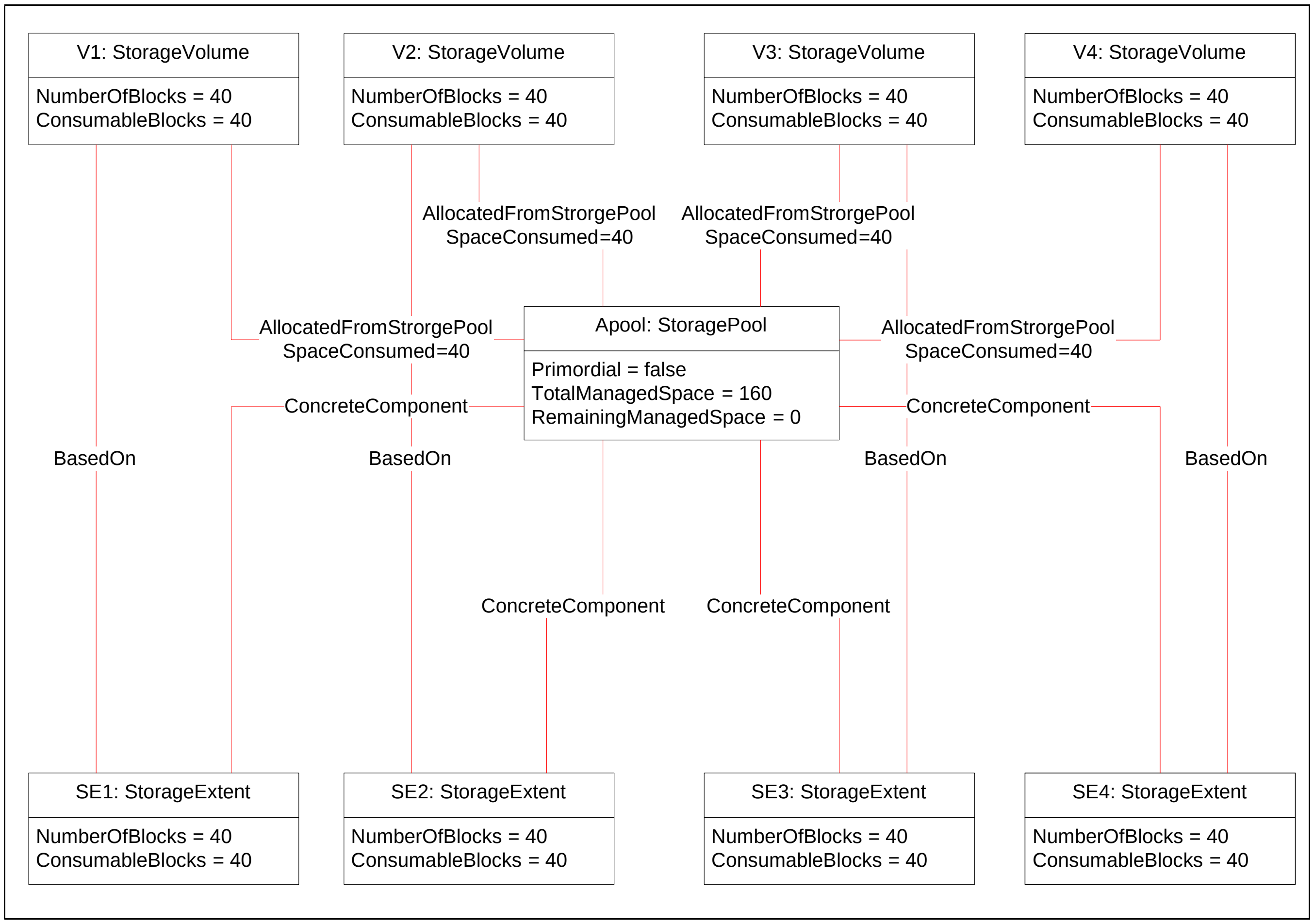

Figure 100 - Example 1 Step 1

Next, a composite volume is created by calling CreateOrModifyCompositeElement using three of the volumes (V1, V2, and V3). The result, shown in Figure 101, 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.

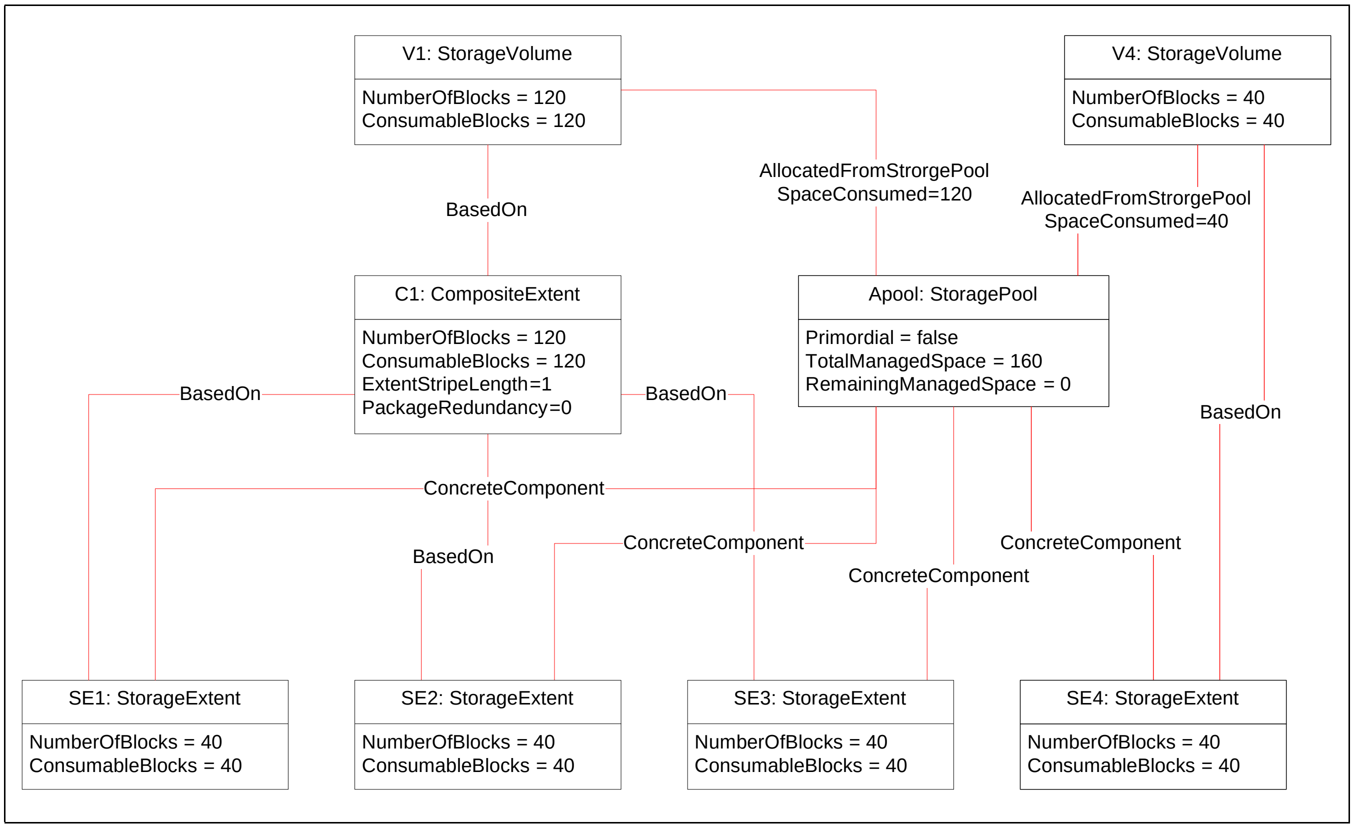

Figure 101 - Example 1 Step 2

23.1.7.2 First Alternative to Example 1

Figure 102 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

.

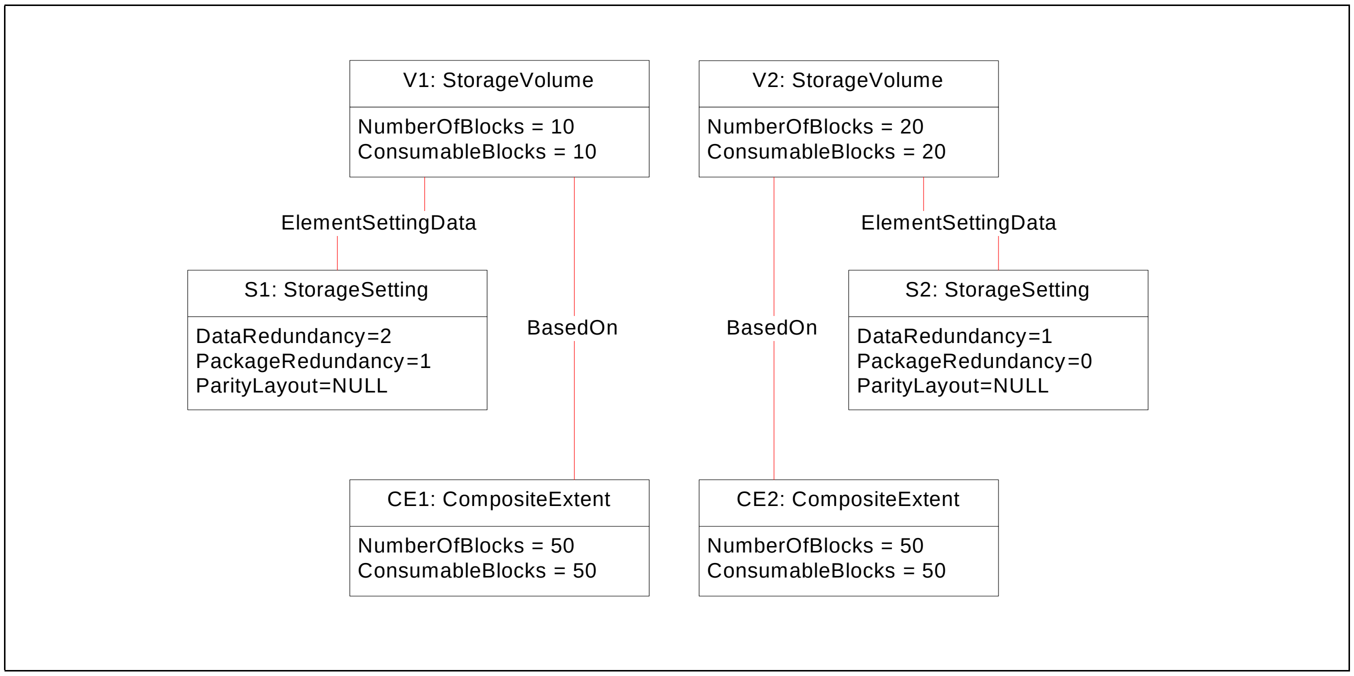

Figure 102 - First Alternative Example - Before Composition

As shown in Figure 103, 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.

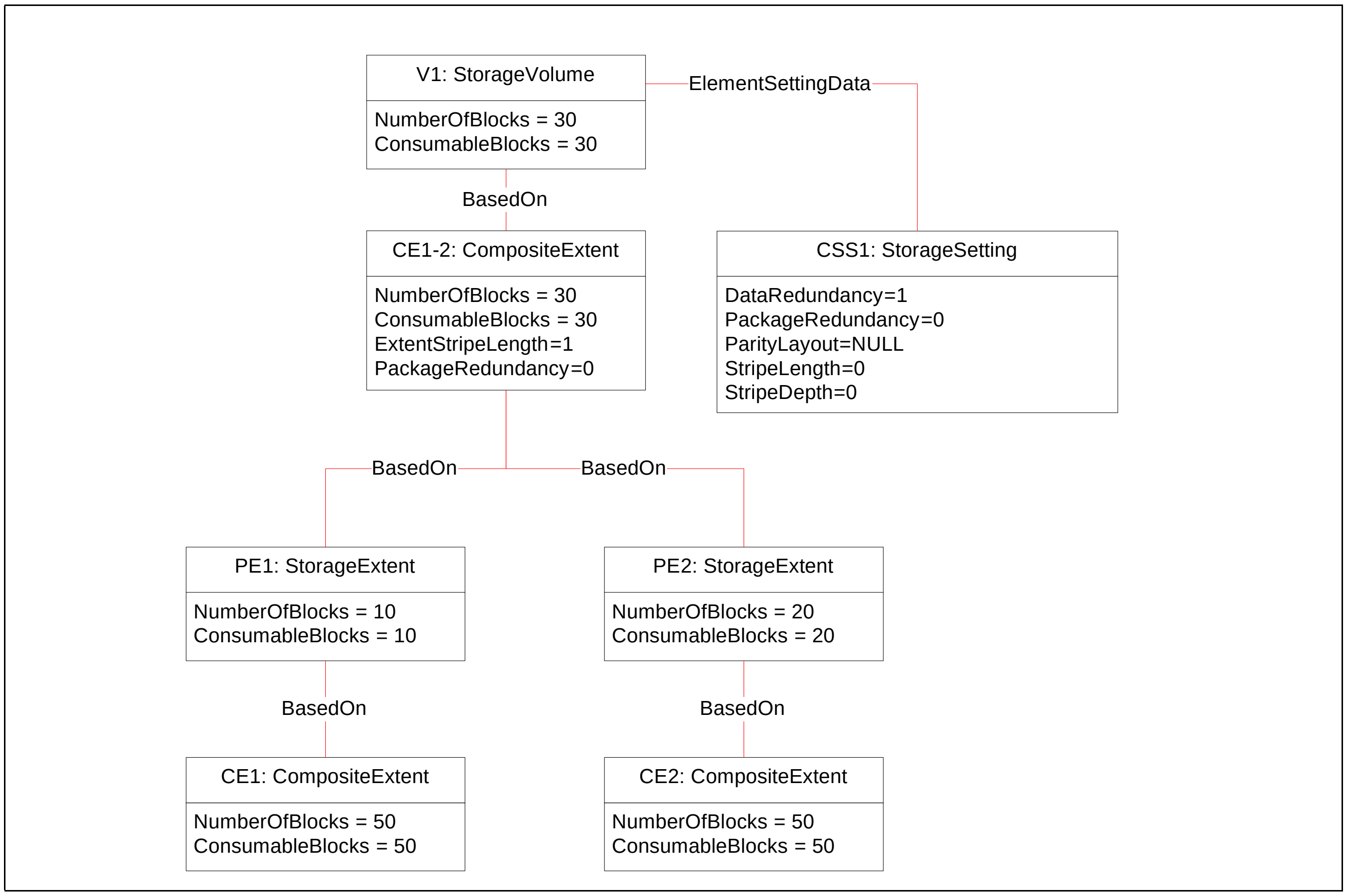

Figure 103 - First Alternative Example - After Composition

23.1.7.3 Second Alternative to Example 1

Figure 104 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.

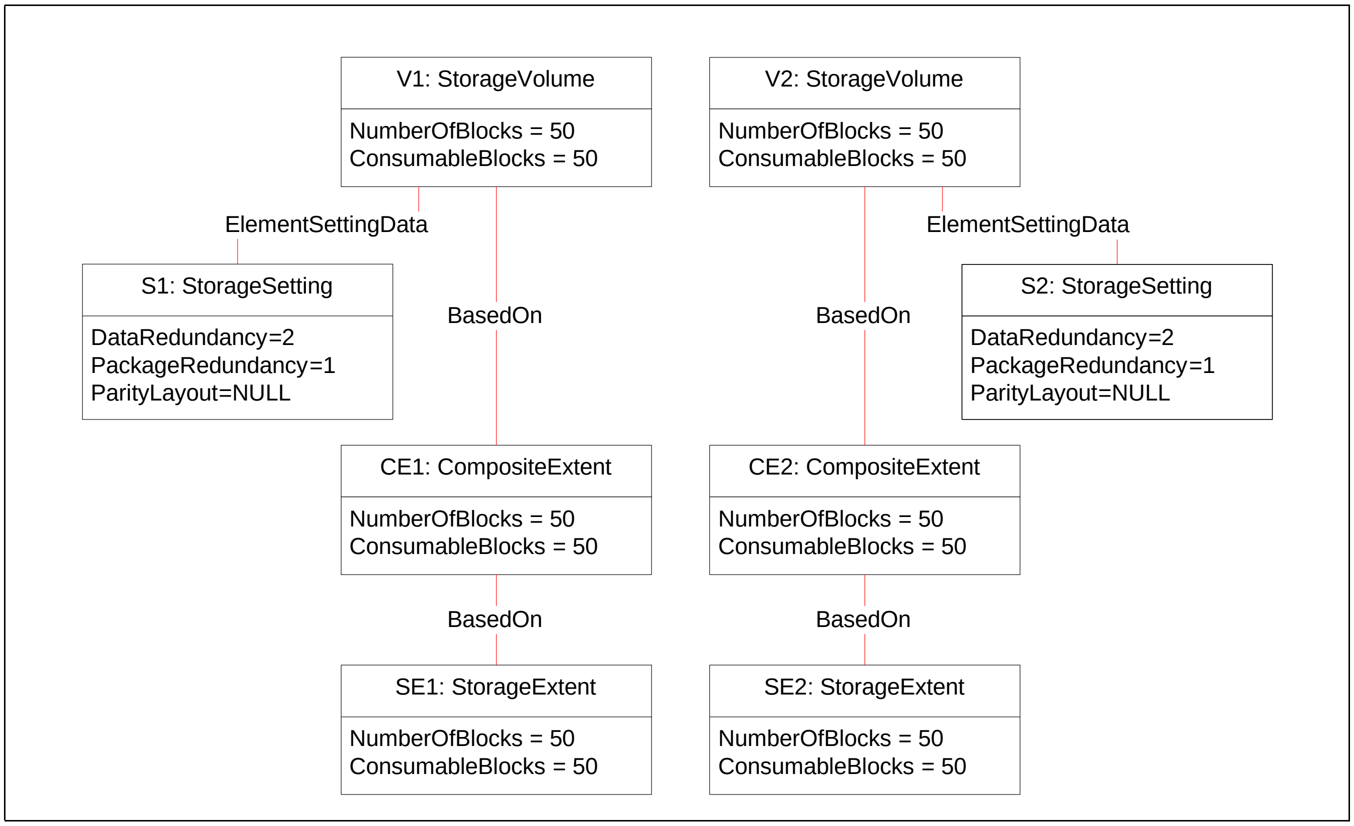

Figure 104 - Second Alternative Example - Before Composition

After composition, as shown in Figure 105, 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.

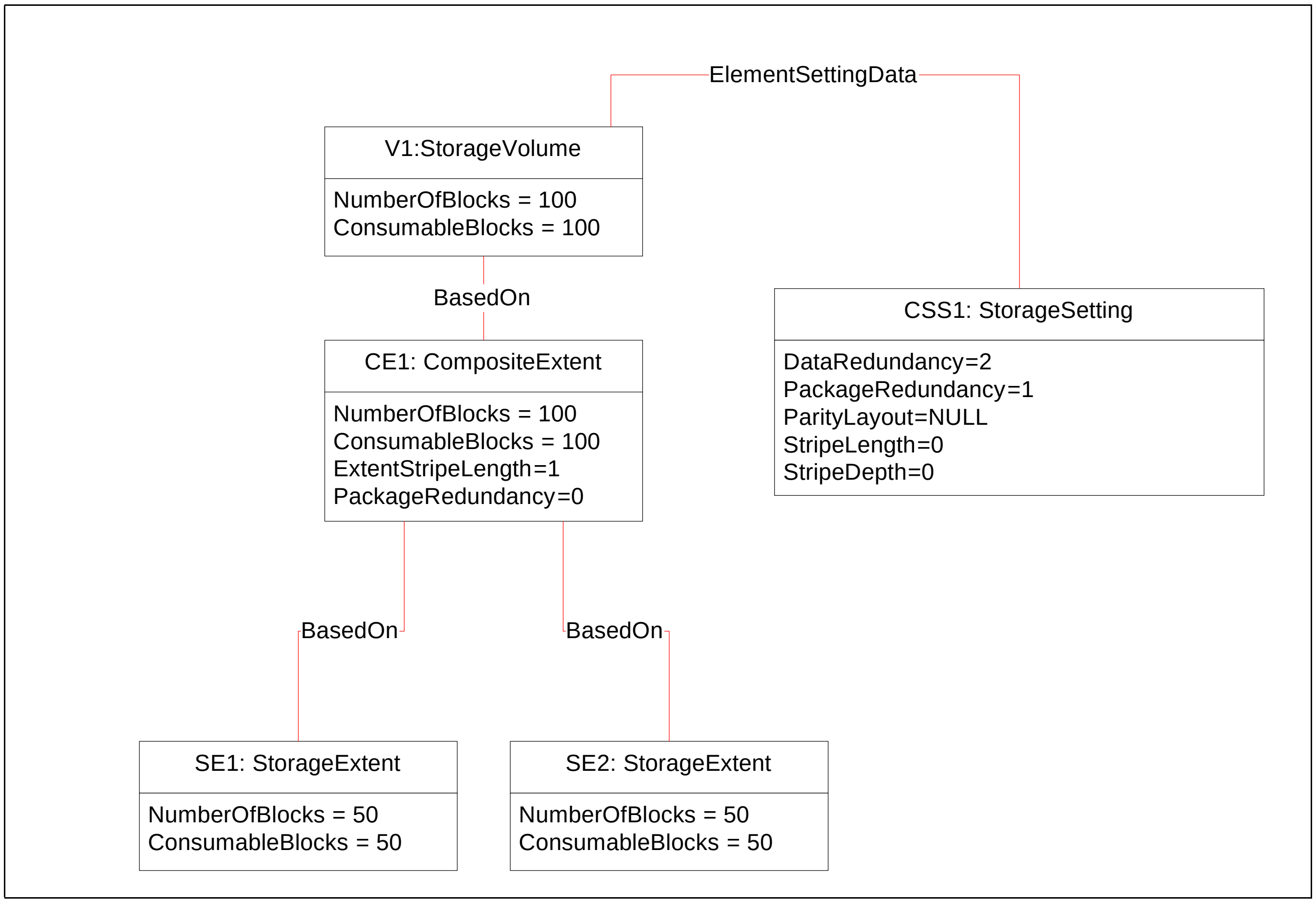

Figure 105 - Second Alternative Example - After Composition

23.1.7.4 Example 2

In this example, shown in Figure 106, 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.

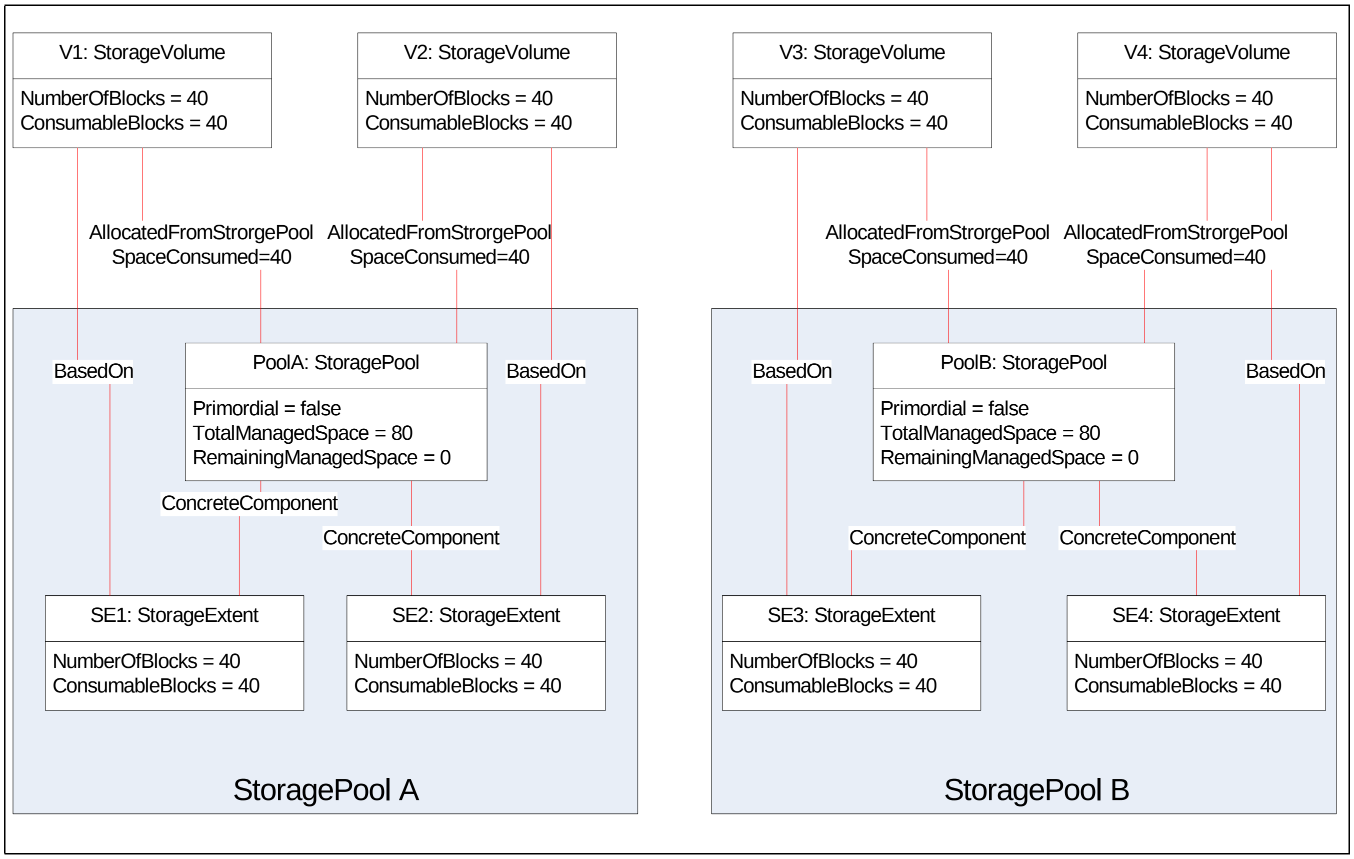

Figure 106 - Example 2 - Before Composition

Like the example shown in Figure 105, 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 107: "Example 2 - After Composition" shows the resulting model.

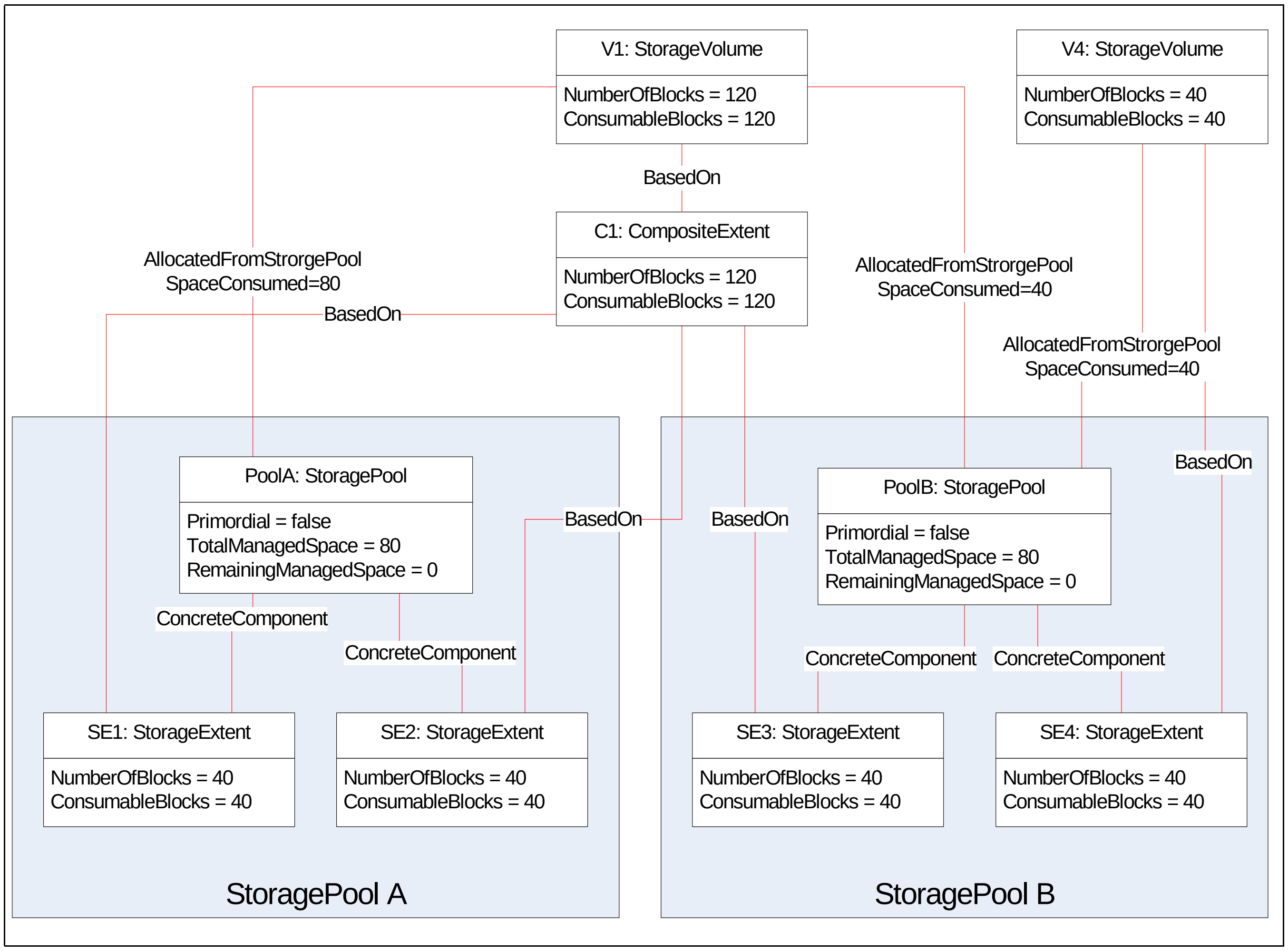

Figure 107 - Example 2 - After Composition

23.2 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 108 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 ore more volumes that are Striped. In this case, the composition is still considered “Concatenate+Stripe”.

Use the method 23.6.5 "GetCompositeElements" to determine which constituent elements are striped and which ones are concatenated.

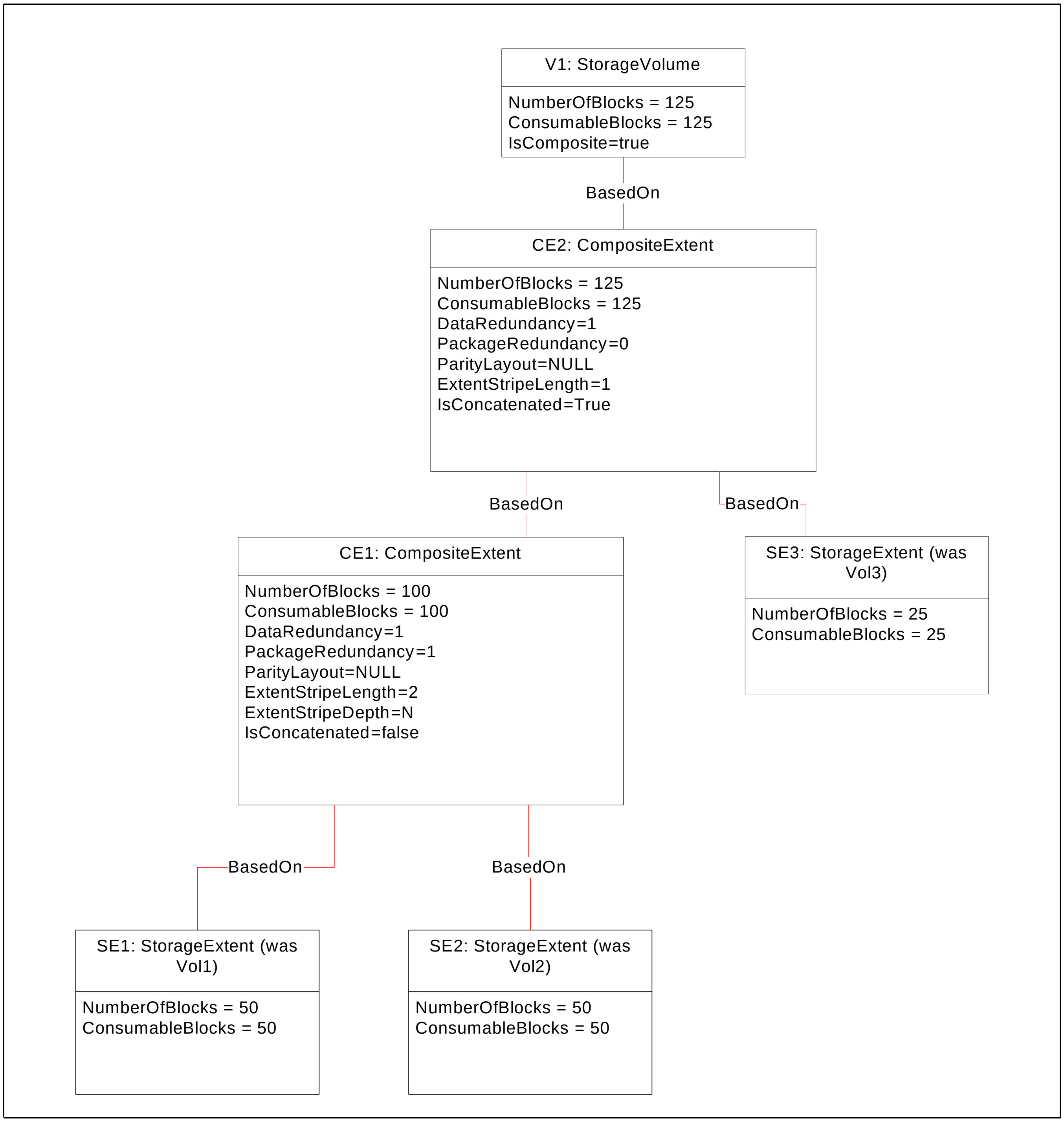

Figure 108 - Striping and Concatenation

23.3 Health and Fault Management Consideration

Not defined in this version of the specification.

23.4 Cascading Considerations

None

23.5 Supported Profiles, Subprofiles, and Packages

Table 456 describes the supported profiles for Volume Composition.

Table 456 - Supported Profiles for Volume Composition

Profile Name	Organization	Version	Requirement	Description
Extent Composition	SNIA	1.6.0	Mandatory	
Block Services	SNIA	1.6.0	Mandatory	

23.6 Methods of the Profile

Table 457 describes the methods of the profile.

Table 457 - 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

23.6.1 CreateOrModifyCompositeElement

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 specification, 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 458. For this version of the specification, 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.

23.6.1.1 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.

23.6.1.2 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 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 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 458 describes the return values for the CreateOrModifyCompositeElement method.

Table 458 - 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

Table 458 - CreateOrModifyCompositeElement

Method: CreateOrModifyCompositeElement			
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.
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

Table 458 - CreateOrModifyCompositeElement

Method: CreateOrModifyCompositeElement			
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.

23.6.2 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 455, “CompositionCharacteristics Property”). Removing all members is the same as calling ReturnElementToElements.

Table 459 describes the return values for the RemoveElementsFromElement method.

Table 459 - 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

Table 459 - RemoveElementsFromElement

Method: RemoveElementsFromElement			
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.

23.6.3 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 460 describes the return values for the ReturnElementToElements method.

Table 460 - 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		
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

23.6.4 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 version of the specification, only StorageVolumes shall be supported as the ElementType.

Table 461 describes the return values for the GetAvailableElements method.

Table 461 - 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
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

Table 461 - GetAvailableElements

Method: **GetAvailableElements**			
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.

23.6.5 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 462 describes the return values for the GetCompositeElements method.

Table 462 - 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 462 - GetCompositeElements

Method: GetCompositeElements			
OUT	OutElementTypes[]	uint16	A parallel array to OutElements array. Possible values: Member of Stripe Set, and Member of Concatenation

23.6.6 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 463 describes the return values for the GetSupportedStripeLengths method.

Table 463 - GetSupportedStripeLengths

Method: GetSupportedStripeLengths			
Return Values			
Value	Description		
0	Method completed with no error.		
1	Method not supported		
2	ElementType not supported		
3	Use GetSupportedStripeLengthRange instead		
Parameters			
Qualifiers	Name	Type	Description/Values
IN	ElementType	uint16	Type of element
OUT	StripeLengths[]	uint64	List of supported stripe Lengths

23.6.7 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 464 describes the return values for the GetSupportedStripeLengthRange method.

Table 464 - GetSupportedStripeLengthRange

Method: GetSupportedStripeLengthRange
Return Values

Table 464 - GetSupportedStripeLengthRange

Method: GetSupportedStripeLengthRange			
Value	Description		
0	Method completed with no error.		
1	Method not supported		
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 must be a multiple of this value

23.6.8 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 465 describes the return values for the GetSupportedStripeDepths method.

Table 465 - 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

23.6.9 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 466 describes the return values for the GetSupportedStripeDepthRange method.

Table 466 - 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 must be a multiple of this value

23.7 Client Considerations and Recipes

23.7.1 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 not 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.

23.7.2 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 accesible
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”>
      }

    }
}
```

23.7.3 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
}
```

23.8 Registered Name and Version

Volume Composition version 1.5.0 (Component Profile)

CIM Schema Version: 2.23.0

23.9 CIM Elements

Table 467 describes the CIM elements for Volume Composition.

Table 467 - CIM Elements for Volume Composition

Element Name	Requirement	Description
23.9.1 CIM_CompositeExtent	Mandatory	
23.9.2 CIM_CompositeExtentBasedOn (Volume Composition)	Mandatory	
23.9.3 CIM_ElementCapabilities	Mandatory	
23.9.4 CIM_ElementSettingData	Mandatory	
23.9.5 CIM_HostedService (Associates ComputerSystem and the ElementCompositionService)	Mandatory	
23.9.6 CIM_StorageElementCompositionCapabilities	Mandatory	
23.9.7 CIM_StorageElementCompositionService	Mandatory	
23.9.8 CIM_StorageSetting	Mandatory	
23.9.9 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.

23.9.1 CIM_CompositeExtent

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

Table 468 describes class CIM_CompositeExtent.

Table 468 - SMI Referenced Properties/Methods for CIM_CompositeExtent

Properties	Flags	Requirement	Description & 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	

Table 468 - SMI Referenced Properties/Methods for CIM_CompositeExtent

Properties	Flags	Requirement	Description & Notes
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 identifer for the Service.

23.9.2 CIM_CompositeExtentBasedOn (Volume Composition)

Created By: Extrinsics: CreateOrModifyCompositeElement, ReturnElementToElements
Modified By: External
Deleted By: Extrinsics: CreateOrModifyCompositeElement, ReturnElementToElements
Requirement: Mandatory

Table 469 describes class CIM_CompositeExtentBasedOn (Volume Composition).

Table 469 - SMI Referenced Properties/Methods for CIM_CompositeExtentBasedOn (Volume Composition)

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

23.9.3 CIM_ElementCapabilities

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

Table 470 describes class CIM_ElementCapabilities.

Table 470 - SMI Referenced Properties/Methods for CIM_ElementCapabilities

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	
Capabilities		Mandatory	

23.9.4 CIM_ElementSettingData

Created By: Extrinsics: CreateOrModifyCompositeElement, ReturnElementToElements
Modified By: Static
Deleted By: Extrinsics: CreateOrModifyCompositeElement, ReturnElementToElements
Requirement: Mandatory

Table 471 describes class CIM_ElementSettingData.

Table 471 - SMI Referenced Properties/Methods for CIM_ElementSettingData

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	StorageVolume or LogicalDisk.
SettingData		Mandatory	The composite setting data object associated with the composite element.

23.9.5 CIM_HostedService (Associates ComputerSystem and the ElementCompositionService)

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

Table 472 describes class CIM_HostedService (Associates ComputerSystem and the ElementCompositionService).

Table 472 - SMI Referenced Properties/Methods for CIM_HostedService (Associates Computer-System and the ElementCompositionService)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

23.9.6 CIM_StorageElementCompositionCapabilities

Created By: Static
Requirement: Mandatory

Table 473 describes class CIM_StorageElementCompositionCapabilities.

Table 473 - SMI Referenced Properties/Methods for CIM_StorageElementCompositionCapabilities

Properties	Flags	Requirement	Description & 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.

23.9.7 CIM_StorageElementCompositionService

Created By: Static
Requirement: Mandatory

Table 474 describes class CIM_StorageElementCompositionService.

Table 474 - SMI Referenced Properties/Methods for CIM_StorageElementCompositionService

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	The scoping System CreationClassName.
SystemName		Mandatory	The scoping System Name.

Table 474 - SMI Referenced Properties/Methods for CIM_StorageElementCompositionService

Properties	Flags	Requirement	Description & Notes
CreationClassName		Mandatory	The name of the concrete subclass.
Name		Mandatory	Unique identifer 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.
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 (a.k.a. stripe size) to use in the CreateOrModifyCompositeElement method.
GetSupportedCompositeStripeDepthRange()		Optional	This method returns the range of possible stripe depths (a.k.a. stripe size) to use in the CreateOrModifyCompositeElement method.

23.9.8 CIM_StorageSetting

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

Table 475 describes class CIM_StorageSetting.

Table 475 - SMI Referenced Properties/Methods for CIM_StorageSetting

Properties	Flags	Requirement	Description & 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	Number of underlying StorageVolumes 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 (aka stripe size).

Table 475 - SMI Referenced Properties/Methods for CIM_StorageSetting

Properties	Flags	Requirement	Description & Notes
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.

23.9.9 CIM_StorageVolume

Created By: Extrinsic: ReturnElementToElements
Modified By: External
Deleted By: Extrinsics: CreateOrModifyCompositeElement, ReturnElementToElements
Requirement: Storage Volumes used as storage elements.

Table 476 describes class CIM_StorageVolume.

Table 476 - SMI Referenced Properties/Methods for CIM_StorageVolume

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

EXPERIMENTAL

DEPRECATED

Clause 24: Volume Management Profile

Note: The Volume Management Profile is scheduled for removal for SMI-S 2.0. The functionality of this profile will not be replaced in SMI-S 2.0. The Storage Network Industry Association (SNIA) is not aware of any implementations of this profile. The SNIA would like to hear from anyone that has implemented the Volume Management Profile. If your company or organization has implemented this profile and is a member of the SNIA, please contact the DRM Technical Working Group or indicate your preference to keep this profile in SMI-S 2.0 during member reviews and ballots. If your company or organization has implemented this profile and is not a member of the SNIA, please indicate your preference to keep this profile as part of SMI-S using the SNIA feedback portal: http://www.snia.org/tech_activities/feedback/ .

24.1 Description

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.

24.1.1 Instance Diagram

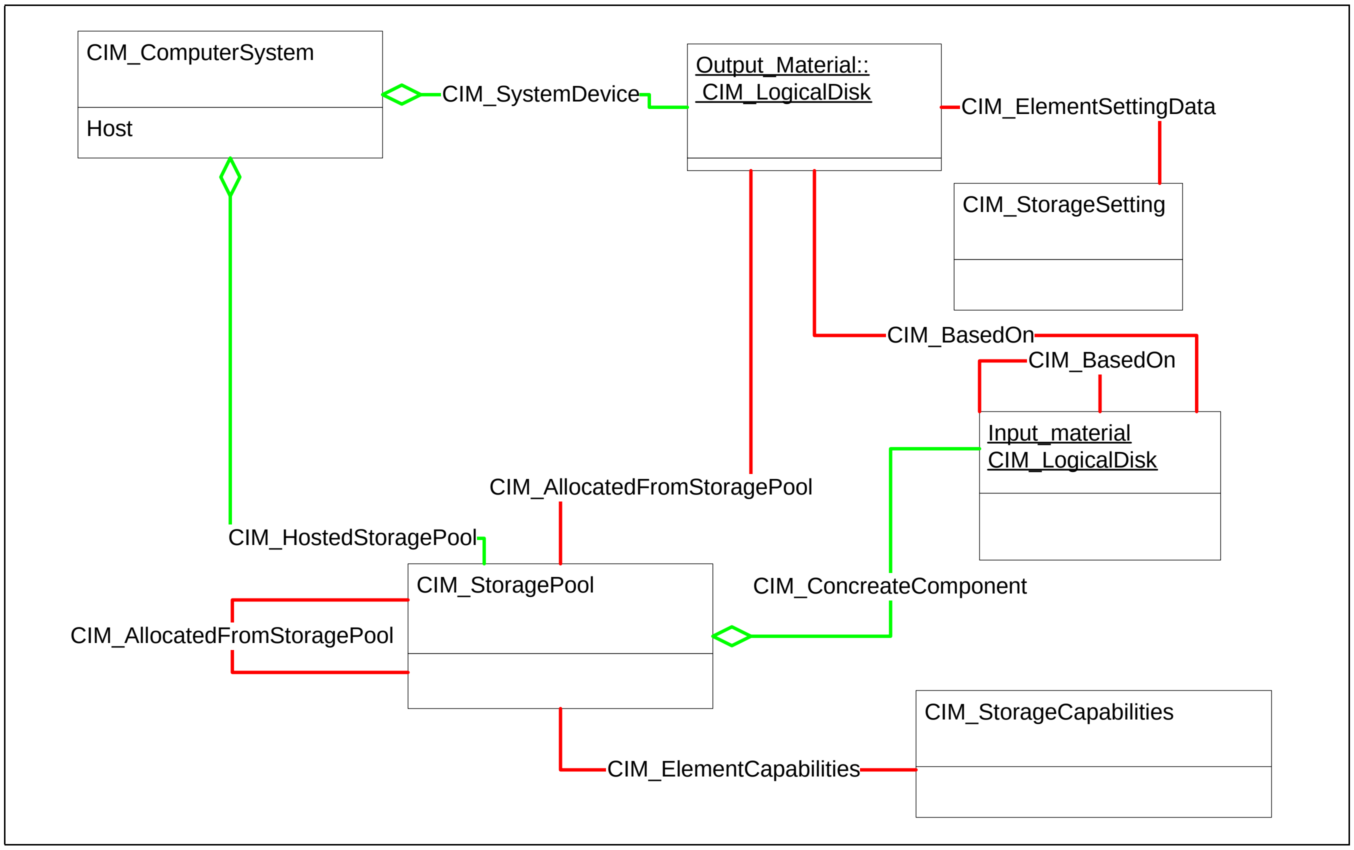

Figure 109 - Volume Management Instance Diagram

24.1.2 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.

24.1.3 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.

24.1.4 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.

24.1.5 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 5.1 Description for more details on methods for creating pools and logical volumes.

24.1.6 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 5: "Block Services Package" for a description of these settings. The StorageSettings.Description 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.

24.1.7 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.

24.2 Health and Fault Management Considerations

Not defined in this standard.

24.3 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 the description of the Cascading Subprofile for usage with the Security Resource Ownership Subprofile.

24.4 Supported Subprofiles and Packages

Table 477 describes the supported profiles for Volume Management.

Table 477 - Supported Profiles for Volume Management

Profile Name	Organization	Version	Requirement	Description
Access Points	SNIA	1.3.0	Optional	
Extent Composition	SNIA	1.6.0	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	

Table 477 - Supported Profiles for Volume Management

Profile Name	Organization	Version	Requirement	Description
Block Server Performance	SNIA	1.5.0	Optional	
Block Services	SNIA	1.6.0	Mandatory	
Health	SNIA	1.2.0	Mandatory	
Indication	SNIA	1.5.0	Mandatory	

24.5 Methods of the Profile

None

24.6 Client Considerations and Recipes

Use Clause 5: Block Services Package to create and modify volumes.

See recipes for creating volumes in 5.6.6 "Conditional: Create StoragePool and Storage Element on Block Server (e.g., Array or Volume Manager)" in Clause 5: 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.

24.6.1 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.

24.7 Registered Name and Version

Volume Management version 1.2.0 (Autonomous Profile)

24.8 CIM Elements

Table 478 describes the CIM elements for Volume Management.

Table 478 - CIM Elements for Volume Management

Element Name	Requirement	Description
24.8.1 CIM_AllocatedFromStoragePool (LogicalDisk from Pool)	Mandatory	
24.8.2 CIM_AllocatedFromStoragePool (Pool from Pool)	Mandatory	

Table 478 - CIM Elements for Volume Management

Element Name	Requirement	Description
24.8.3 CIM_ComputerSystem	Mandatory	Associated to RegisteredProfile.
24.8.4 CIM_ElementCapabilities	Mandatory	
24.8.5 CIM_ElementSettingData	Mandatory	
24.8.6 CIM_HostedStoragePool	Mandatory	
24.8.7 CIM_LogicalDisk	Mandatory	
24.8.8 CIM_StorageCapabilities	Mandatory	
24.8.9 CIM_StoragePool (Concrete)	Mandatory	Logical Disks are allocated from 'concrete' pools.
24.8.10 CIM_StoragePool (Primordial)	Mandatory	At least one primordial pool must exist for a host. This is the 'unallocated storage' of the host, and contains unused disks.
24.8.11 CIM_StorageSetting	Mandatory	
24.8.12 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.

24.8.1 CIM_AllocatedFromStoragePool (LogicalDisk from Pool)

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

Table 479 describes class CIM_AllocatedFromStoragePool (LogicalDisk from Pool).

Table 479 - SMI Referenced Properties/Methods for CIM_AllocatedFromStoragePool (LogicalDisk from Pool)

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

24.8.2 CIM_AllocatedFromStoragePool (Pool from Pool)

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

Table 480 describes class CIM_AllocatedFromStoragePool (Pool from Pool).

Table 480 - SMI Referenced Properties/Methods for CIM_AllocatedFromStoragePool (Pool from Pool)

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

24.8.3 CIM_ComputerSystem

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

Shall be associated to RegisteredProfile using ElementConformsToProfile association. The RegisteredProfile instance shall have RegisteredName set to 'Volume Management', RegisteredOrganization set to 'SNIA', and RegisteredVersion set to '1.2.0'.

Table 481 describes class CIM_ComputerSystem.

Table 481 - SMI Referenced Properties/Methods for CIM_ComputerSystem

Properties	Flags	Requirement	Description & 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.
Dedicated		Mandatory	This should include 0. This indicates that this computer system is not dedicated to volume management.
PrimaryOwnerContact		Optional	
PrimaryOwnerName		Optional	

24.8.4 CIM_ElementCapabilities

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

Table 482 describes class CIM_ElementCapabilities.

Table 482 - SMI Referenced Properties/Methods for CIM_ElementCapabilities

Properties	Flags	Requirement	Description & Notes
Capabilities		Mandatory	
ManagedElement		Mandatory	

24.8.5 CIM_ElementSettingData

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

Requirement: Mandatory

Table 483 describes class CIM_ElementSettingData.

Table 483 - SMI Referenced Properties/Methods for CIM_ElementSettingData

Properties	Flags	Requirement	Description & Notes
SettingData		Mandatory	
ManagedElement		Mandatory	

24.8.6 CIM_HostedStoragePool

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

Table 484 describes class CIM_HostedStoragePool.

Table 484 - SMI Referenced Properties/Methods for CIM_HostedStoragePool

Properties	Flags	Requirement	Description & Notes
PartComponent		Mandatory	
GroupComponent		Mandatory	

24.8.7 CIM_LogicalDisk

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

Table 485 describes class CIM_LogicalDisk.

Table 485 - SMI Referenced Properties/Methods for CIM_LogicalDisk

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	Opaque identifer.
ElementName		Mandatory	User friendly name.

Table 485 - SMI Referenced Properties/Methods for CIM_LogicalDisk

Properties	Flags	Requirement	Description & Notes
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	

24.8.8 CIM_StorageCapabilities

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

Table 486 describes class CIM_StorageCapabilities.

Table 486 - SMI Referenced Properties/Methods for CIM_StorageCapabilities

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
ElementName		Mandatory	
NoSinglePointOfFailure		Mandatory	
NoSinglePointOfFailureDefault		Mandatory	
DataRedundancyMin		Mandatory	
DataRedundancyMax		Mandatory	
DataRedundancyDefault		Mandatory	

Table 486 - SMI Referenced Properties/Methods for CIM_StorageCapabilities

Properties	Flags	Requirement	Description & Notes
PackageRedundancyMin		Mandatory	
PackageRedundancyMax		Mandatory	
PackageRedundancyDefault		Mandatory	
DeltaReservationDefault		Mandatory	
DeltaReservationMax		Mandatory	
DeltaReservationMin		Mandatory	

24.8.9 CIM_StoragePool (Concrete)

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

Table 487 describes class CIM_StoragePool (Concrete).

Table 487 - SMI Referenced Properties/Methods for CIM_StoragePool (Concrete)

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
ElementName		Mandatory	
PoolID		Mandatory	
TotalManagedSpace		Mandatory	
RemainingManagedSpace		Mandatory	
Primordial		Mandatory	Set to false.

24.8.10 CIM_StoragePool (Primordial)

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

Table 488 describes class CIM_StoragePool (Primordial).

Table 488 - SMI Referenced Properties/Methods for CIM_StoragePool (Primordial)

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
ElementName		Mandatory	
PoolID		Mandatory	
TotalManagedSpace		Mandatory	
RemainingManaged Space		Mandatory	
Primordial		Mandatory	Set to true.

24.8.11 CIM_StorageSetting

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

Table 489 describes class CIM_StorageSetting.

Table 489 - SMI Referenced Properties/Methods for CIM_StorageSetting

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	Opaque identifier.
ElementName		Mandatory	User friendly name.... can be used for 'potted' settings for specific RAID levels.
DataRedundancyMin		Mandatory	
DataRedundancyMax		Mandatory	
DataRedundancyGoa l		Mandatory	
PackageRedundancy Min		Mandatory	
PackageRedundancy Max		Mandatory	
PackageRedundancy Goal		Mandatory	
DeltaReservationGoa l		Mandatory	

Table 489 - SMI Referenced Properties/Methods for CIM_StorageSetting

Properties	Flags	Requirement	Description & Notes
DeltaReservationMax		Mandatory	
DeltaReservationMin		Mandatory	

24.8.12 CIM_SystemDevice

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

Table 490 describes class CIM_SystemDevice.

Table 490 - SMI Referenced Properties/Methods for CIM_SystemDevice

Properties	Flags	Requirement	Description & Notes
PartComponent		Mandatory	
GroupComponent		Mandatory	

DEPRECATED

EXPERIMENTAL

Clause 25: Storage Element Protection SubProfile

25.1 Description

25.1.1 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.

25.1.2 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.

25.1.3 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.

25.1.4 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 110: "Storage Element Protection Class Model". After the retention

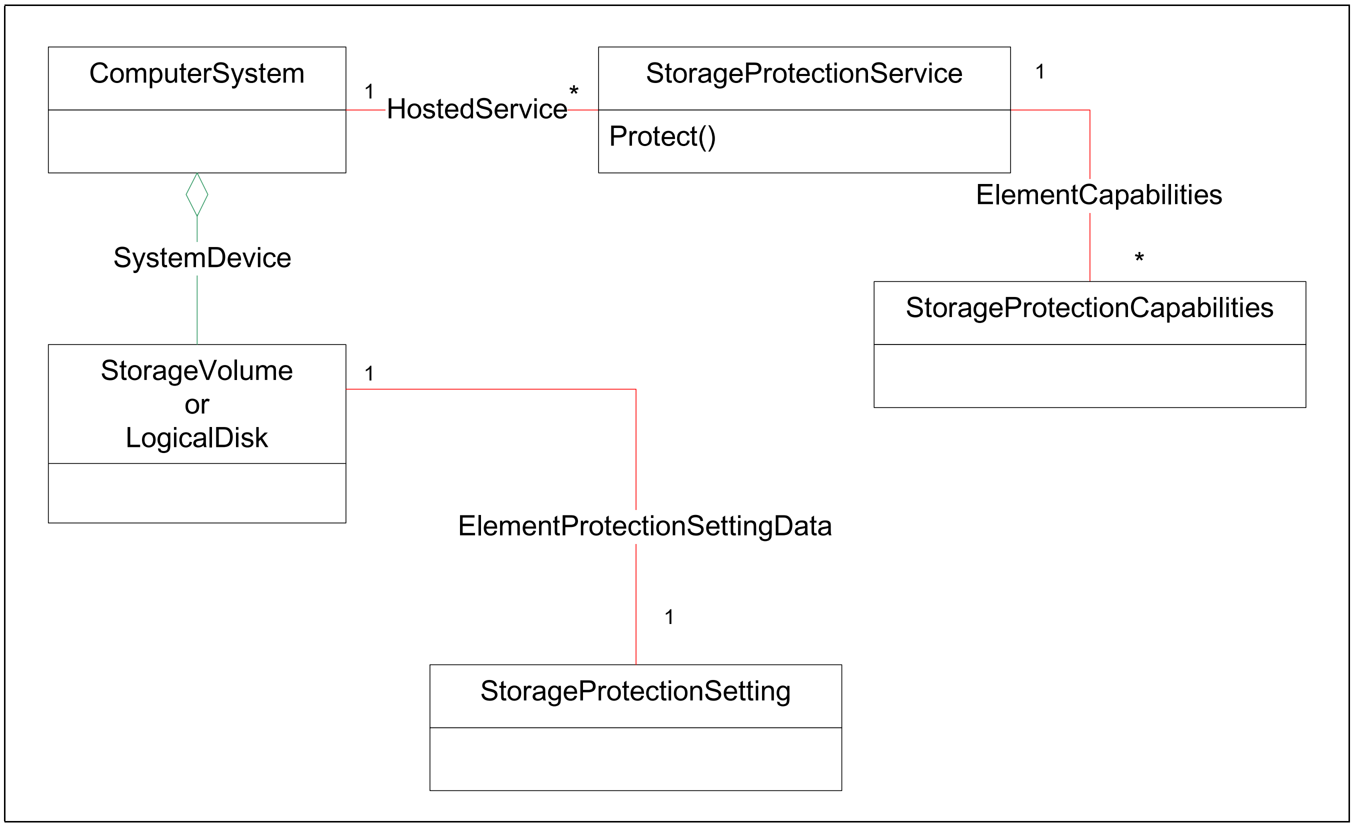

Figure 110 - Storage Element Protection Class Model

period has expired and every 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 491 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 491 - Properties for StorageProtectionCapabilities

Property	Flags	Type	Descriptions & 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

Table 491 - Properties for StorageProtectionCapabilities

Property	Flags	Type	Descriptions & Notes
SupportedSynchronousActions		uint16[]	Methods that will not create a job. Possible values: 1 - Storage Element Protection
SupportedAsynchronousAction s		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 492. 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 492 - Properties for StorageProtectionSetting

Property	Flags	Type	Descriptions & Notes
ProtectionControlled		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 has but currently the service has been withdrawn or not available to obtain protection attributes by some accident.
Access		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 Protect() to transition to "Write Once", it's still needed for correct reporting of status
InquiryProtection		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 StorageVolume and it is optional to implement
DenyAsCopyTarget		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

Table 492 - Properties for StorageProtectionSetting

Property	Flags	Type	Descriptions & Notes
LUNMappingConfigurable		boolean	Whether LU assignment to the storage element is configurable or not. This property is utilized in the protection of a StorageVolume and is optional to implement
ProtectExpirationSpecified		uint16	Duration type of the storage element protection. 1: None 2: Limited Expiration 3: Permanent
RemainingProtectionTime		datetime	Amount of remaining time before a management application can change the access permission.

25.1.5 Access permission

The overall state of the StorageVolume or LogicalDisk protection is indicated by the combination of several properties. Table 493, Table 494, Table 495, Table 496, and Table 497 show the possible values of each property listed in Table 492. These tables apply to properties in the StorageProtectionSetting class.

Table 493 - Values for ProtectionControlled

Value	Description
TRUE	Storage element is under protection control.
FALSE	Storage element is NOT under protection control.

Table 494 - 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 495 - 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 496 - 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 497 - Values for LUNMappingConfigurable

Value	Description
TRUE	LU assignment to the storage volume is configurable
FALSE	LU assignment to the storage volume is not configurable

25.1.6 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 498 and Table 499 show the meaning of each property value. These tables apply to properties in the StorageProtectionSetting class.

Table 498 - 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 499 - 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 111: "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.

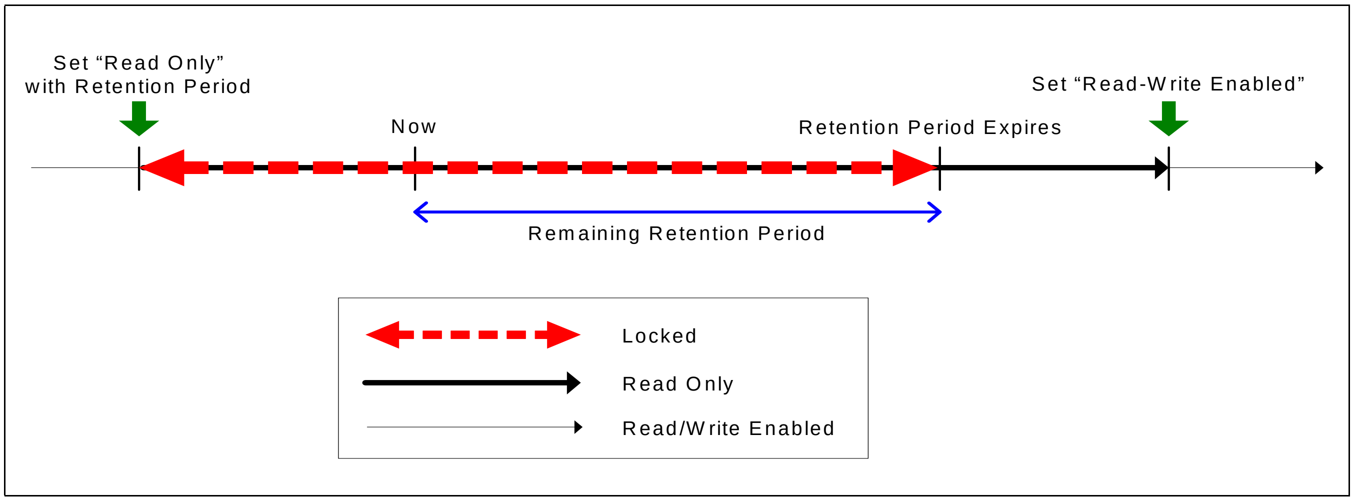

Figure 111 - Retention Time Line

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.

25.1.7 Protection State Transition

Figure 112: "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".

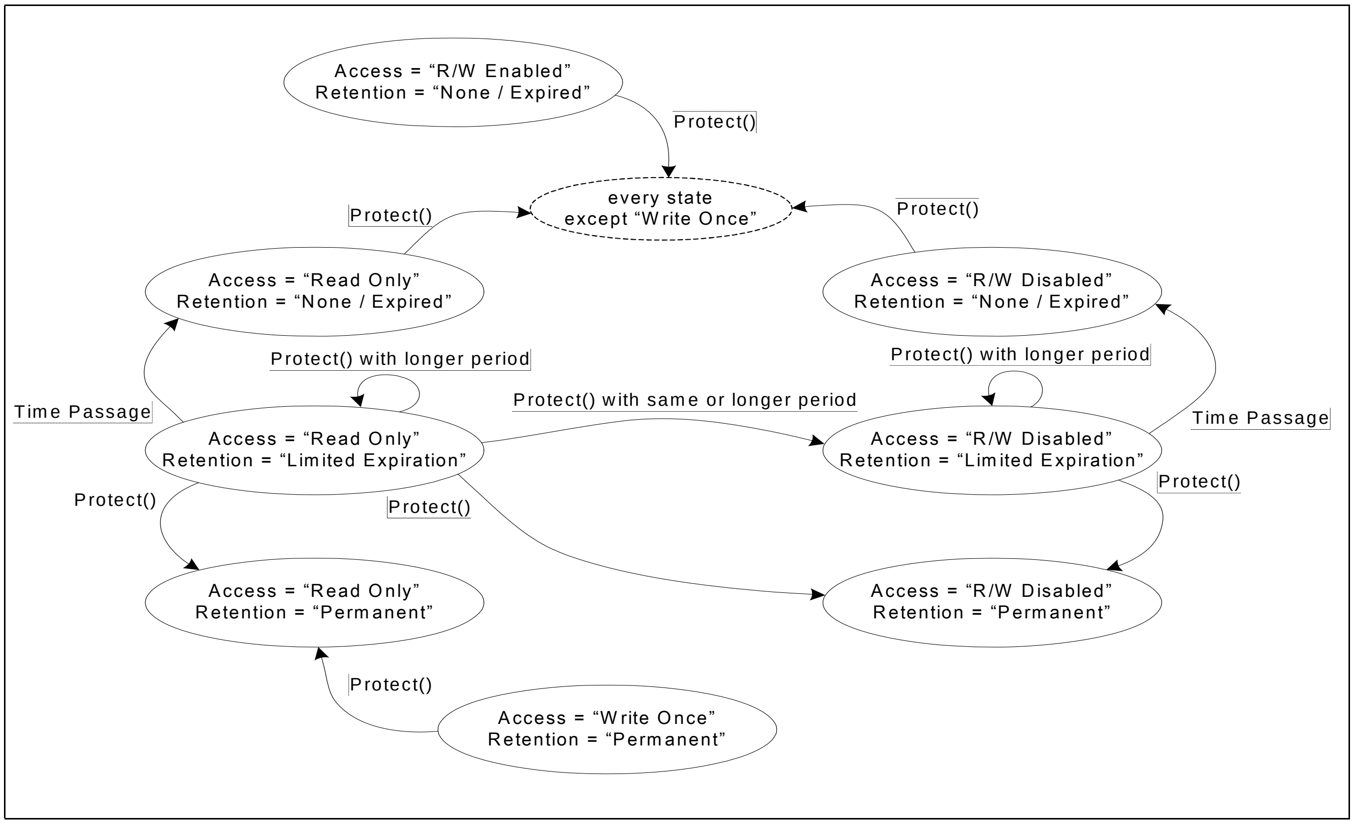

Figure 112 - Protection State Transition DIagram

25.1.8 Sample Usage Scenario

Figure 113: "Step 1 - Initial State", Figure 114: "Step 2 - Volume Set to Read-only", Figure 115: "Step 3 - Second Volume Set to Read-only", Figure 116: "Step 4 - Volume Set to Read/Write Disabled", and Figure 117: "Step 5 Volume Access Changed" show the progression of a typical usage scenario for StorageVolume protection.

25.1.8.1 Step 1: StorageVolume not protected

Figure 113: "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.

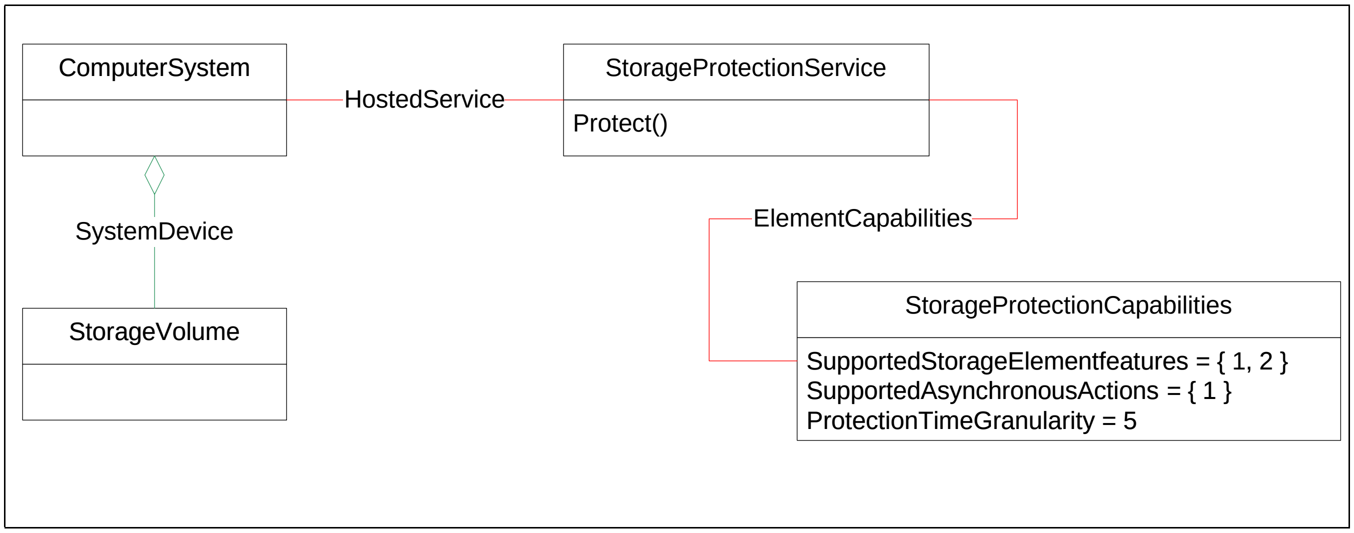
ComputerSystem
HostedService
StorageProtectionService
Protect()
SystemDevice
ElementCapabilities
StorageVolume
StorageProtectionCapabilities
SupportedStorageElementfeatures = { 1, 2 }
SupportedAsynchronousActions = { 1 }
ProtectionTimeGranularity = 5

Figure 113 - Step 1 - Initial State

25.1.8.2 Step 2: Volume Set to Read-only

In Figure 114: "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.

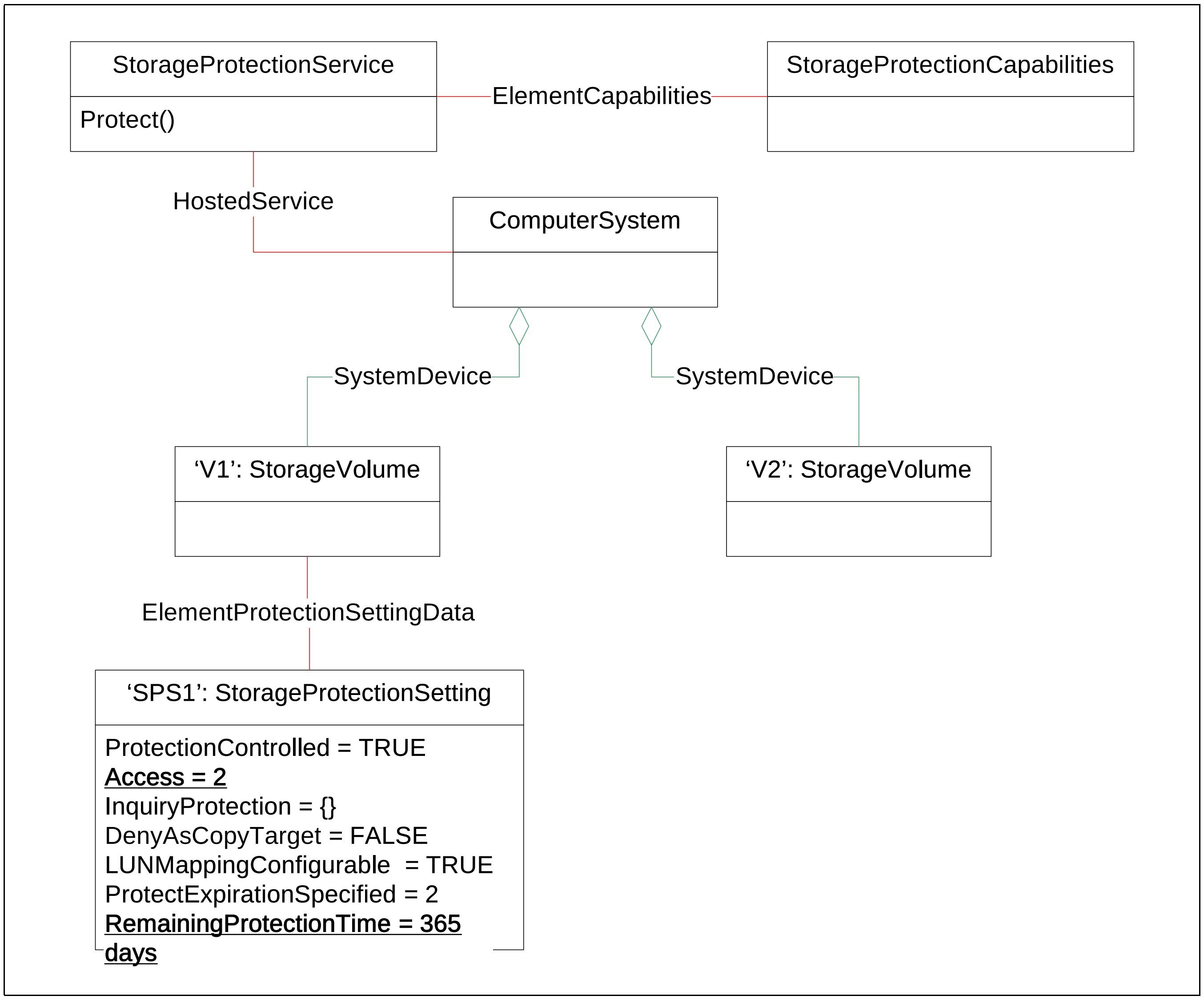
StorageProtectionService
Protect()
ElementCapabilities
StorageProtectionCapabilities
HostedService
ComputerSystem
SystemDevice
SystemDevice
'V1': StorageVolume
'V2': StorageVolume
ElementProtectionSettingData
'SPS1': StorageProtectionSetting
ProtectionControlled = TRUE
Access = 2
InquiryProtection = {}
DenyAsCopyTarget = FALSE
LUNMappingConfigurable = TRUE
ProtectExpirationSpecified = 2
RemainingProtectionTime = 365 days

Figure 114 - Step 2 - Volume Set to Read-only

25.1.8.3 Step 3: Second Volume Set to Read-only

Figure 115: "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.

.

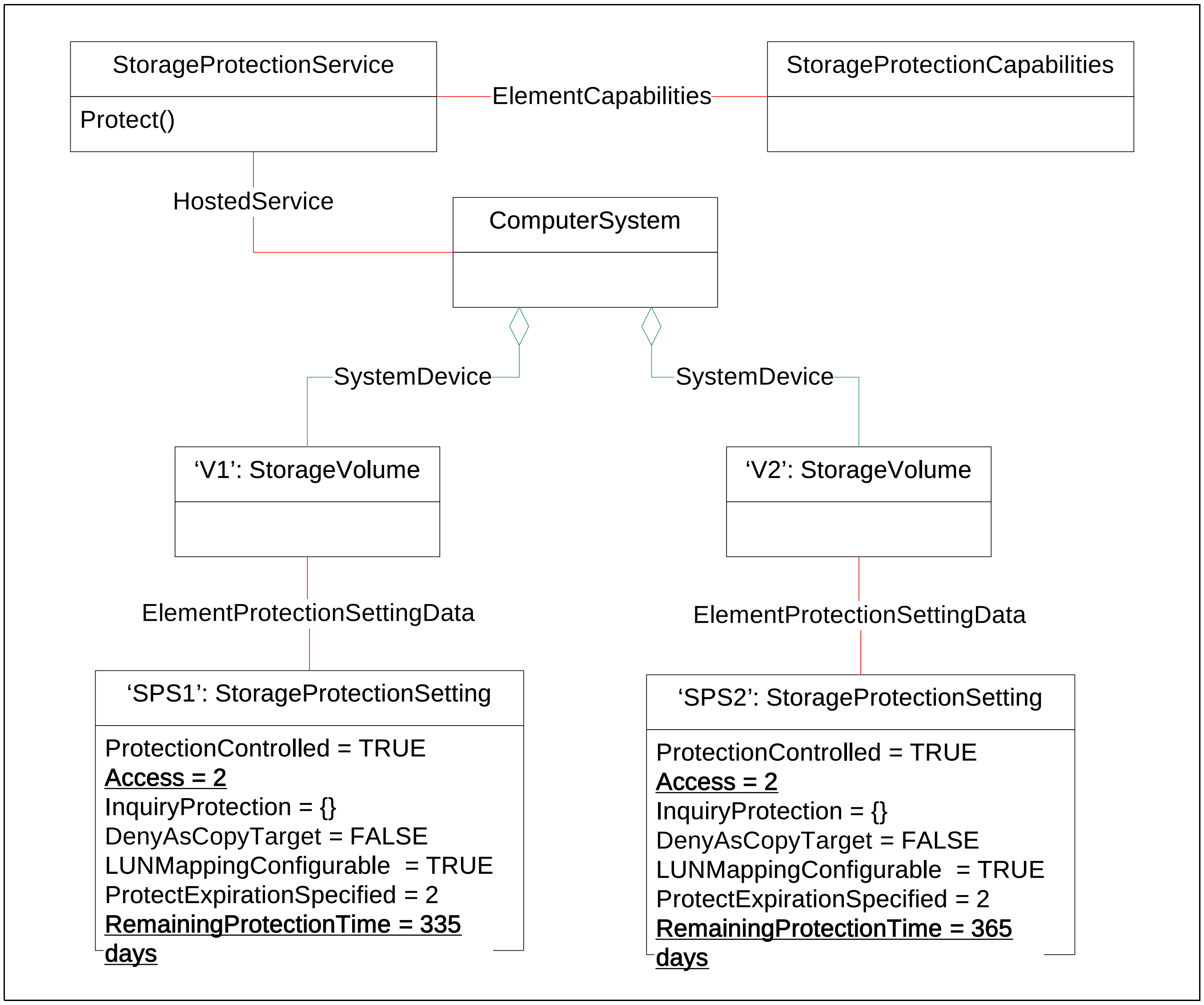
StorageProtectionService
Protect()
ElementCapabilities
StorageProtectionCapabilities
HostedService
ComputerSystem
SystemDevice
SystemDevice
'V1': StorageVolume
'V2': StorageVolume
ElementProtectionSettingData
ElementProtectionSettingData
'SPS1': StorageProtectionSetting
ProtectionControlled = TRUE
Access = 2
InquiryProtection = {}
DenyAsCopyTarget = FALSE
LUNMappingConfigurable = TRUE
ProtectExpirationSpecified = 2
RemainingProtectionTime = 335 days
'SPS2': StorageProtectionSetting
ProtectionControlled = TRUE
Access = 2
InquiryProtection = {}
DenyAsCopyTarget = FALSE
LUNMappingConfigurable = TRUE
ProtectExpirationSpecified = 2
RemainingProtectionTime = 365 days

Figure 115 - Step 3 - Second Volume Set to Read-only

25.1.8.4 Step 4: Volume Set to Read/Write Disabled

Figure 116: "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.

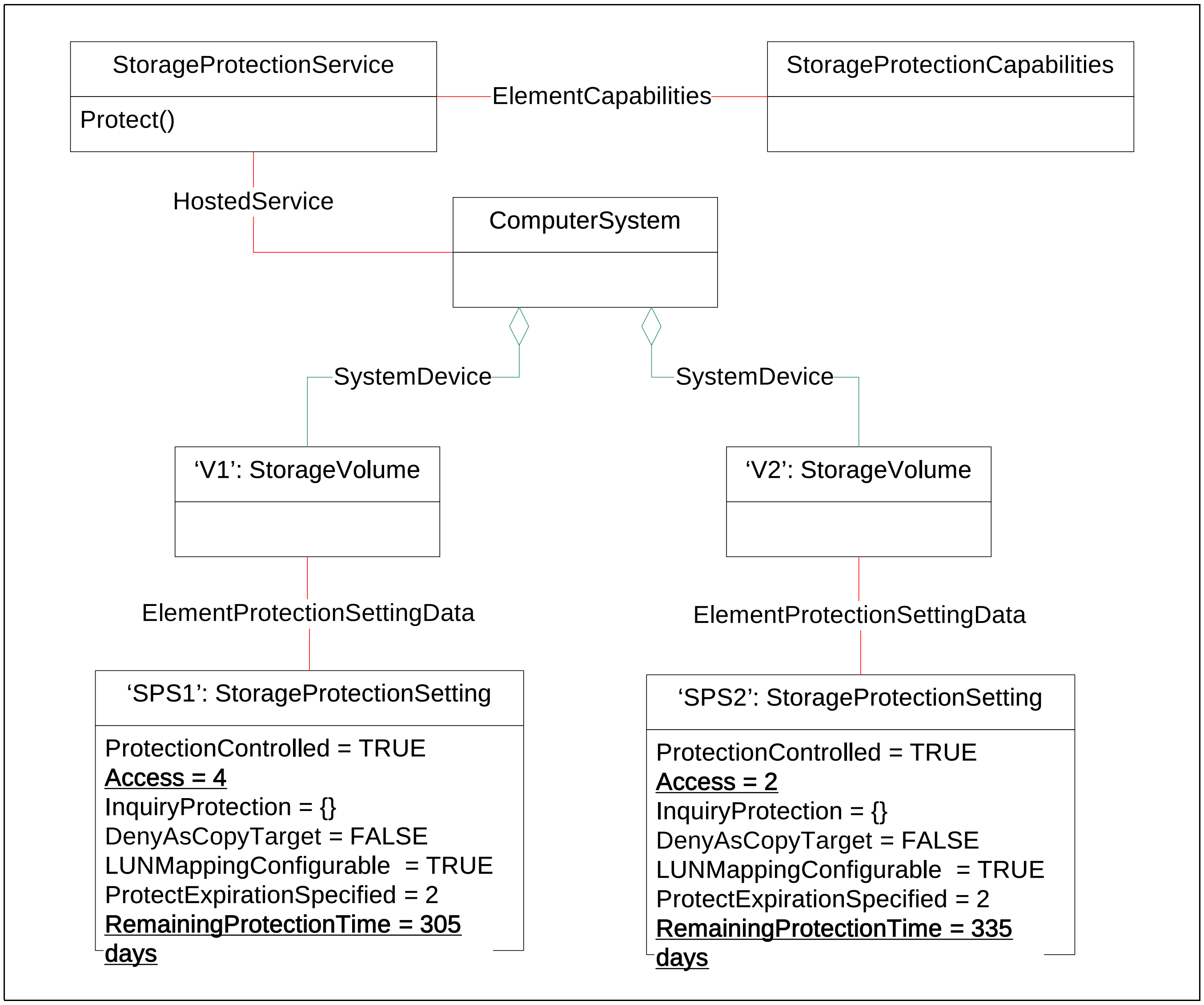
StorageProtectionService
Protect()
ElementCapabilities
StorageProtectionCapabilities
HostedService
ComputerSystem
SystemDevice
SystemDevice
'V1': StorageVolume
'V2': StorageVolume
ElementProtectionSettingData
ElementProtectionSettingData
'SPS1': StorageProtectionSetting
ProtectionControlled = TRUE
Access = 4
InquiryProtection = {}
DenyAsCopyTarget = FALSE
LUNMappingConfigurable = TRUE
ProtectExpirationSpecified = 2
RemainingProtectionTime = 305 days
'SPS2': StorageProtectionSetting
ProtectionControlled = TRUE
Access = 2
InquiryProtection = {}
DenyAsCopyTarget = FALSE
LUNMappingConfigurable = TRUE
ProtectExpirationSpecified = 2
RemainingProtectionTime = 335 days

Figure 116 - Step 4 - Volume Set to Read/Write Disabled

25.1.8.5 Step 5: Volume Access Change

Figure 117: "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.

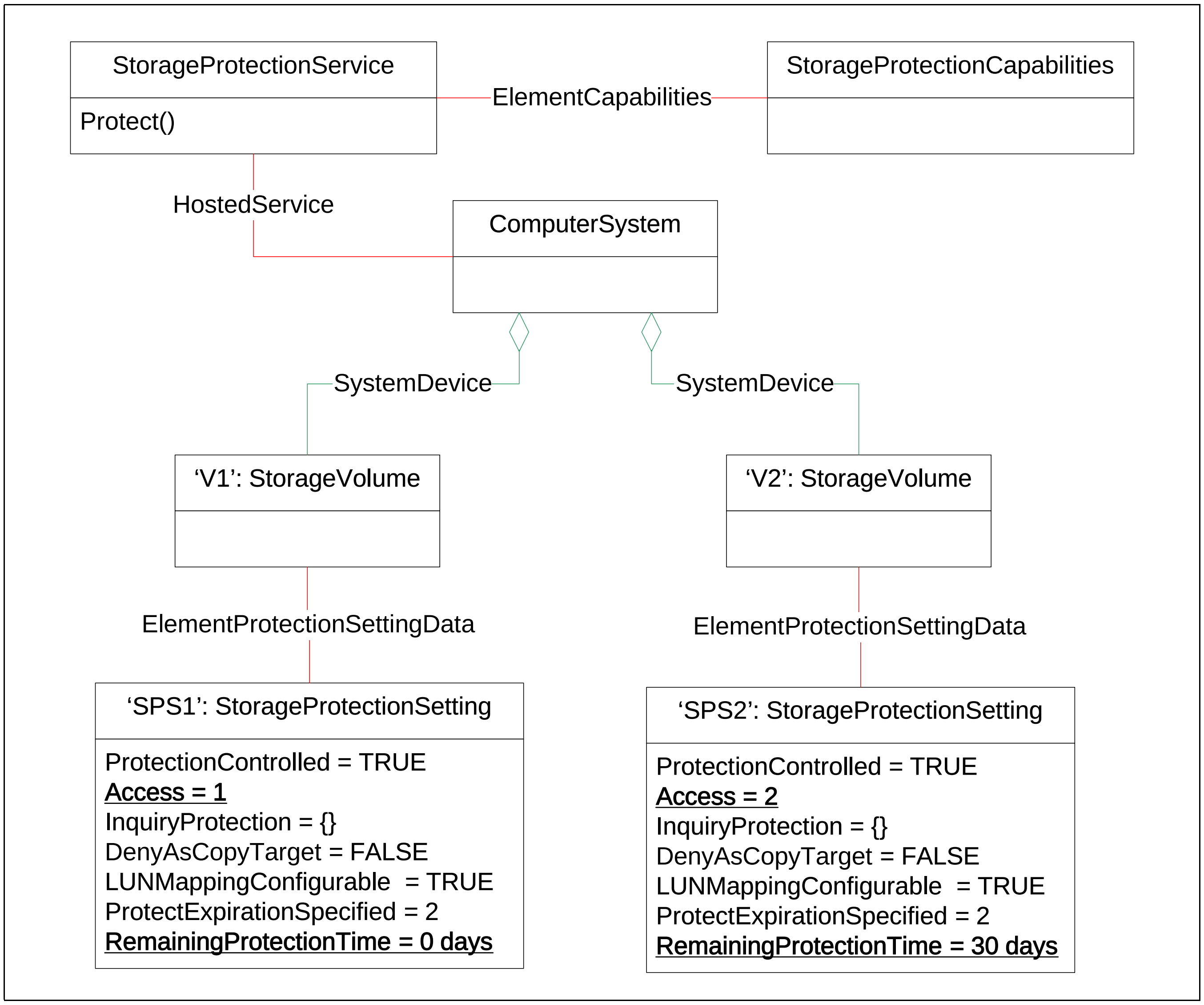
StorageProtectionService
Protect()
ElementCapabilities
StorageProtectionCapabilities
HostedService
ComputerSystem
SystemDevice
SystemDevice
'V1': StorageVolume
'V2': StorageVolume
ElementProtectionSettingData
ElementProtectionSettingData
'SPS1': StorageProtectionSetting
ProtectionControlled = TRUE
Access = 1
InquiryProtection = {}
DenyAsCopyTarget = FALSE
LUNMappingConfigurable = TRUE
ProtectExpirationSpecified = 2
RemainingProtectionTime = 0 days
'SPS2': StorageProtectionSetting
ProtectionControlled = TRUE
Access = 2
InquiryProtection = {}
DenyAsCopyTarget = FALSE
LUNMappingConfigurable = TRUE
ProtectExpirationSpecified = 2
RemainingProtectionTime = 30 days

Figure 117 - Step 5 Volume Access Changed

25.2 Health and Fault Management Consideration

Not defined in this standard

25.3 Cascading Considerations

Not applicable

25.4 Supported Profiles, Subprofiles, and Packages

Related Profiles for Storage Element Protection: Not defined in this standard.

25.5 Methods of the Profile

25.5.1 Protect

This method, defined in Table 500, 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 500 - 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 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

Table 500 - Methods of the Storage Element Protection Profile

Method: Protect			
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
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

25.6 Client Considerations and Recipes

25.6.1 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 obejct 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 #TwoYeasValue 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
```

25.6.2 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 obejct 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
```

25.7 Registered Name and Version

Storage Element Protection version 1.4.0 (Component Profile)

CIM Schema Version: 2.18.0

25.8 CIM Elements

Table 501 describes the CIM elements for Storage Element Protection.

Table 501 - CIM Elements for Storage Element Protection

Element Name	Requirement	Description
25.8.1 CIM_ElementCapabilities	Mandatory	Associates the capabilities to the service.
25.8.2 CIM_HostedService	Mandatory	Associates the service to the system providing the service.
25.8.3 SNIA_ElementProtectionSettingData	Mandatory	SNIA_ElementProtectionSettingData represents the association between the storage element to be protected and applicable protection setting.
25.8.4 SNIA_StorageProtectionCapabilities	Mandatory	
25.8.5 SNIA_StorageProtectionService	Mandatory	
25.8.6 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.

25.8.1 CIM_ElementCapabilities

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

Table 502 describes class CIM_ElementCapabilities.

Table 502 - SMI Referenced Properties/Methods for CIM_ElementCapabilities

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	The service.
Capabilities		Mandatory	The associated capabilities.

25.8.2 CIM_HostedService

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

Table 503 describes class CIM_HostedService.

Table 503 - SMI Referenced Properties/Methods for CIM_HostedService

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	The protection service.
Antecedent		Mandatory	The system providing the service.

25.8.3 SNIA_ElementProtectionSettingData

Created By: Extrinsic: Protect
Modified By: Static
Deleted By: External
Requirement: Mandatory

Table 504 describes class SNIA_ElementProtectionSettingData.

Table 504 - SMI Referenced Properties/Methods for SNIA_ElementProtectionSettingData

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	The storage element to be protected.
SettingData		Mandatory	The protection setting and status of the storage element.

25.8.4 SNIA_StorageProtectionCapabilities

Created By: Static
Requirement: Mandatory

Table 505 describes class SNIA_StorageProtectionCapabilities.

Table 505 - SMI Referenced Properties/Methods for SNIA_StorageProtectionCapabilities

Properties	Flags	Requirement	Description & 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.

Table 505 - SMI Referenced Properties/Methods for SNIA_StorageProtectionCapabilities

Properties	Flags	Requirement	Description & Notes
SupportedSynchronousActions		Mandatory	Value for storage element protection. 1 (Storage Element Protection).
SupportedAsynchronousActions		Mandatory	Value for element protection. 1 (Storage Element Protection).

25.8.5 SNIA_StorageProtectionService

Created By: Static
Requirement: Mandatory

Table 506 describes class SNIA_StorageProtectionService.

Table 506 - SMI Referenced Properties/Methods for SNIA_StorageProtectionService

Properties	Flags	Requirement	Description & Notes
Protect()		Mandatory	Configures the protection attributes of the storage element and prevent 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.

25.8.6 SNIA_StorageProtectionSetting

Created By: Extrinsic: Protect
Modified By: Extrinsic: Protect
Deleted By: DeleteInstance
Requirement: Mandatory

Table 507 describes class SNIA_StorageProtectionSetting.

Table 507 - SMI Referenced Properties/Methods for SNIA_StorageProtectionSetting

Properties	Flags	Requirement	Description & 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.

Table 507 - SMI Referenced Properties/Methods for SNIA_StorageProtectionSetting

Properties	Flags	Requirement	Description & Notes
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

Clause 26: Replication Services Profile

26.1 Description

26.1.1 Synopsis

Profile Name: Replication Services

Version: 1.6.0

Organization: SNIA

CIM schema version: 2.29

Central Class: ReplicationService

Scoping Class: ComputerSystem

26.1.2 Supported Profiles, Subprofiles, and Packages

Table 508 describes the supported profiles for Replication Services.

Table 508 - Supported Profiles for Replication Services

Profile Name	Organization	Version	Requirement	Description
Block Services	SNIA	1.6.0	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.

26.1.3 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 26.8 "CIM Elements" for a complete list of all properties and methods, including their description.

26.1.4 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

26.1.5 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 26.5.4 "Replication Services and Copy Services Properties and Methods Mapping" for mapping between Copy Services specific properties and properties introduced for Replication Services.

26.1.6 Key Components

Table 509 shows a list of key classes used by Replication Services. Refer to 26.5 "Methods of the Profile" and “CIM Elements” for additional details on methods and properties of these classes.

Clients should refer to 26.6 "Client Considerations and Recipes" for a list of steps to follow to utilize the replication service.

Table 509 - Key Classes

Class Name	Notes
ReplicationService	The main class for Replication Services. It contains methods for replication and group management, for example, CreateGroup, CreateElementReplica, CreateGroupReplica, ModifyReplicaSynchronization.
ReplicationServiceCapabilities	Contains a set of properties and methods that describe the capabilities of the service, for example, SupportedReplicationTypes, GetSupportedFeatures.
ReplicationGroup	Represents a group of elements participating in replication activities.
ReplicationSettingData	Contains options to customize replication operations, for example, pairing of group elements, TargetElementSupplier, CopyMethodology, ThinProvisioningPolicy.
ReplicationEntity	Represents information about an addressable entity without a known object model.
GroupSynchronized	Associates source and target groups.
StorageSynchronized	Associates source and target elements.

26.1.7 Replication Services Discovery

Figure 118 depicts the Replication Services discovery instance diagram.

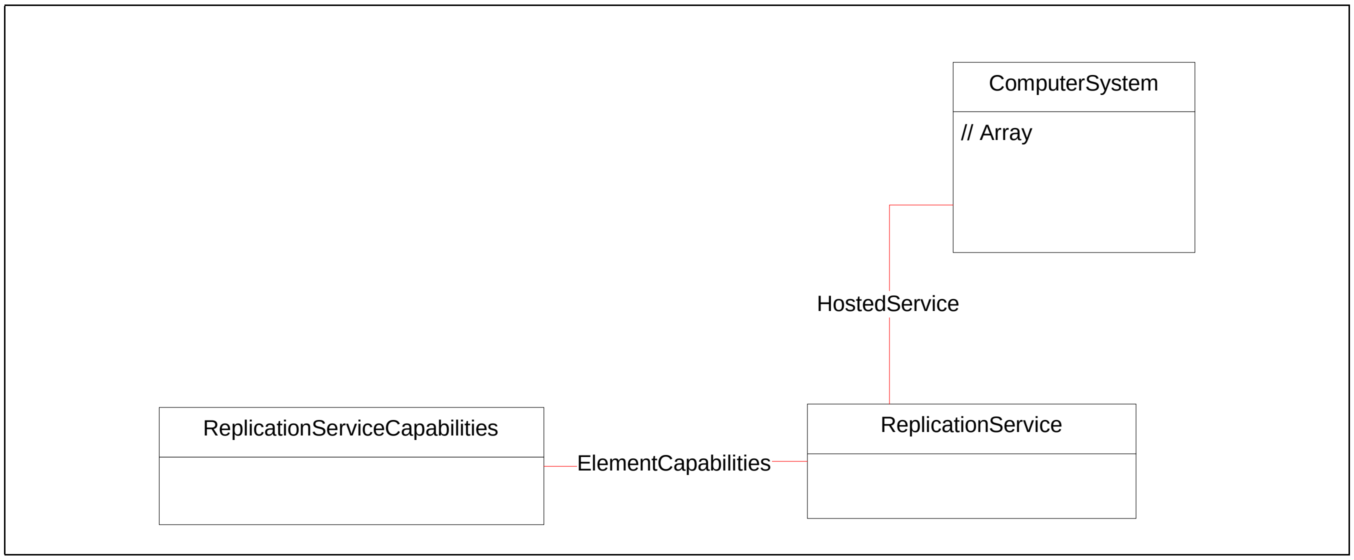

Figure 118 - Replication Services Discovery

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; and on the Storage Element Protection Profile for changing the protection of elements. The profile also relies on Cascading Profile, Masking and Mapping Profile, and Device Credential Profile for access to the remote resources.

26.1.8 Replication Services Capabilities

The single instance of the class ReplicationServiceCapabilities and its methods describe the various capabilities of the service. Clients should examine the ReplicationServiceCapabilities instance and invoke its methods to determine the specific capabilities of a replication service implementation.

26.1.9 SyncTypes

SyncTypes 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. Identical blocks in the source and target elements are shared via implementation-dependent means, to achieve space savings compared to full copies. Snapshots are commonly known as delta replicas.

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 510 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 510 - 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	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 must remain associated with the source element; ***Independent*** indicates the target element can exist without the source element.

With respect to “Target is Virtual copy of the Source,” the target element is not a “physical” copy of the source element, instead the system holds a collection of mapping information that map the target element data to the source element data.

26.1.10 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.

26.1.11 Locality of Target Elements

Locality specifies the relationship between the source and the target elements. Replication Services defines the following localities:

Local: It indicates the source and target elements are contained in a single managed system.

Remote: It indicates the source and target elements are contained in separate managed systems. In this case, the service must rely on a networking protocol for the copy operations.

The networking protocols are modeled using ProtocolEndpoint, which enables a replication service to reach a remote element. The property ProtocolEndpoint.ProtocolIFType specifies the protocol type, for examples, TCP, Fibre Channel, Other, etc.

Locality is important because it advertises the capability of replication service. For example, the property ReplicationServiceCapabilities.SupportedReplicationType may have values such as “Synchronous Mirror Local” and “Synchronous Mirror Remote.”

Figure 119 and Figure 120 show the local and remote instance diagrams, respectively.

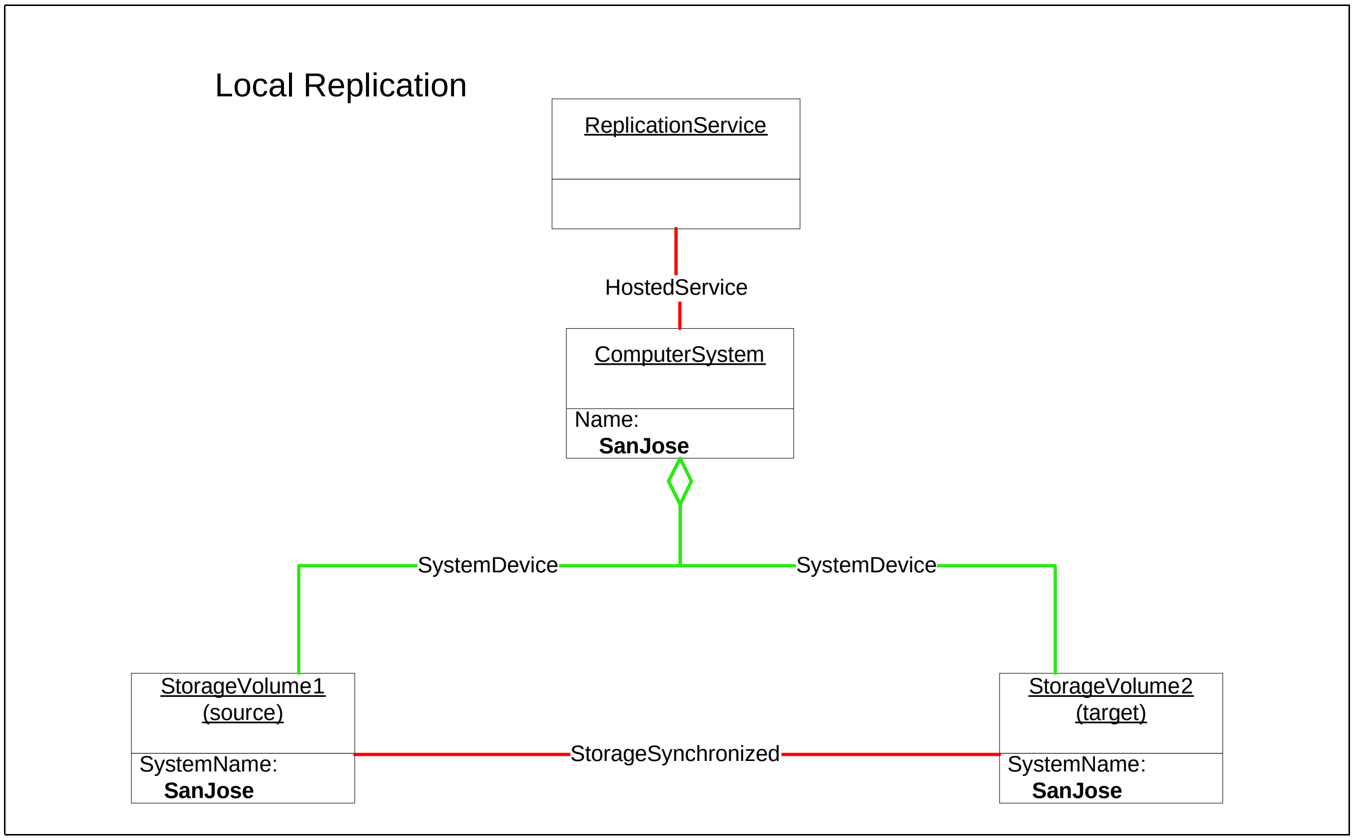

Figure 119 - Local Replica

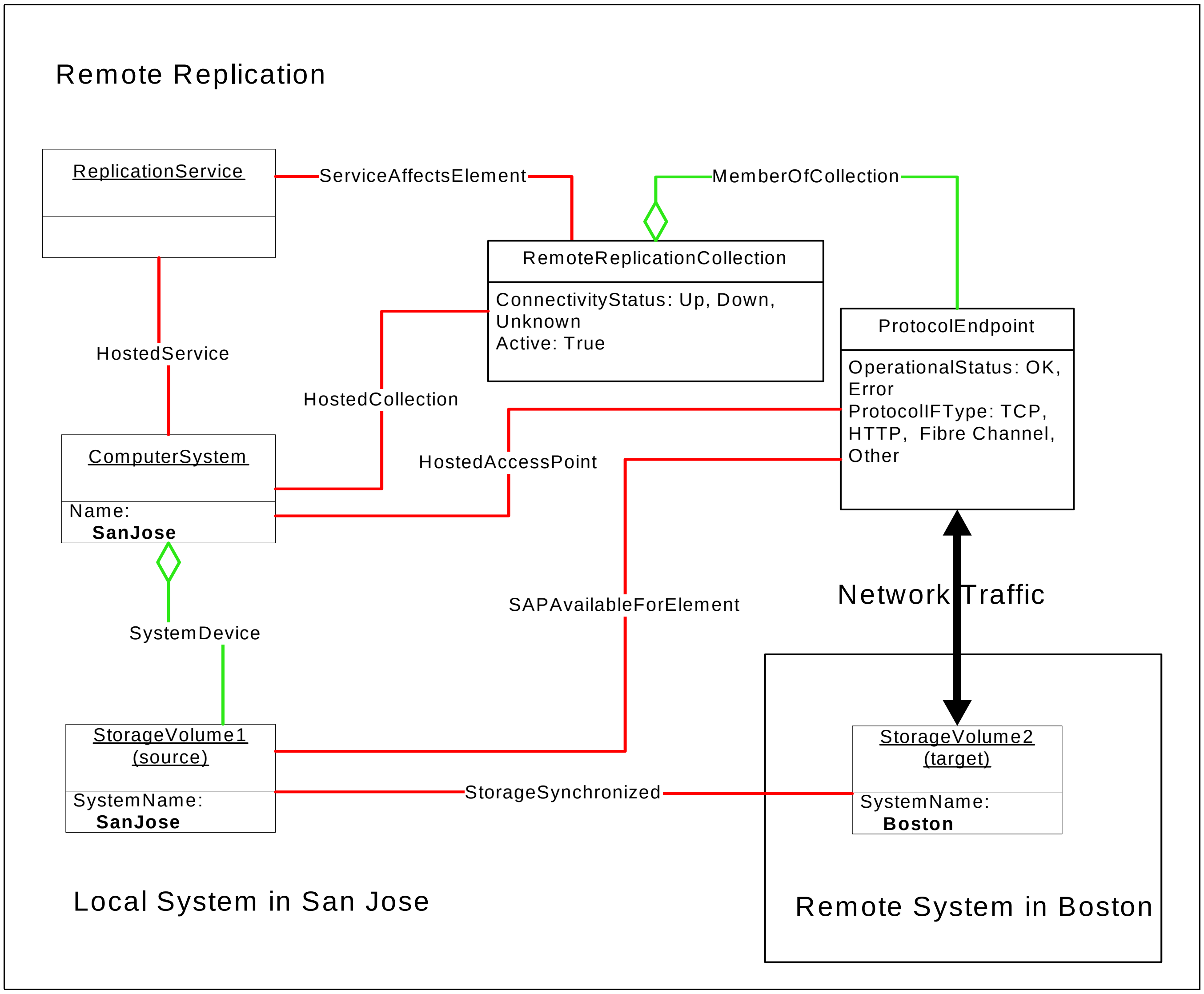

Figure 120 - Remote Replica

Each RemoteReplicationCollec-tion can have one or more paths to the remote system. As long as one of these paths to the remote system is up, the property RemoteReplicationCollec-tion.ConnectivityStatus indicates “UP”

Instances of RemoteReplicationCollection may statically be created by the implementation, or clients may be required to create such instances by invoking the extrinsic method CreateRemoteReplicationCollection. Clients subsequently can manipulate instances of RemoteReplicationCollection by invoking the intrinsic method ModifyInstance and/or the extrinsic methods AddToRemoteReplicationCollection and RemoveFromRemoteReplicationCollection.

The RemoteReplicationCollection abstracts the details of network connections to a remote system to allow clients to focus on whether a remote system is reachable or not. For example, the Figure 121, “Remote Replication over two Paths” shows the local system has two connections to a remote system. As long as one connection is functioning, there are replication operations between the local and the remote system.

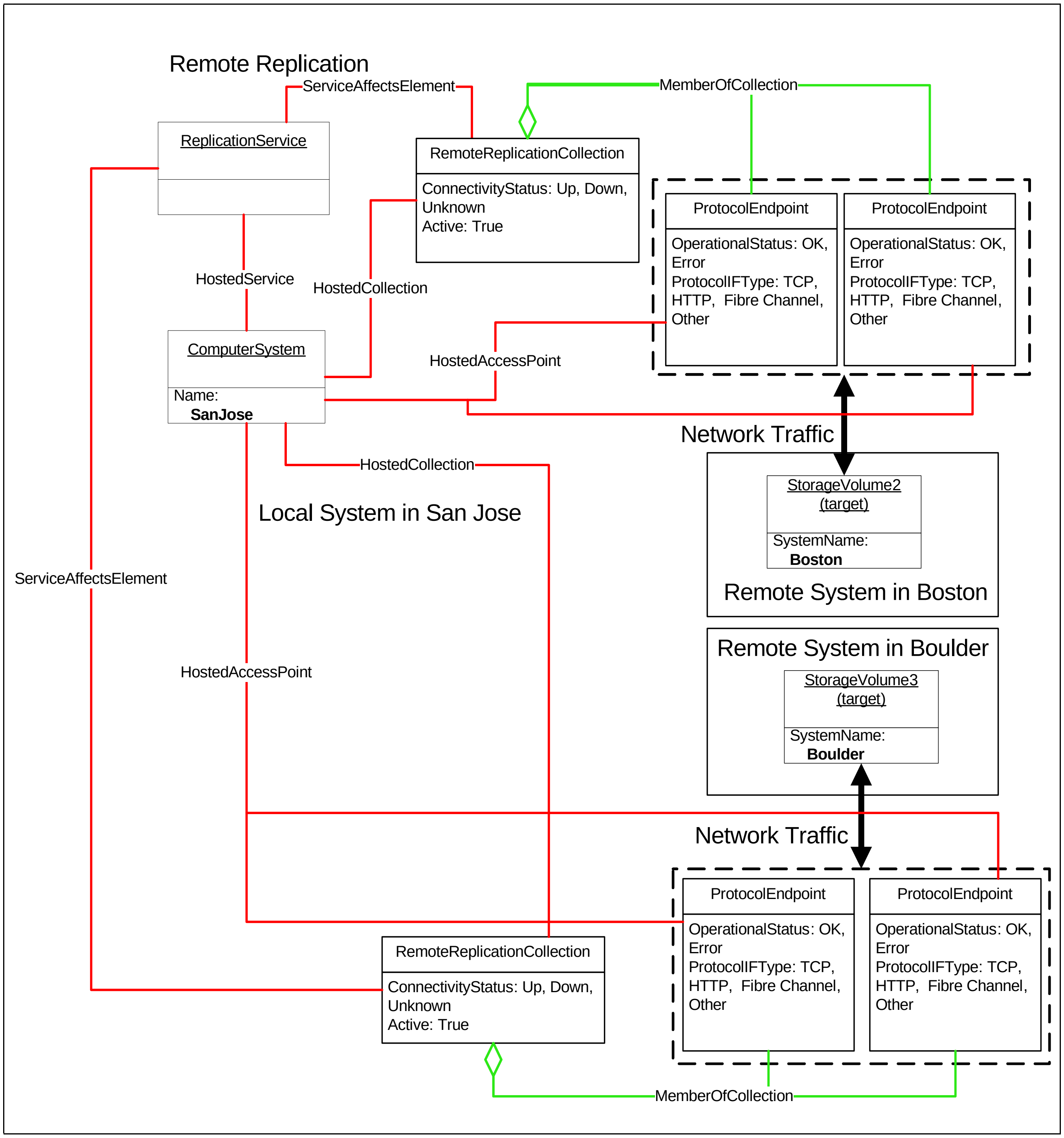

Figure 121 - Remote Replication over two Paths

Figure 122, “Expanded Remote Replica” shows a local system and two remote systems. The remote elements are associated to a remote ComputerSystem. In this configuration, all the replication operations utilize a single connection (ProtocolEndpoint) to all remote systems

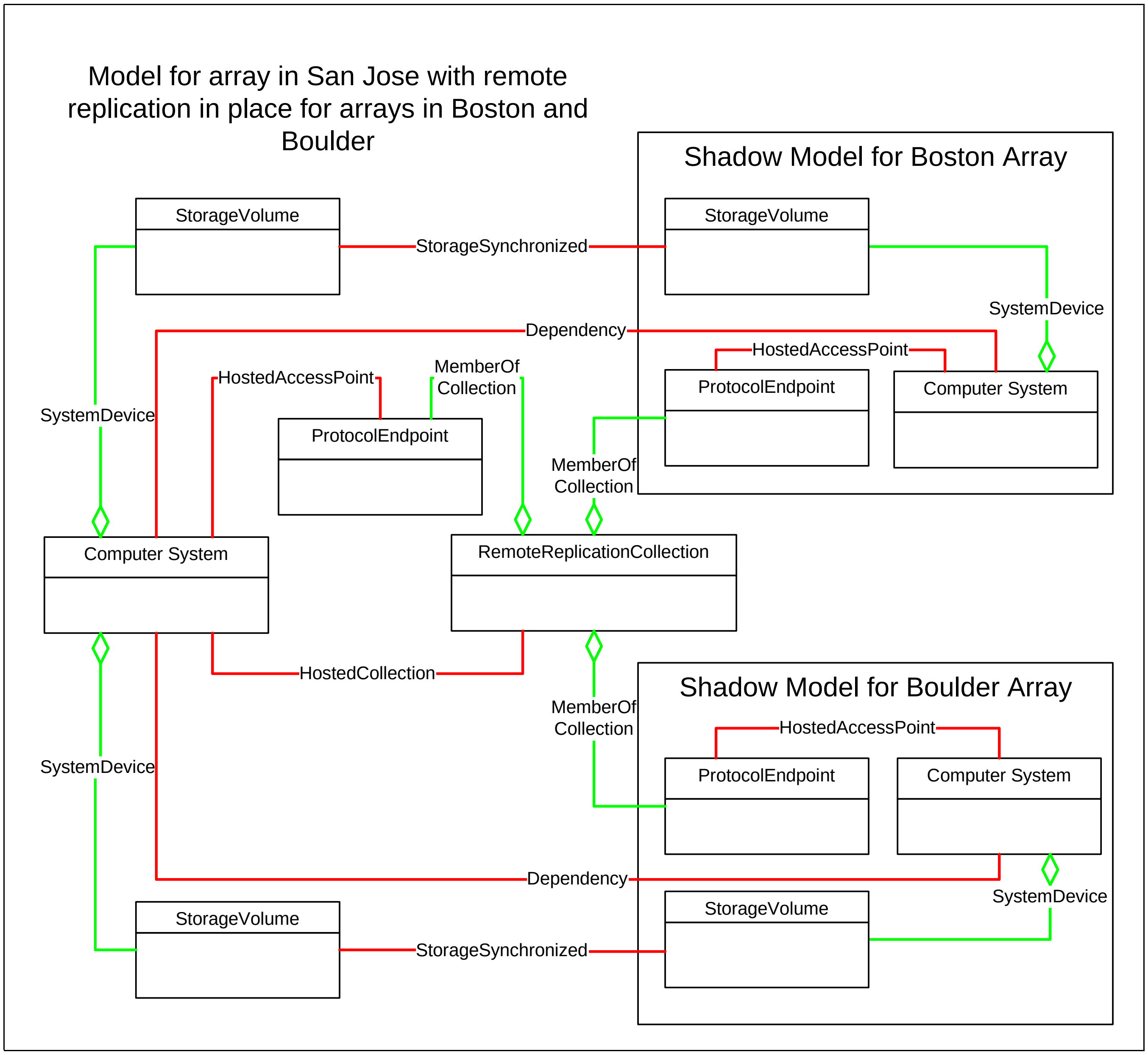

Figure 122 - Expanded Remote Replica

26.1.12 Remote Replication

Remote replication may require access information such as an RemoteServiceAccessPoint instance for the remote resources. See 26.3 "Cascading Considerations" for additional information.

26.1.13 Undiscovered Resources

An undiscovered resource is any addressable entity without a known object model. Generally, clients identify an undiscovered resource using one or more of the following:

- WWN (World Wide Name)
- URI (Uniform Resource Identifier)

- IP Address
- Remote ComputerSystem Objectpath
- Remote Filesystem Objectpath

In all cases, the assumption is that the underlying implementation "knows" how to perform the copy operation.

The Replication Service includes the necessary methods to create and manage the instances representing undiscovered resources. See the class ReplicationEntity (in 26.8 "CIM Elements") and the method AddReplicationEntity (26.5.2.15). Also in the replication service capabilities the absence of “Requires full discovery of target ComputerSystem“ in the SupportedFeatures property indicates the service supports undiscovered resources.

Figure 123 shows an instance of ReplicationEntity and its association to ReplicationService.

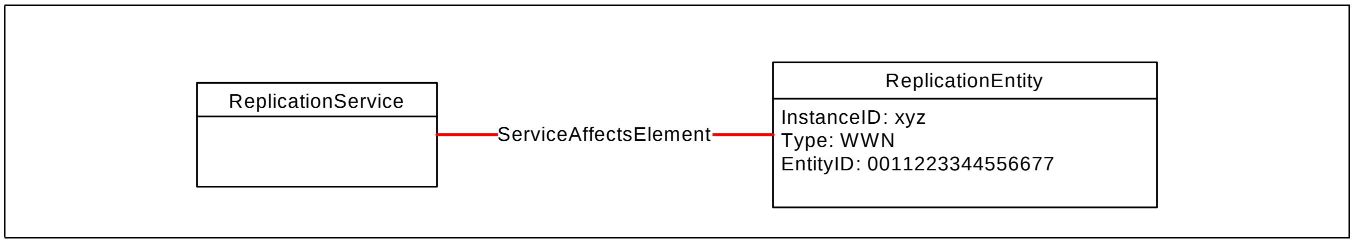

Figure 123 - An instance of ReplicationEntity

An instance of the StorageSynchronized association identifies the source and the target elements of a copy operation even in the case where the source or the target element is an instance of ReplicationEntity, which is a ManagedElement. Additionally, the StorageSynchronized.UndiscoveredElement property may indicate which elements in the copy operation are “undiscovered”. The possible values are:

- SystemElement -- the source element.
- SyncedElement -- the target element.
- Both -- both the source and the target elements.

Figure 124 shows an example of a StorageSynchronized association where the source element is a StorageVolume and the target element is a ReplicationEntity.

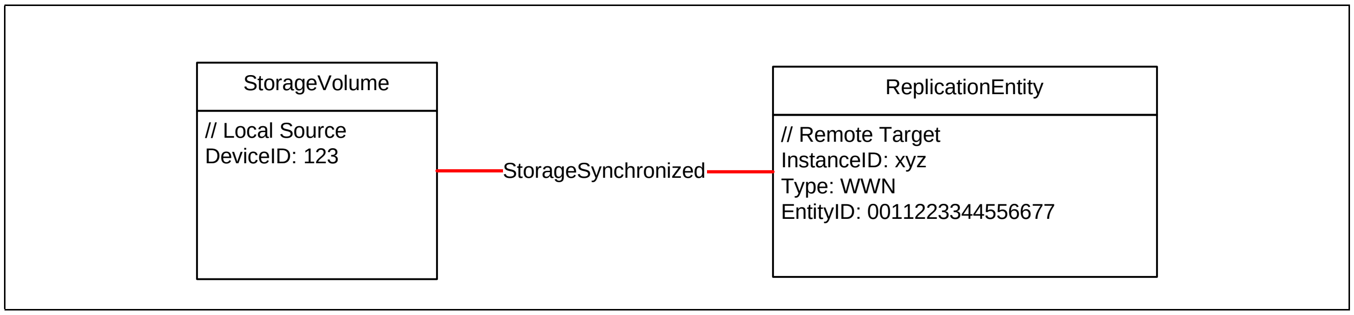

Figure 124 - StorageSynchronized and ReplicationEntity

26.1.14 Multi-hop Replication

In multi-hop replication, the target element of one copy operation can simultaneously be the source for another copy operation. As shown in Figure 125, multi-hop replication involves at least three elements.

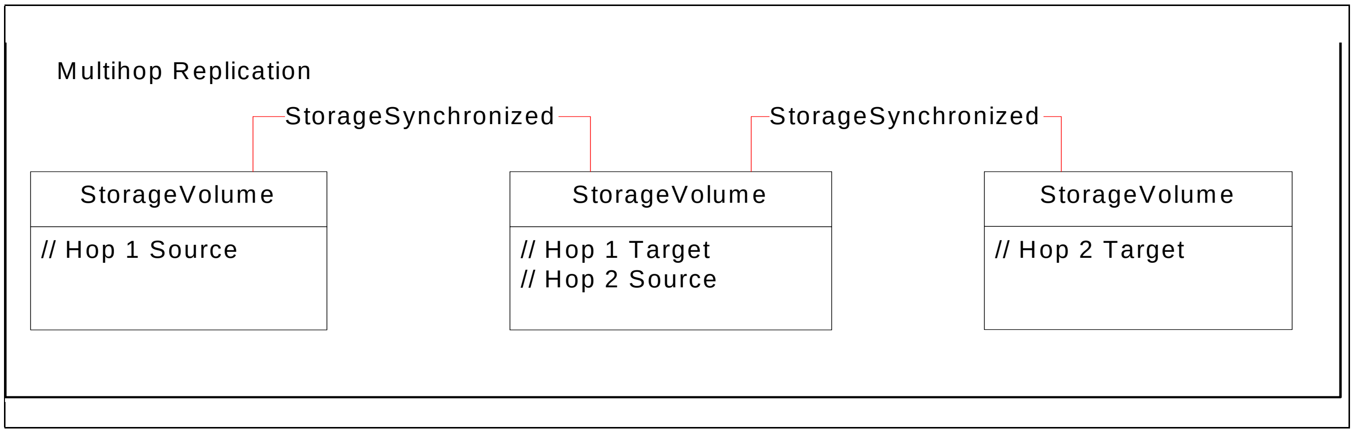

Figure 125 - Multi-hop Replication

If an implementation supports multi-hop replication, the supported features capabilities 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, must set the property ReplicationSettingData.Multihop appropriately (see 26.8 "CIM Elements" for details on Multi-hop specification). The capabilities method GetSupportedMaximum indicates the maximum number of hops supported by the implementation.

26.1.15 Groups

Replication Services utilizes Groups of elements to manage replication activities that include more than one source or target element in a copy operation. A major advantage of using groups is that an *operation*, such as *fracture,* (see 26.5.3.9 "GetSupportedOperations") may be performed on the group as a whole, instead of fracturing individual element pairs one by one.

The optional ReplicationGroup class represents a collection of ordered storage elements.

Key features of replication groups are:

- A group can be the source and/or the target of a copy operation.
- Elements of a group may be optionally declared *Consistent.*
- A group may optionally be declared as temporary (Persistent = false).
- A group may contain zero elements (an empty group).

Replication Services includes methods to create and delete a group, and methods to add elements or pair of elements to an existing group(s) or to remove elements from a group.

Certain copy operations such as copying one source element to many target elements (one-to-many) may result in the service creating a temporary group to keep track of all the target elements. The service may delete temporary groups that are no longer associated with a copy operation. Deleting a temporary group does not affect the elements associated with the group.

The method ReplicationService.CreateGroupReplica() is used to copy a group of elements. The property ReplicationSettingData.Pairing determines the pairing of the source and the target elements. Possible values are: *Exact order* and *Optimum*. *Exact order* means the first element of the source group is copied to the first element of the target group, the second element of the source group is copied to the second element of the target group, and so on. *Optimum* means in order to minimize any resource and data flow contentions, if possible, pair the source and the target elements in such a way that they are on different data paths.

An implementation may allow the target group to have more (or fewer) elements than the source group.

See the ReplicationServiceCapabilities.GetSupportedReplicationSettingData() method for Pairing and for UnequalGroupsAction capabilities.

Figure 126 shows group instances and the associated storage volumes.

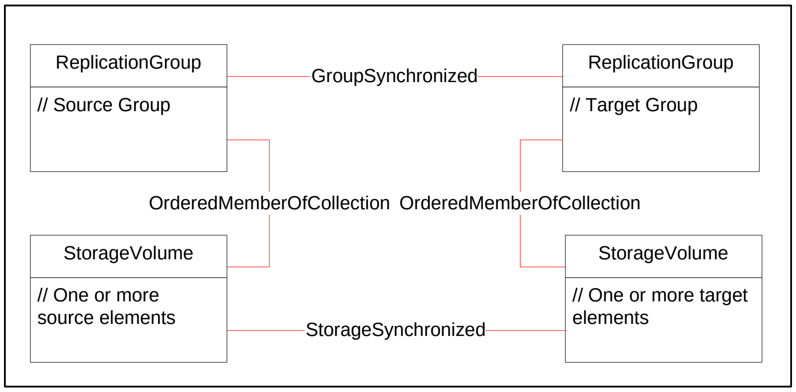

Figure 126 - Group Instances

The association between ReplicationGroup and its storage elements (e.g. StorageVolume) is OrderedMemberOfCollection to maintain the order of the storage elements to facilitate pairing of the source and the target group elements.

26.1.15.1 Composite Groups

A Composite Group is a group that includes storage elements from multiple storage arrays.

26.1.15.2 Consistency Groups

A Consistency Group is a set of elements that have an "Application Consistent View." Application Consistent View is a set of elements that collectively represent some resource in a known state.

Block Storage Systems can only maintain state as to whether a group of elements is “sequentially consistent” or not.

The instrumentation may support consistency groups for a given copy type and mode. The CreateGroupReplica method allows a client to specify the target group to be consistent.

26.1.15.2.1 Sequentially Consistent

A group of target elements is considered to be "sequentially consistent” if each element is updated in the same order as the application updates the corresponding source elements. Sequentially Consistent is also known as Dependent Write Consistency.

Figure 127 shows the target elements that have a sequentially consistent view at all times. Once the connection between volume2 and volume5 fails, all subsequent copy operations to the target elements stop, therefore maintaining the consistency of the target elements.

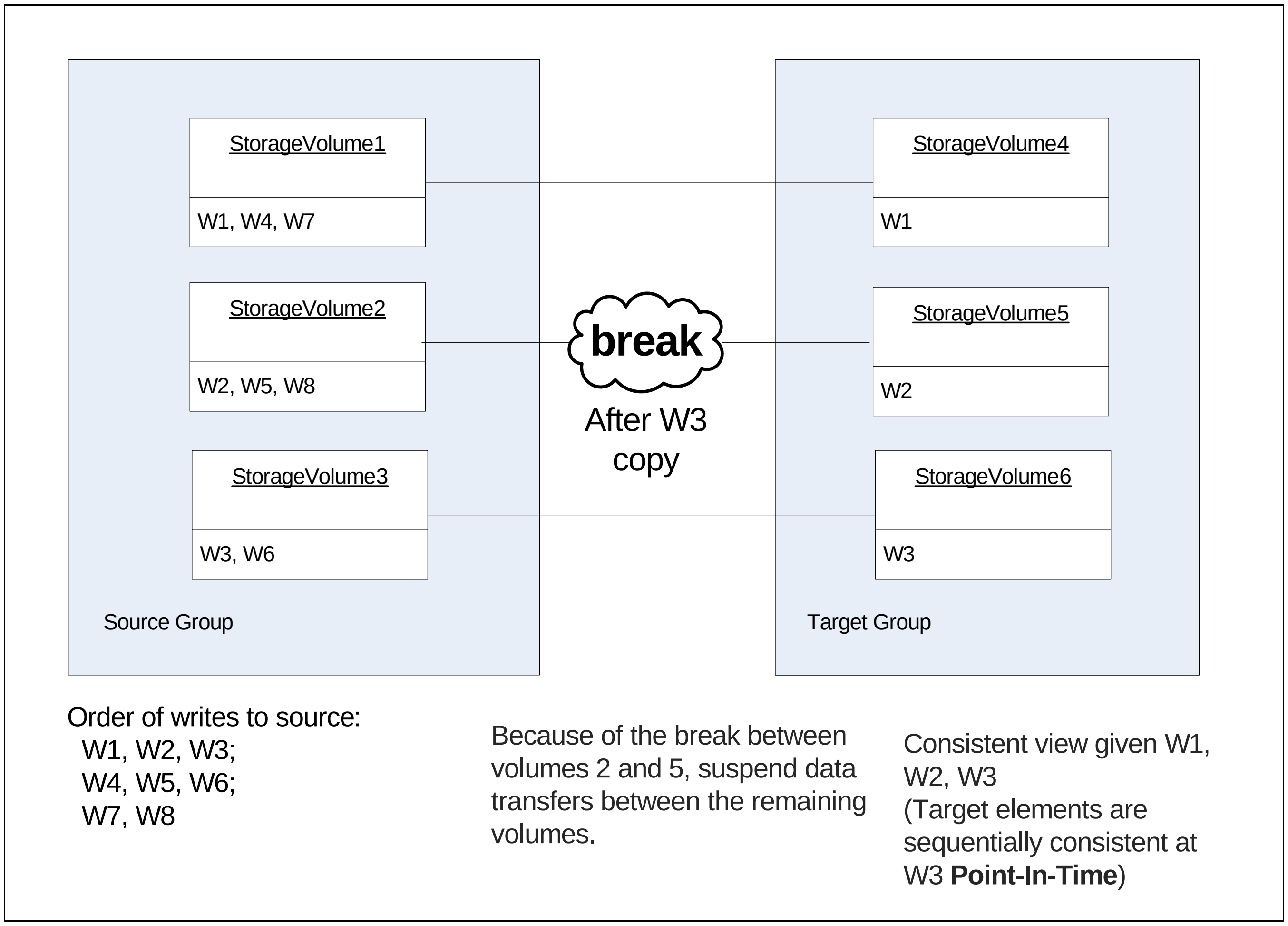
StorageVolume1
W1, W4, W7
StorageVolume2
W2, W5, W8
StorageVolume3
W3, W6
Source Group
StorageVolume4
W1
StorageVolume5
W2
StorageVolume6
W3
Target Group
break
After W3 copy
Order of writes to source: W1, W2, W3; W4, W5, W6; W7, W8
Because of the break between volumes 2 and 5, suspend data transfers between the remaining volumes.
Consistent view given W1, W2, W3 (Target elements are sequentially consistent at W3 Point-In-Time)

Figure 127 - Sequentially Consistent Example

26.1.16 Associations

Replication Services utilizes a number of stateful associations to associate source and target groups, source and target elements, and, when necessary, the individual elements to their corresponding point-in-time aspect.

Figure 128 shows the associated groups with equal number of source and target elements.

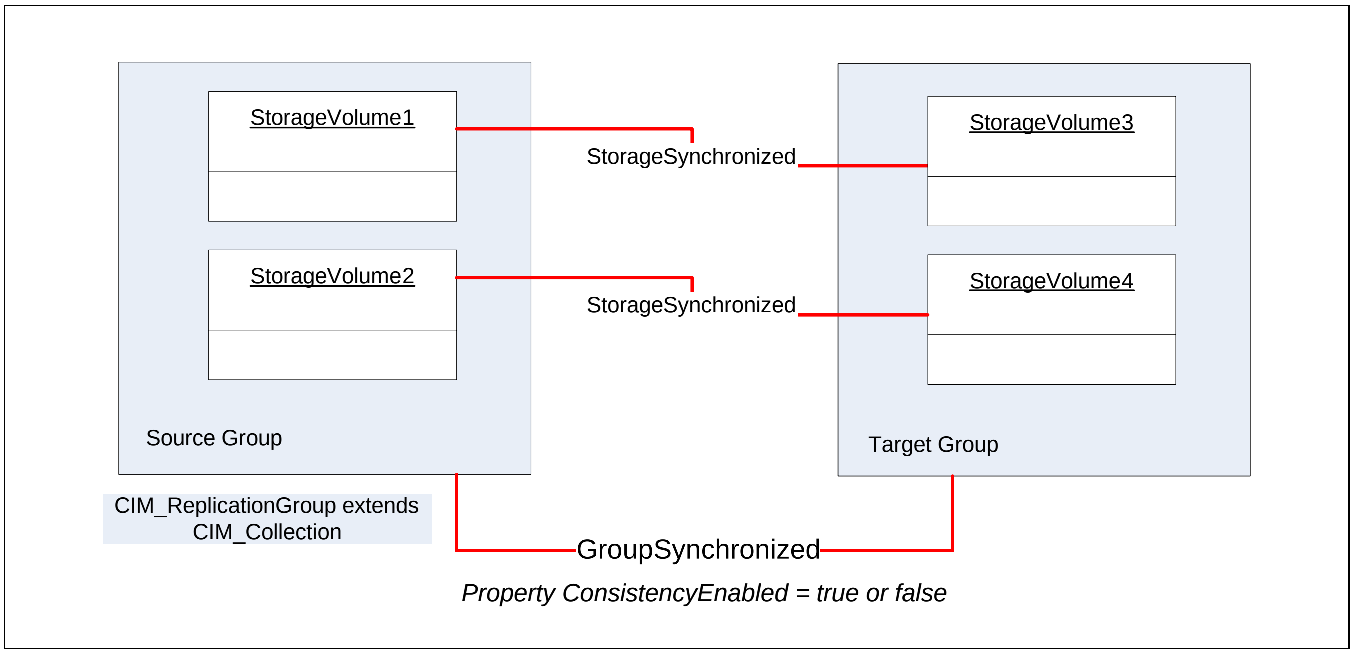

Figure 128 - Associated Groups and Elements

26.1.16.1 GroupSynchronized Association

This association relates source and target groups, or, for a one-to-many relationship, relates a source element to a target group. The association's property ConsistencyEnabled indicates whether the target elements are required to be *Consistent* or not.

Within a group, the SyncType and Mode properties of all subordinate StorageSynchronized associations between the source and the target elements shall be the same. The SyncType and Mode properties of the GroupSynchronized association shall also be the same as the SyncType and Mode properties of subordinate StorageSynchronized associations.

This association relates the individual source and target elements. The association's property CopyState indicates the current state of the association. Some possible values of CopyState are *Initialized* or *Synchronized*.

A StorageSynchronized association can participate in only one pair of related replication groups.

26.1.16.2 SettingsDefineState Association and SynchronizationAspect Instance

The SettingsDefineState associates an element (e.g., a StorageVolume), or a group of elements (e.g. a ReplicationGroup), 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 129). 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). Creating a SynchronizationAspect of a Snapshot is particularly useful when a client wants to capture a point-in-time copy at a given time; however, the client wants to create the actual target element at a later time, perhaps when it is more convenient.

If an instance of a SynchronizationAspect is associated to a group of elements, the property “WhenPointInTime” applies to all elements of the group, indicating the point-in-time copy of all elements is created at the same exact time.

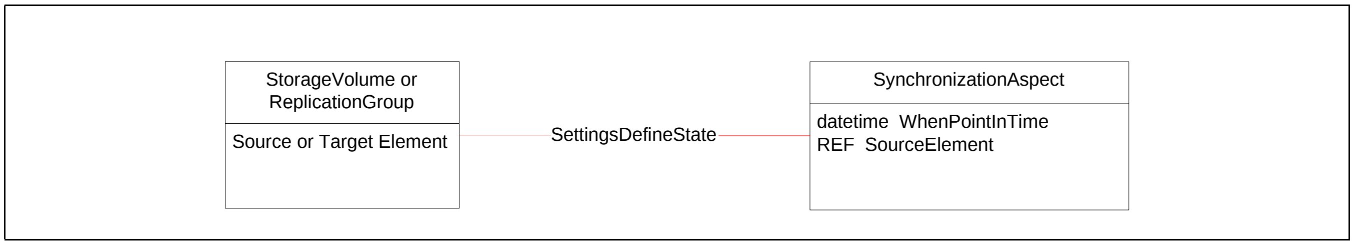

Figure 129 - SettingsDefineState Association

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 130 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.CopyState has a value of “Operation Completed.”

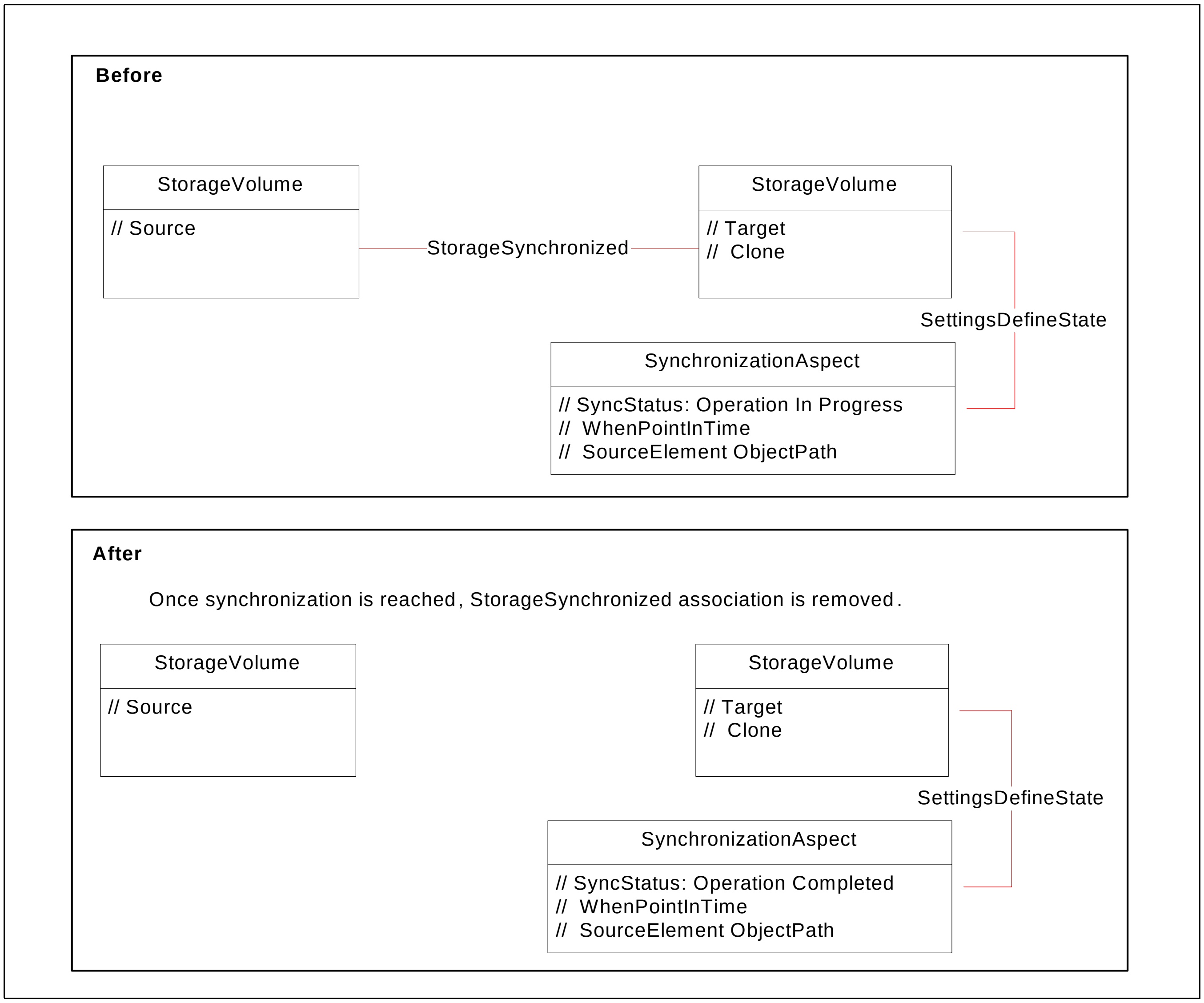

Figure 130 - SynchronizationAspect Instance

26.1.16.3 One-to-Many Association

Using a replication group, Replication Services allows for one source element to be copied to many target elements.

As shown in Figure 131, one source element is associated to more than one target element. With ConsistencyEnabled set to true, if the link to a target element is broken, all subsequent copy operations to all other target elements are suspended. This ensures all the target elements contain the same exact data.

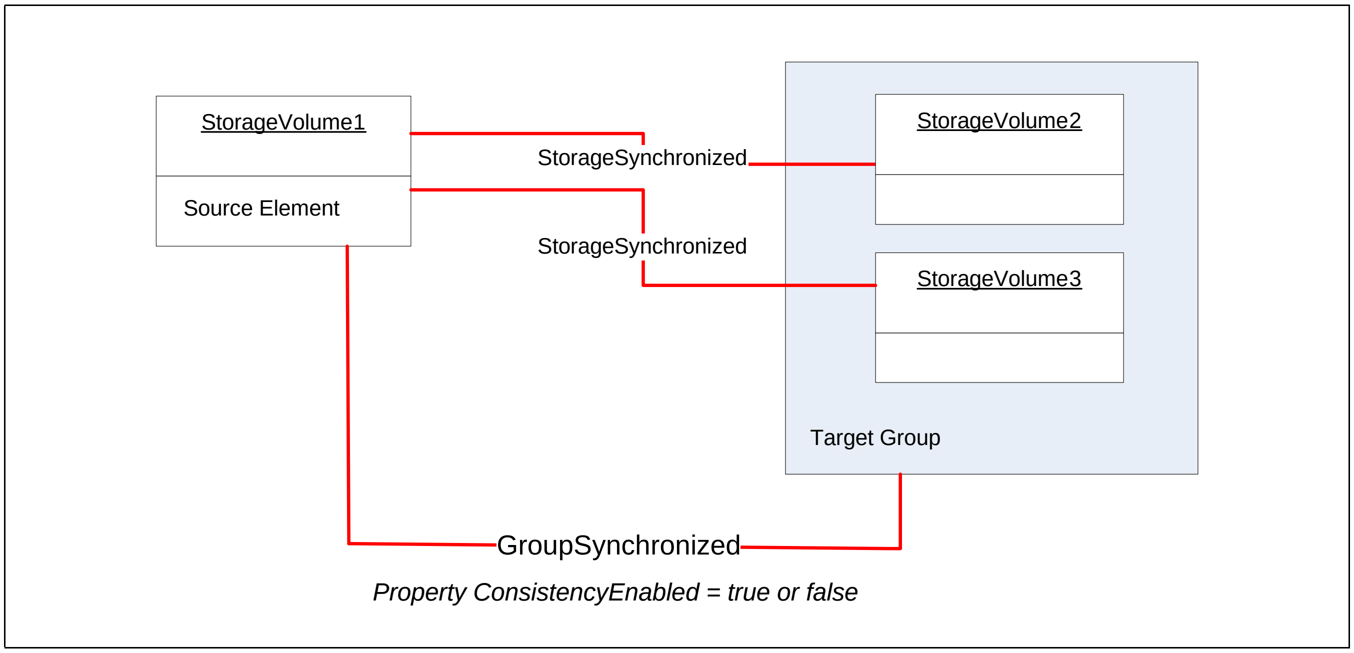

Figure 131 - One-to-Many Association

26.1.17 Operations on List of Synchronizations

Primarily for scalability reasons, an implementation optionally may offer the ability to perform an operation, such as fracture, on a list of synchronization associations. The list of synchronization associations may be a collection of independent associations or a subset of StorageSynchronized associations belonging to a source and a target replication groups. The method ModifyListSynchronization and GetSupportedListOperations are used for list modifications.

26.1.18 State Management For Associated Replicas

Both mirror and snapshot replicas maintain stateful associations with source elements. In the case of clone replicas, the replication associations to the source elements exist while the copy operation is in progress.

The CopyState property of the replication 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 RequestedCopyState. 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 CopyState, as reflected in the property RequestedCopyState, is Synchronized, which indicates the replica element has the same data as the source element. The RequestedCopyState property will contain “Not Applicable” once the requested CopyState is achieved.

The GroupSynchronized association between the source and target groups also includes the CopyState properties. If all values of StorageSynchronized.CopyState of source and target associations are the same (i.e., Synchronized), GroupSynchronized.CopyState will also have the same value. On the other hand, any mismatch in the StorageSynchronized.CopyState values, will render the GroupSynchronized.CopyState property to have a value of *Mixed*.

Unplanned states, such as Broken, Aborted, or Partitioned can be entered from any other state and generally indicate an unusual circumstance. Recovery from the Broken or Partitioned state may be automatic once the error condition is resolved, or it may require a client to intervene with a “Resync” operation (see 26.5.3.3

"GetSupportedFeatures"), or a “Resume” operation. Continuing from an Aborted state requires a client to intervene with a Resync operation. In this situation, the implementation may indicate a Resync operation is required by the setting the ProgressStatus to "Waiting for resync". Additionally, the copy operation may be temporarily stopped due to system or connection bandwidth. In this case the ProgressStatus will be set to “Pending.” See 26.5.3.3 "GetSupportedFeatures".

If after the error condition is resolved, the CopyState indicates “Suspended” state, in order to resume the copy operation it is necessary for the client to issue a “Resume” operation.

If the CopyState indicates “Invalid”, generally, it means the storage system is unable to determine the state of the copy operation. In this situation, the client needs to “detach” and ‘re-establish” the replication relationship.

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.

Table 511 describes the supported CopyStates.

Table 511 - CopyStates Values

CopyState value	Description
Initialized	The source and target elements are associated. The copy operation has not started -- no data flow.
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 a resume operation is completed.
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.
Failedover	Reads and writes to/from the target element. Source element is not “reachable”.
Inactive	Copy operation has stopped, writes to source element will not be sent to target element.
Prepared	Initialization is completed, the copy operation has started, however, the data flow has not started.
Aborted	The copy operation is aborted with the Abort operation. Use the Resync Replica operation to restart the copy operation.
Skewed	The target has been modified and is no longer synchronized with the source element or the point-in-time view. Use the Resync Replica operation to resynchronize the source and target elements.
Mixed	Applies to the CopyState of GroupSynchronized. It indicates the StorageSynchronized associations of the elements in the groups have different CopyState values.

Table 511 - CopyStates Values

CopyState value	Description
Partitioned	The state of replication relationship can not be determined, for example, due to a connection problem.
Invalid	The array is unable to determine the state of the replication relationship, for example, after the connection is restored; however, either source or target element has an unknown status.

Figure 132 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 such time a Modify method is used to Activate the synchronization.

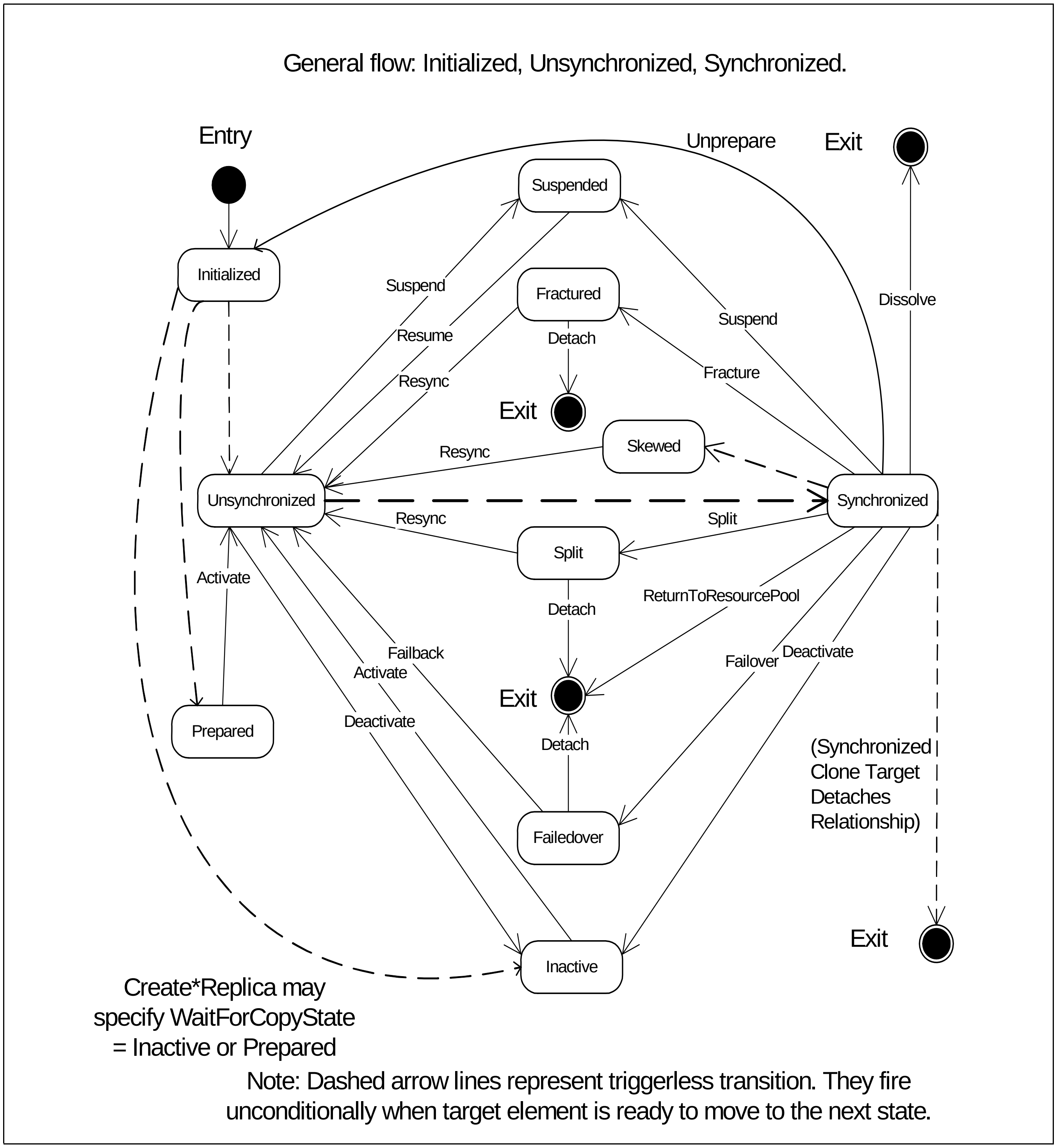
General flow: Initialized, Unsynchronized, Synchronized.
Entry
Unprepare
Exit
Suspended
Initialized
Suspend
Fractured
Dissolve
Suspend
Resume
Detach
Fracture
Resync
Exit
Skewed
Resync
Unsynchronized
Synchronized
Resync
Split
Split
Activate
Detach
ReturnToResourcePool
Failback
Activate
Failover
Deactivate
Exit
Deactivate
Prepared
Detach
(Synchronized Clone Target Detaches Relationship)
Failedover
Exit
Inactive
Create*Replica may specify WaitForCopyState = Inactive or Prepared
Note: Dashed arrow lines represent triggerless transition. They fire unconditionally when target element is ready to move to the next state.

Figure 132 - CopyState Transitions

26.1.18.1 Synchronized CopyState

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/mapping information) for the snapshot have been created. Synchronization for the snapshots is achieved rapidly in comparison to synchronization of Mirrors or Clones.

Depending on implementation, the clone target element detaches automatically when the target element becomes synchronized; otherwise, the client needs to explicitly request a detach operation. See the method ReplicationServiceCapabilities.GetSupportedFeatures in 26.5.3.3.

Figure 133 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”

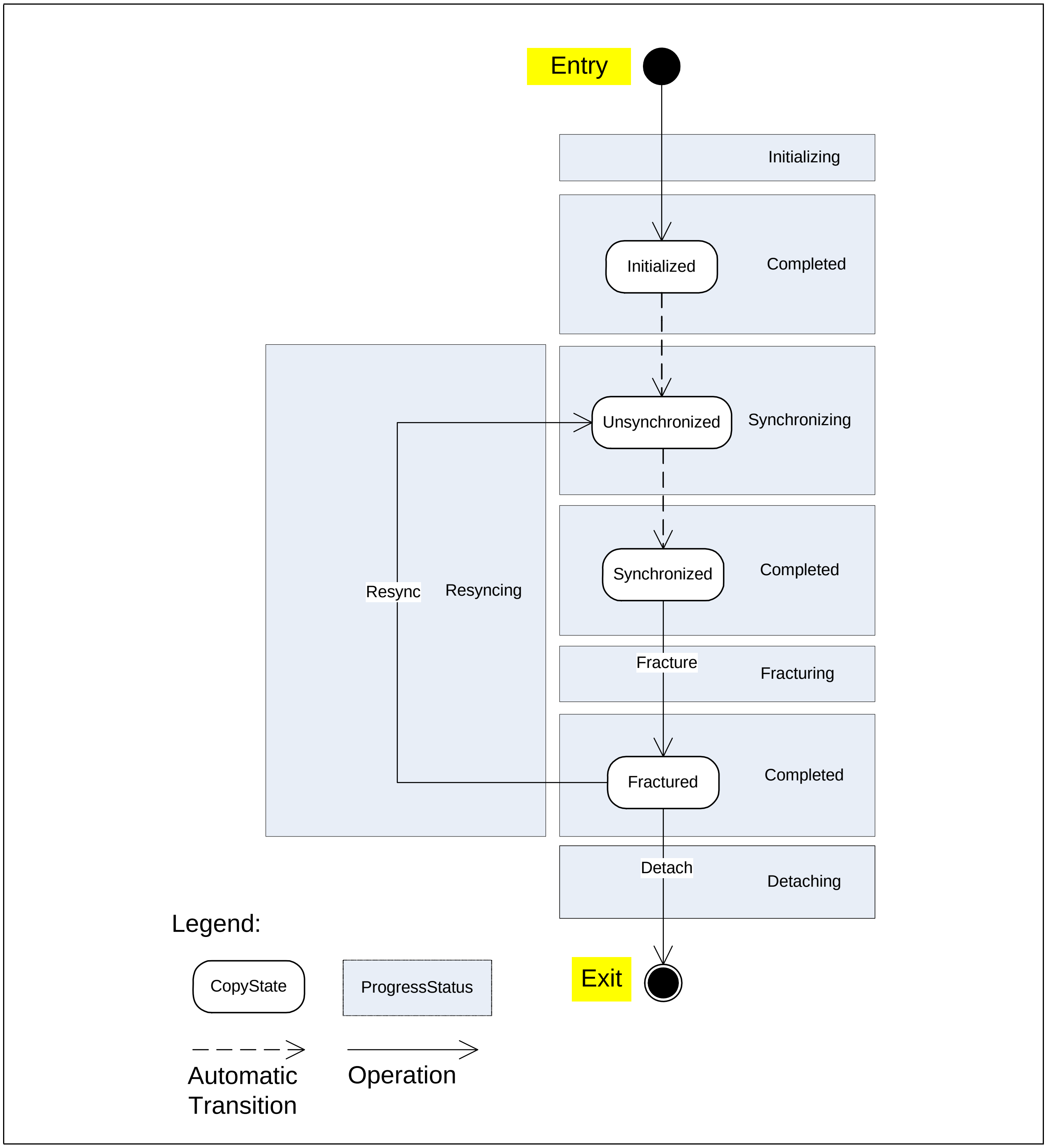

Figure 133 - Sample CopyState and ProgressStatus Transitions

26.1.19 Unsynchronized and Skewed CopyStates

Unsynchronized CopyState indicates the target element is not an exact copy of the source element (or the source's point-in-time). The copy operation automatically continues until the synchronization between the source element (or its point-in-time) and the target element is reached.

The Skewed CopyState is similar to the Unsynchronized CopyState except that the synchronized relationship remains in the Skewed state until a client issues the Resync operation (ModifyReplicaSynchronization or ModifyListSynchronization invoke methods). As an example: Committing write operations to a Snapshot target element causes the source and the target elements to become Skewed.

26.1.20 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 associations (StorageSynchronized and GroupSynchronized for Mirror and Snapshot copy types) and the newly created target elements shall be accessible. The StorageSynchronized or GroupSynchronized associations 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 (or group), unless there is no target element (or group) such as CreateSynchronizationAspect or when the target element (or group) is deleted (ReturnToResourcePool). In this case, the AffectedJobElement points to the source element (or group). 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 methods CreateElementReplica and CreateGroupReplica.

26.1.21 Host Access Restrictions

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 must 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 18: 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.

26.1.22 Read Only Elements

Clients can request a newly created target element to be "Read Only" to the host. For example, to create a "Read Only" target element, as a parameter, supply a ReplicationSettingData object with ReadOnly = 3 to a method such as ReplicationService.CreateElementReplica. As a result of this request, the StorageSynchronized association's ReadOnly property will have a value of 3 to indicate the SynchedElement (i.e. the target element) is read only to the host.

The implementation may also support the ability for the clients to make the source element "Read Only" to the host, before the copy operation begins to ensure there is no change in the source element's data while the data is being copied. Once the copy operation completes and the StorageSynchronized association between the source and the target elements is removed, the replication service will remove the "Read Only" state of the source element.

Refer to the GetSupportedReplicationSettingData (section 26.5.3.17) and GetDefaultReplicationSettingData (section 26.5.3.18) methods of the ReplicationServiceCapabilities class to determine whether the implementation supports Read Only elements.

The implementation may have also implemented the Storage Element Protection Profile to allow clients to set the state of the elements to “Read Only” etc.

26.1.23 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 ModifyListSynchronization) method with the “Return To ResourcePool” operation parameter, which also removes the synchronization association.

26.1.24 Completion of Long Operations

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 (or GroupSynchronized) 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 a replication operation was specified using a WaitForCopyState parameter and the method is executing under the control of a job, 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 operation may continue until a steady state is achieved. For example, in the Figure 132, “CopyState Transitions” diagram, Inactive and Synchronized states are considered steady states; whereas Initialized and Unsynchronized are transient states.

During the copy operation, the AffectedJobElement association associates the job to the target element or to the target group. In case an operation does not have a target element (e.g. CreateSynchronizationAspect), the AffectedJobElement is the source element.

26.1.25 Managing Background Copy

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 and CreateGroupReplica 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).

26.1.26 Managing CopyPriority

A client may be able to manipulate the CopyPriority of a StorageSynchronized association -- see the ReplicationServiceCapabilities.GetSupportedFeatures method in 26.8 "CIM Elements", which would indicate “Adjustable CopyPriority”.

CopyPriority allows a client to manage the copy I/O rate and the priority of peer I/O operations relative to host I/O operations. Before the copy operation starts, the CopyPriority may be specified in ReplicationSettingData parameter supplied to the CreateElementReplica or CreateGroupReplica. After the copy operation starts, the StorageSynchronized.CopyPriority property may be modified by invoking the intrinsic ModifyInstance method.

The CopyPriority values are:

- Low - copy operation lower priority than host I/O.
- Same - copy operation has the same priority as host I/O.
- High - copy operation has higher priority than host I/O.
- Urgent - copy operation to be performed as soon as possible, regardless of the host I/O requests

In a group copy operation, adjusting the CopyPriority of one StorageSynchronized association belonging to the group shall cause the CopyPriority of the remaining group StorageSynchronized associations to be adjusted likewise.

26.1.27 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 5: "Block Services Package" for the details of creating and modifying a storage setting.

26.1.28 Finding and Creating Target Elements

The extrinsic method ReplicationService.GetAvailableTargetElements is used to locate the available target elements for a given source and SyncType. 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 5: "Block Services Package" for creating (specialized) elements and modifying the Usage value of existing elements.

Refer to 26.5.3.17 "GetSupportedReplicationSettingData" and 26.5.3.3 "GetSupportedFeatures" to determine if the implementation automatically creates target elements, and if specialized elements are required for the desired SyncType.

26.1.29 Using StoragePools (e.g. ResourcePools) for Replicas

Replicas are allocated from storage pools (e.g. resource 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”.

26.1.29.1 Delta Replica StoragePools

Depending on the implementation, the Snapshot targets may require a fixed space consumption or variable space consumption. Refer to 26.5.3.3 "GetSupportedFeatures" to determine if specialized resource pool are 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 134 and Figure 135 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 CreateGroupReplica or CreateSynchronizationAspect methods). For variable space consumption, there are no special properties to set by the client.

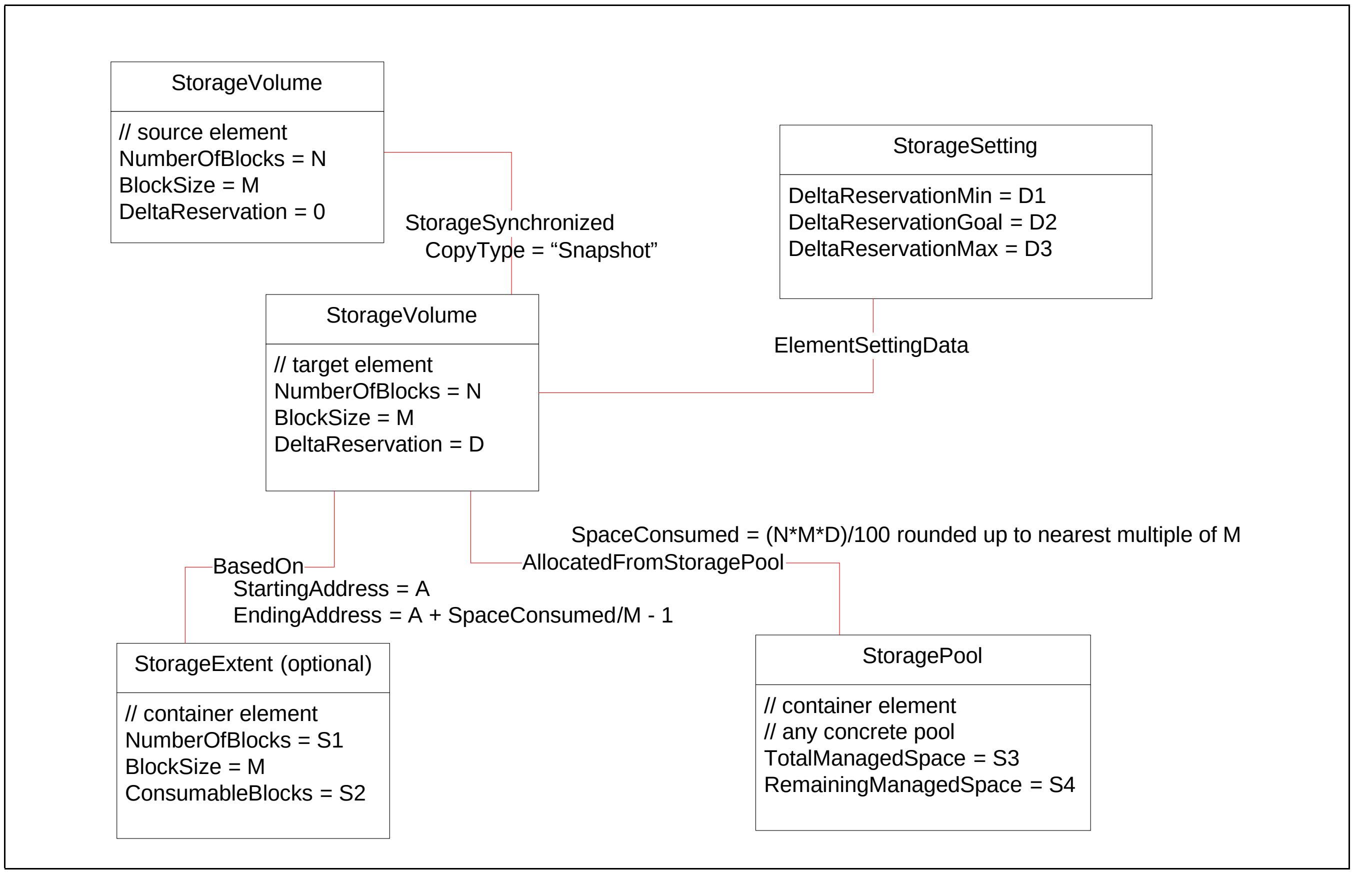

Figure 134 - Fixed Space Consumption

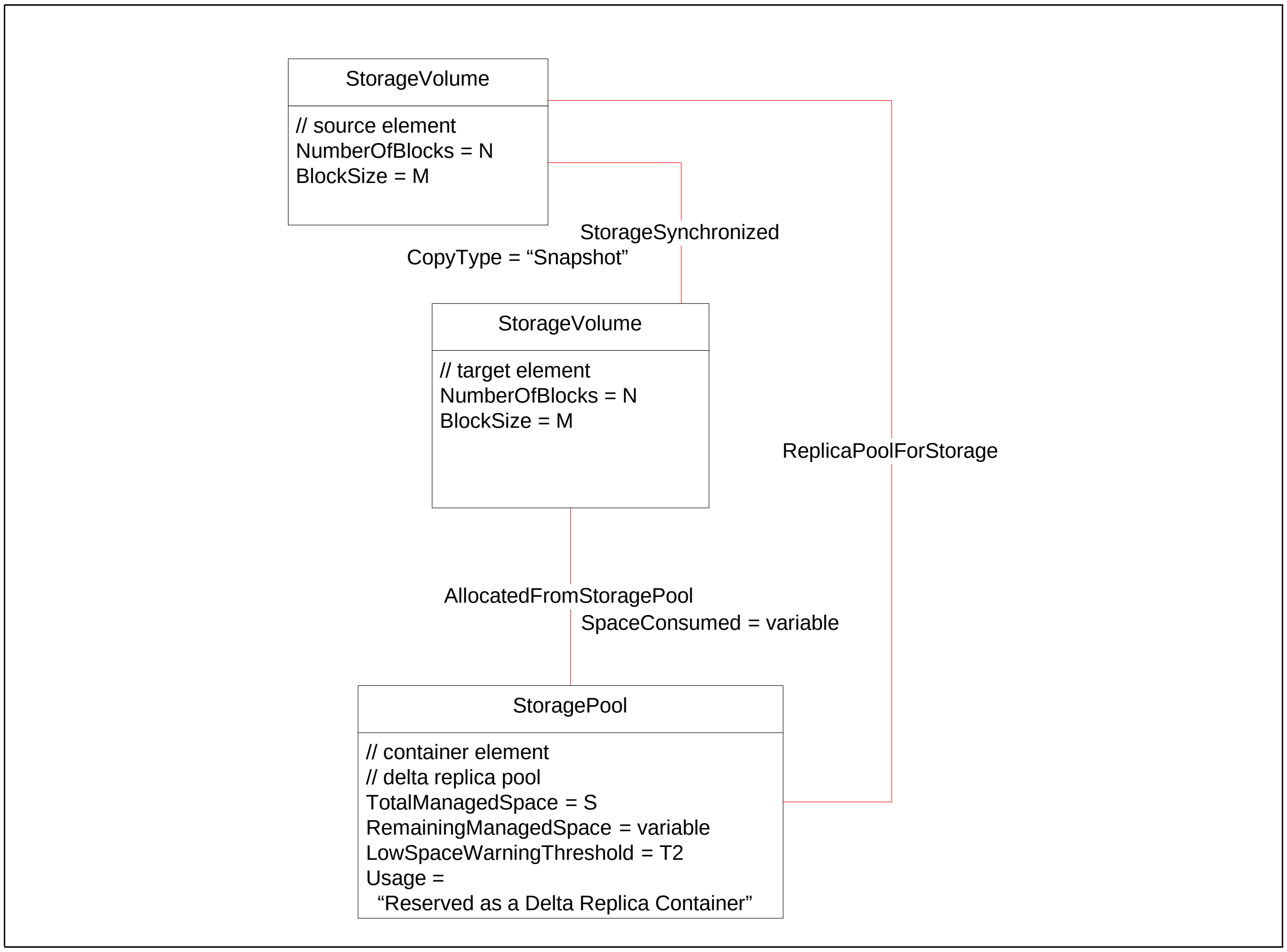

Figure 135 - Variable Space Consumption

26.1.30 Provider Configurations for Remote Replication

Remote replication involves a minimum of two peer system instances. There are two possible provider configurations for controlling remote replication service access points:

Configuration 1: One instance of the provider controls both peers. A client interfaces to one SMI-S server and CIMOM. The only stitching required between arrays is a StorageSynchronized (and GroupSynchronized) association between storage elements in separate arrays.

Configuration 2: A separate instance of the provider controls each peer system. Each provider has its own SMI-S server/CIMOM instance. Clients are required to interact with two providers: the provider controlling the source element and the provider controlling the target element. See the method ReplicationServiceCapabilities.GetSupportedFeatures in 26.5.3.3 "GetSupportedFeatures" for the capability “Remote resource requires remote CIMOM“.

The remote replication model allows connections that are bi-directional or uni-directional. By default, connections to remote systems are bi-directional, unless it is stated otherwise. Refer to 26.5.3.19 "GetSupportedConnectionFeatures".

26.1.31 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 or CreateGroupReplica to request a specific combination. Clients can set the property ReplicationSettingData.ThinProvisioningPolicy for the desired results.

Refer to the capabilities for the allowable combinations supported by the implementation. See 26.5.3.13 "GetSupportedThinProvisioningFeatures", 26.5.3.12 "GetSupportedSettingsDefineStateOperations" and 26.5.3.18 "GetDefaultReplicationSettingData".

26.1.32 Indications

Depending on the implementation, the Replication Services Profile generates a number of different alert and life cycle indicatons, as shown in Table 512. Clients decide what indications they wish to receive by subscribing to the appropriate indications.

Because on a large system with many copy operations in progress simultaneously, there is a potential to receive many unwanted indications. Therefore, it is recommended for the clients to subscribe to indications that have a query that is constrained to a specific replication association. See 26.8 "CIM Elements" for the indication queries. For the storage pool and job indications, refer to Clause 5: "Block Services Package" and *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* Clause 26: "Job Control Subprofile".

Table 512 - Indications

Indication	Source Of
CIM_InstCreation	• New Job Creation • New Target Element Creation • New StorageSynchronized Association Creation • New GroupSynchronized Association Creation
CIM_InstDeletion	• Job Deletion • Target Element Deletion (e.g. Snapshot) • StorageSynchronized Association Deletion • GroupSynchronized Association Deletion
CIM_InstModification	• Job Progress and Status Changes • Source and Target Elements Status Changes • CopyState Changes • ProgressStatus Changes • ProtocolEndpoints and RemoteReplicationCollections Status Changes

Table 512 - Indications

Indication	Source Of
CIM_AlertIndication	• StoragePool space consumption Alerts (especially by Snapshot targets). • Error conditions, such as • StorageSynchronized and GroupSynchronized State set to *Broken*. • ProtocolEndpoints.OperationalStatus set to Error. • RemoteReplicationCollection.ConnectivityStatus set to “down”

26.2 Health and Fault Management Consideration

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
- ProtocolEndpoints
- RemoteReplicationCollections

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 Replication Services Profile generates alert indications that allow monitoring of storage pool consumption by the replica elements.

26.3 Cascading Considerations

For remote replication, the Replication Services Profile requires a cascading provider to perform the “stitching” of resources between the cascading profile (replication services), and a leaf profile (for example the Array Profile), where the remote resources are contained. The cascading provider ensures that the leaf resources represent real instances of ComputerSystem, ProtocolEndpoint, and storage objects such as StorageVolume in the Cascading Profile. Furthermore, the cascading provider shall ensure that state and status properties such as OperationalStatus and CopyState have consistent values between the leaf and real resources.

The replication service relies on other profiles to facilitate access to the leaf resources. For example, the RemoteServiceAccessPoint instance identifies the necessary information to establish access to the leaf system’s resources. See Figure 136 for an instance diagram of establishing access to the leaf resources. This figure also shows instances of additional objects inherited from the class ServiceAccessPoint that can facilitate access to remote resources.

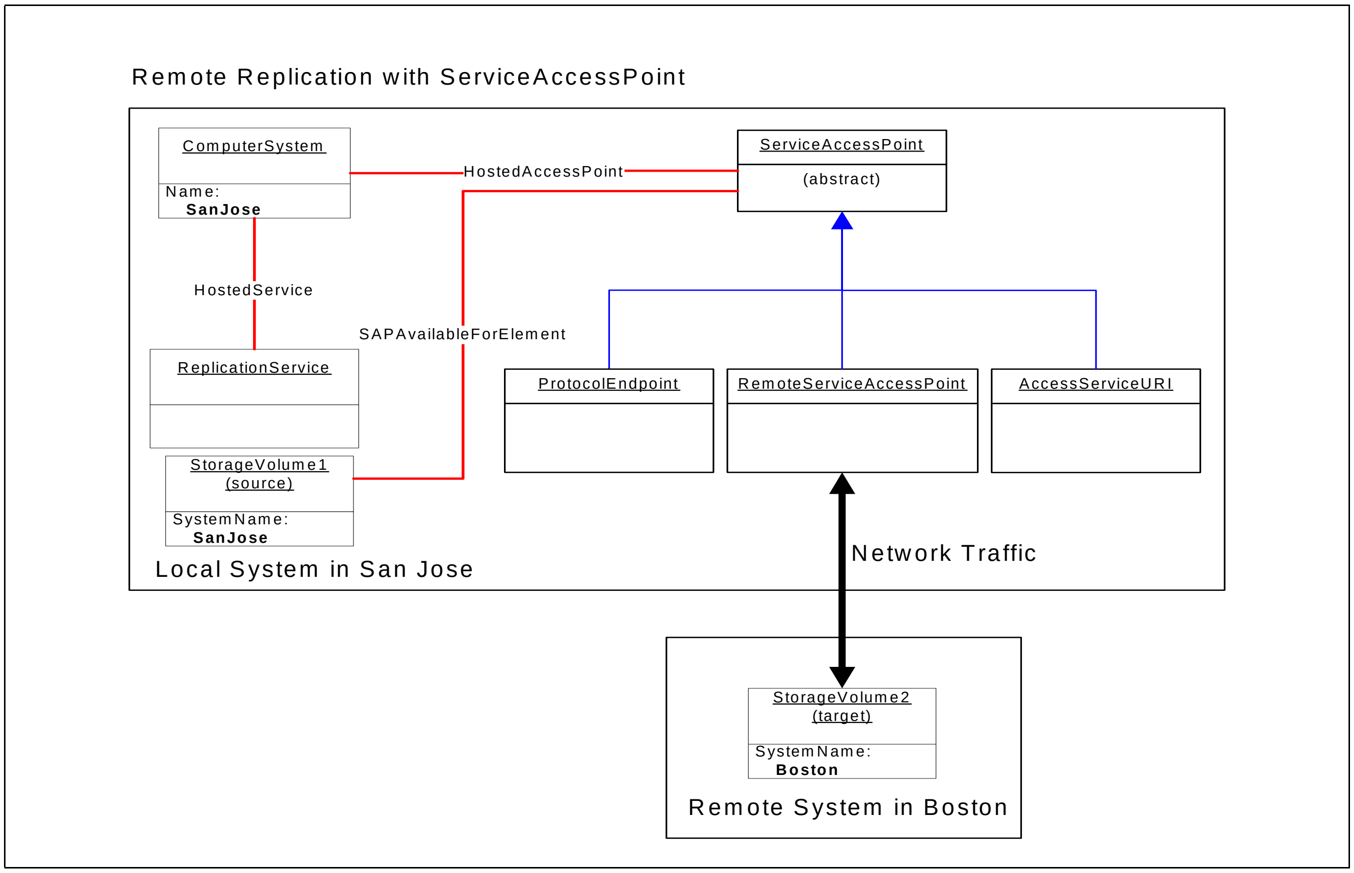

Figure 136 - Instance Diagram for Access to Leaf Resources

26.3.1 ServiceAccessPoint and SharedSecret Instances

Access to remote resources may require an instance of ServiceAccessPoint such as RemoteServiceAccessPoint (inherited from ServiceAccessPoint) and its associated SharedSecret instance, which describes response to a challenge question (i.e., password).

Figure 137 shows an instance of ServiceAccessPoint associated to an instance of SharedSecret via the CredentialContext association.

The method AddServiceAccessPoint (26.5.2.16) and the method AddSharedSecret (26.5.2.17) can be used to create the required instances.

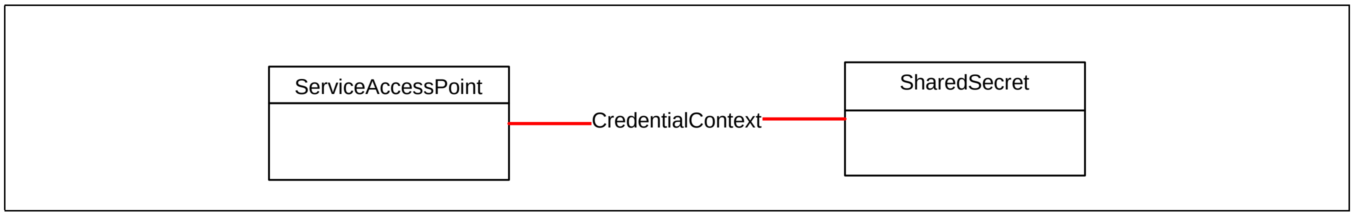

Figure 137 - Instance of ServiceAccessPoint

26.3.2 Cascading Support

Figure 138 illustrates the Replication Services support for cascading.

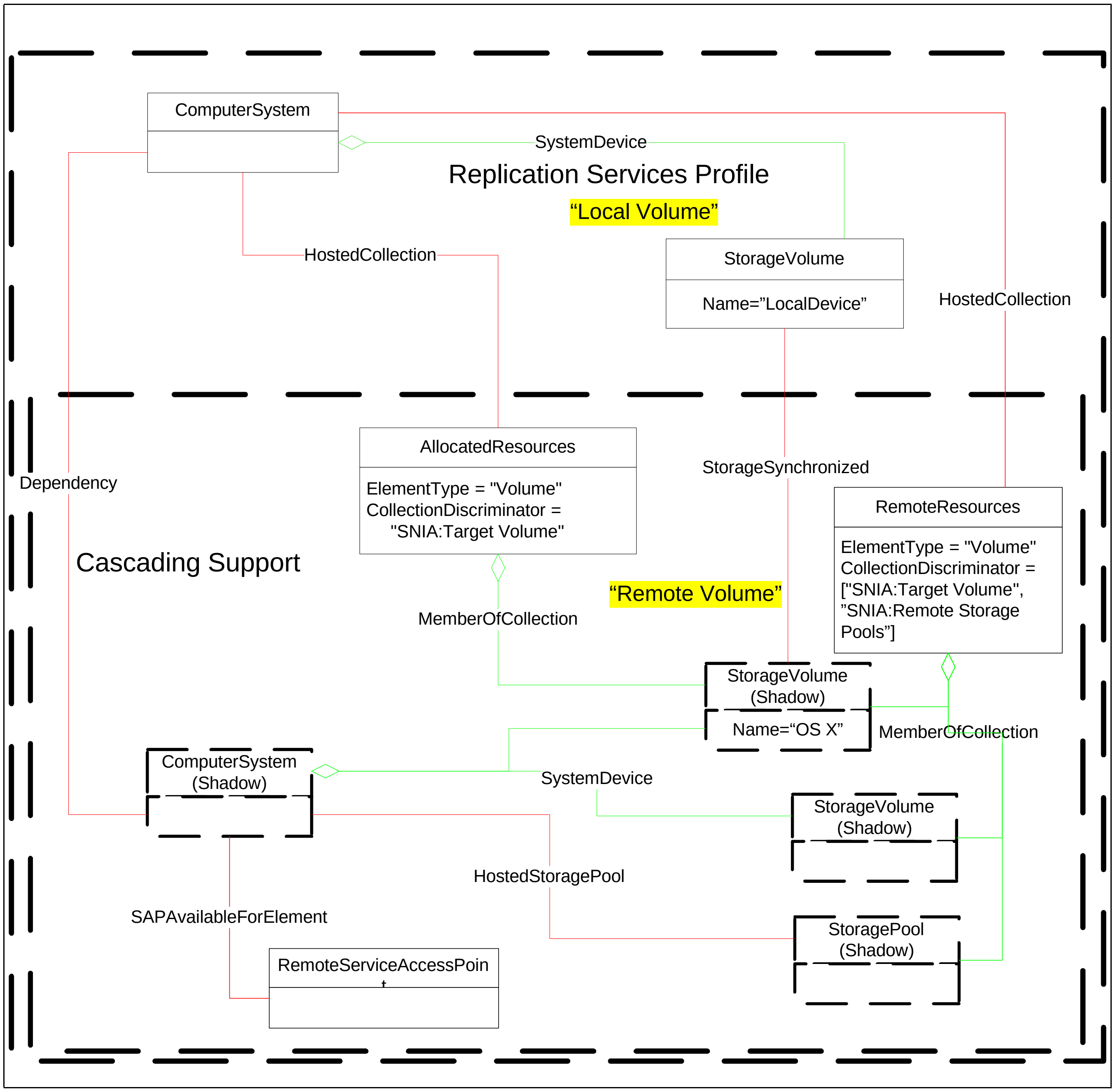

Figure 138 - Replication Services support for Cascading

The embedded dashed box in the figure illustrates the classes and associations of the cascading support. The dashed classes are shadow of instances provided by the remote system. The collection AllocatedResources collects all the components in use by the replication service. The RemoteResources collection collects all components (StorageVolumes, LogicalDisks, StoragePools, etc.) accessible to the replication service (whether used or not).

Figure 139 shows cascading support utilizing replication groups.

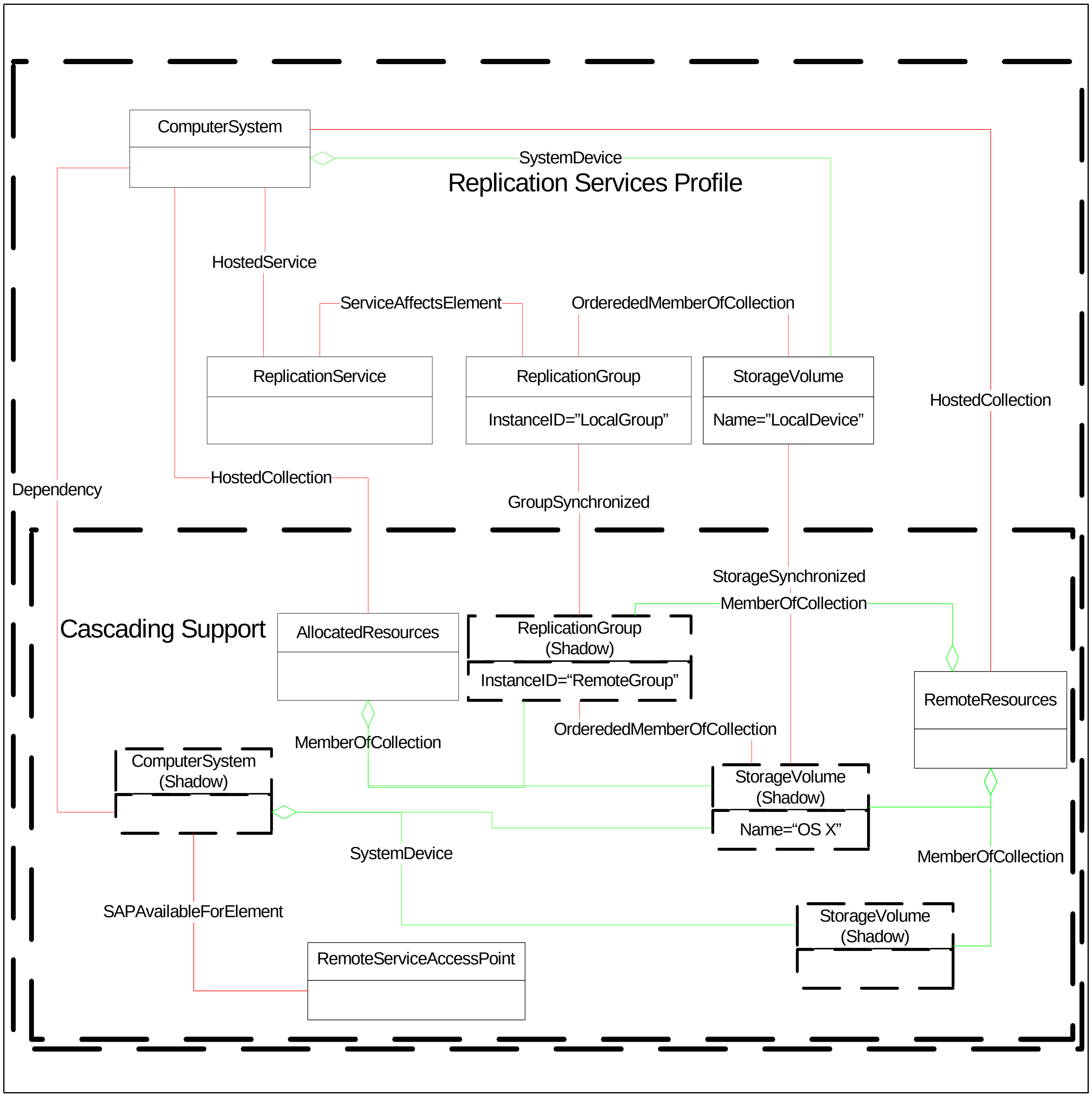

Figure 139 - Cascading and Replication Groups

26.4 Mapping of Copy Services and Replication Services Properties and Methods

Any action taken using the Replication Services methods is reflected, where applicable, appropriately in the properties used by the Copy Services Subprofile (Clause 9: Copy Services Subprofile). The reverse is also true in that any action taken by the Copy Services methods is reflected correctly in the properties used by the Replication Services Profile. Refer to Table 211, “Alignment of SyncType/Mode and CopyType” and Table 212, “Alignment of CopyState and SyncState” for alignment of the specific properties used by Copy Services and this profile.

26.5 Methods of the Profile

The Replication Services Profile has a number of extrinsic methods for group management and replication management. Additionally, there are a number of extrinsic methods in the ReplicationServiceCapabilities that advertise the implemented replication services capabilities. Also, the Profile is dependent on other extrinsic methods provided by the Block Services Package for storage pool and storage element manipulations. Furthermore, the Profile relies on a number of intrinsic methods such as ModifyInstance, DeleteInstance for certain optional 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 513 summarizes the extrinsic methods for group management (class ReplicationService).

Table 513 - Extrinsic Methods for Group Management

Method	Described in
CreateGroup()	See 26.5.1.1
DeleteGroup()	See 26.5.1.2
AddMembers()	See 26.5.1.3
RemoveMembers()	See 26.5.1.4

Table 514 summarizes the extrinsic methods for replication management (class ReplicationService).

Table 514 - Extrinsic Methods for Replication Management

Method	Described in
CreateElementReplica()	See 26.5.2.1
CreateGroupReplica()	See 26.5.2.2
CreateListReplica()	See 26.5.2.3
CreateGroupReplicaFromElements()	See 26.5.2.4
CreateSynchronizationAspect()	See 26.5.2.5
ModifyReplicaSynchronization()	See 26.5.2.6
ModifyListSynchronization()	See 26.5.2.7
ModifySettingsDefineState()	See 26.5.2.8
ModifyListSettingsDefineState()	See 26.5.2.9

Table 514 - Extrinsic Methods for Replication Management

Method	Described in
GetAvailableTargetElements()	See 26.5.2.10
GetPeerSystems()	See 26.5.2.11
GetServiceAccessPoints()	See 26.5.2.14
GetReplicationRelationships()	See 26.5.2.12
GetReplicationRelationshipInstances()	See 26.5.2.13
AddReplicationEntity	See 26.5.2.15
AddServiceAccessPoint	See 26.5.2.16
AddSharedSecret	See 26.5.2.17
CreateRemoteReplicationCollection()	See 26.5.2.18
AddToRemoteReplicationCollection()	See 26.5.2.19
RemoveFromRemoteReplicationCollection()	See 26.5.2.20

Table 515 summarizes the extrinsic methods for examining the implemented capabilities (class **ReplicationServiceCapabilities**). The majority of these methods accept the ReplicationType as an input parameter. The supplied ReplicationType must be a supported replication type corresponding to the property ReplicationServicesCapabilities.SupportedReplicationTypes; otherwise the method returns “Not Supported” (or throws a “Not Supported” exception).

Table 515 - Extrinsic Methods for Getting Supported Capabilities

Method	Described in
ConvertSyncTypeToReplicationType()	See 26.5.3.1
ConvertReplicationTypeToSyncType()	See 26.5.3.2
GetSupportedFeatures()	See 26.5.3.3
GetSupportedGroupFeatures()	See 26.5.3.4
GetSupportedCopyStates()	See 26.5.3.5
GetSupportedGroupCopyStates()	See 26.5.3.6
GetSupportedWaitForCopyStates()	See 26.5.3.7
GetSupportedConsistency()	See 26.5.3.8
GetSupportedOperations()	See 26.5.3.9
GetSupportedGroupOperations()	See 26.5.3.10
GetSupportedListOperations()	See 26.5.3.11
GetSupportedSettingsDefineStateOperations()	See 26.5.3.12
GetSupportedThinProvisioningFeatures()	See 26.5.3.13
GetSupportedMaximum()	See 26.5.3.14

Table 515 - Extrinsic Methods for Getting Supported Capabilities

Method	Described in
GetDefaultConsistency()	See 26.5.3.15
GetDefaultGroupPersistency()	See 26.5.3.16
GetSupportedReplicationSettingData	See 26.5.3.17
GetDefaultReplicationSettingData()	See 26.5.3.18
GetSupportedConnectionFeatures()	See 26.5.3.19

26.5.1 Group Managemenet Methods

26.5.1.1 CreateGroup

```
uint32 ReplicationService.CreateGroup(
        [IN]  string GroupName,
        [IN]  CIM_LogicalElement REF Members[],
        [IN]  boolean Persistent,
        [IN]  boolean DeleteOnEmptyElement,
        [IN]  boolean DeleteOnUnassociated,
        [IN]  CIM_ServiceAccessPoint REF ServiceAccessPoint,
        [OUT] CIM_ReplicationGroup REF ReplicationGroup,
        [IN, EmbeddedInstance ( "CIM_ReplicationSettingData" ) ]
             string ReplicationSettingData );
```

This method allows a client to create a new replication group. Any required associations (such as HostedCollection) are created in addition to the instance of the group. The parameters are as follows:

- GroupName: If nameable, represents a user friendly name for the group being created. If null or not nameable, then the implementation assigns a name.
- Members[]: An array of strings containing object references to the elements to add to the group -- order is maintained. If null, the group will be empty, assuming empty groups are supported. Duplicates members are not allowed.
- Persistent: If true, the group must persist across Provider reboots (group is not temporary). If null, the implementation decides. Use the intrinsic method ModifyInstance to change Persistency of a group if the group persistency is supported by the implementation.
- DeleteOnEmptyElement: If true and empty groups are allowed, the group will be deleted when the last element is removed from the group. If empty groups are not allowed, the group will be deleted automatically when the group becomes empty. If this parameter is not null, its value will be used to set the group's DeleteOnEmptyElement property. Use the intrinsic method ModifyInstance to change this property after the group is created.
- DeleteOnUnassociated: If true, the group will be deleted when the group is no longer associated with another group. This can happen if all synchronization associations to the individual elements of the group are “deleted”. If this parameter is not null, its value will be used to set the group's DeleteOnUnassociated property. Use the intrinsic method ModifyInstance to change this property after the group is created.
- ServiceAccessPoint: Reference to access point information to allow the service to create a group on a remote system. If null, the group is created on the local system.

- ReplicationGroup: If the method completes successfully, then the ReplicationGroup is a reference to the group that is created.
- ReplicationSettingData: If supplied, it provides additional replication settings for the method. For example, to supply the “Description” for the created group.

This method returns the following additional values/statuses:

- If groups are not nameable and a name is supplied, the method returns 7 (“Groups are not nameable“) or throws an appropriate exception.
- If the ServiceAccessPoint is not specified, the replication group is created on the system hosting the replication service, via the HostedService association.

26.5.1.2 DeleteGroup

```
uint32 ReplicationService.DeleteGroup(
        [IN, Required]  CIM_ReplicationGroup REF ReplicationGroup.
        [IN]  CIM_ServiceAccessPoint REF ServiceAccessPoint,
        [IN]  boolean RemoveElements,
        [IN, EmbeddedInstance ( "CIM_ReplicationSettingData" ) ]
             string ReplicationSettingData  );
```

This method allows a client to delete a replication group. All associations to the deleted group are also removed as part of the action. The parameters are as follows:

- ReplicationGroup: This is a reference to the group that the client wants to delete.
- ServiceAccessPoint: Reference to access point information to allow the service to delete the group on a remote system. If null, the group is on the local system.
- RemoveElements: The client can request to delete the group even if it is not empty. If one or more elements in the group are in a replication relationship, RemoteElements is ignored.
- ReplicationSettingData: If supplied, it provides additional replication settings for the method. For example, what should happen OnGroupOrListError.

This method returns the following additional values/statuses:

- If an element in the group is in a replication association, the method returns 7 (“One or more element in a replication relationship“) or throws an appropriate exception.

26.5.1.3 AddMembers

```
uint32 ReplicationService.AddMembers(
        [IN]  CIM_LogicalElement REF Members[],
        [IN, Required]  CIM_ReplicationGroup REF ReplicationGroup,
        [IN]  CIM_ServiceAccessPoint REF ServiceAccessPoint,
        [IN, EmbeddedInstance ( "CIM_ReplicationSettingData" ) ]
             string ReplicationSettingData );
```

This method allows a client to add members to an existing replication group. The parameters are as follows:

- Members[]: An array of strings containing object references to the new elements to add to the replication group. The new elements are added at the end of current members of the replication group. Duplicate members are not allowed.

- ReplicationGroup: A reference to an existing replication group.
- ServiceAccessPoint: Reference to access point information to allow the service to access the group on a remote system. If null, the group is on the local system.
- ReplicationSettingData: If supplied, it provides additional replication settings for the method. For example, what should happen OnGroupOrListError.

26.5.1.4 RemoveMembers

```
uint32 ReplicationService.RemoveMembers(
        [IN]  CIM_LogicalElement REF Members[],
        [IN]  boolean DeleteOnEmptyElement,
        [IN, Required]  CIM_ReplicationGroup REF ReplicationGroup,
        [IN]  CIM_ServiceAccessPoint REF ServiceAccessPoint,
        [IN, EmbeddedInstance ( "CIM_ReplicationSettingData" ) ]
             string ReplicationSettingData );
```

This method allows a client to remove members from an existing replication group. If empty replication groups are not supported by the implementation, deleting all members will delete the group. The parameters are as follows:

- Members[]: An array of strings containing object references to the elements to remove from the replication group. Attempting to remove a member that is not in the replication group, returns an error.
- DeleteOnEmptyElement: If true and removal of the members causes the group to become empty, the group will be deleted. Note, if empty groups are not allowed, the group will be deleted automatically when the group becomes empty. If this parameter is not null, it overrides the group's property DeleteOnEmptyElement.
- ReplicationGroup: A reference to an existing replication group.
- ServiceAccessPoint: Reference to access point information to allow the service to access the group on a remote system. If null, the group is on the local system.
- ReplicationSettingData: If supplied, it provides additional replication settings for the method. For example, what should happen OnGroupOrListError.

This method returns the following additional values/statuses:

- Attempting to remove a group member that is in a replication association, returns 7 (“One or more element in a replication relationship“) or throws an appropriate exception.

26.5.2 Replication Management Methods

26.5.2.1 CreateElementReplica

```
uint32 ReplicationService.CreateElementReplica(
   [IN]  string ElementName,
   [IN, Required]  uint16 SyncType,
   [IN]  uint16 Mode,
   [IN, Required]  CIM_LogicalElement REF SourceElement,
   [IN]  CIM_ServiceAccessPoint REF SourceAccessPoint,
   [IN, OUT]  CIM_LogicalElement REF TargetElement,
   [IN]  CIM_ServiceAccessPoint REF TargetAccessPoint,
   [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.
- SourceAccessPoint: Reference to source access point information. If null, service does not need access information to access the source element.
- 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 26.5.3.17 "GetSupportedReplicationSettingData".
  - 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.
- TargetAccessPoint: Reference to target access point information. If null, service does not need access information to access the target element.
- 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 511 for the CopyStates).

Method Notes:

- Creates a storage element of the same type as the source element.
- If the TargetElement, the TargetPool, or the TargetAccessPoint are not specified, the TargetElement is created on the system hosting the replication service, via the HostedService association. Additionally, when required, the created TargetElement will have the applicable association to the top level ComputerSystem. For example,

if the TargetElement is a StorageVolume, the created TargetElement will have a SystemDevice association to the top level computer system.

- Creates a StorageSynchronized association.
- Creates SystemDevice, AllocatedFromStoragePool, and ElementSettingData associations to the newly created target element.
- May create BasedOn and ReplicaPoolForStorage associations.

Table 516 shows selected optional parameters that can interact.

Table 516 - Selected CreateElementReplica optional parameters

TargetElement	TargetSettingGoal	TargetPool	Comment
Null	Null	Null	Implementation locates/ creates target element*
Supplied	Null	Null	
Null	Supplied	Null	Goal is used to locate/ create target element*
Null	Supplied	Supplied	Goal is used to locate/ create target element* in the supplied Pool
Null	Null	Supplied	Pool is used to locate/ create target element* in Pool. Implementation determines the Goal

Note: * See capabilities (Table 534, “Target Element Suppliers”) for whether implementation locates/creates target elements.

26.5.2.2 CreateGroupReplica

```
uint32 ReplicationService.CreateGroupReplica(
   [IN]   string RelationshipName,
   [IN, Required]   uint16 SyncType,
   [IN]   uint16 Mode,
   [IN]   CIM_ReplicationGroup REF SourceGroup,
   [IN]   CIM_LogicalElement REF SourceElement,
   [IN]  CIM_ServiceAccessPoint REF SourceAccessPoint,
   [IN, OUT]   CIM_ReplicationGroup REF TargetGroup,
   [IN]   uint64 TargetElementCount,
   [IN]   CIM_ServiceAccessPoint REF TargetAccessPoint,
   [IN]   uint16 Consistency,
   [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 group of storage objects which are replicas of the specified source storage or a group of source storage objects (SourceElements). The parameters are as follows:

- RelationshipName: A user relevent name for the relationship between the source and target groups or between a source element and a target group (i.e., one-to-many). If null, the implementation assigns a name. If the individual target elements require an ElementName, a name would be constructed using RelationshipName (or ReplicationSettingData.ElementName) as prefix followed by \"_n\" sequence number, where n is a number beginning with 1.

  If the method is expected to create the target group, and the parameter ReplicationSettingData is supplied, the property ReplicationSettingData.ElementName may be used as the group name.
- SyncType: See CreateElementReplica's parameters (26.5.2.1).
- Mode: See CreateElementReplica's parameters (26.5.2.1).
- SourceGroup: A group of source storage objects which may be a StorageVolume or storage object. If this parameter is not supplied, SourceElement is required. Both SourceGroup and SourceElement shall not be supplied.
- SourceElement: The source storage object which may be a StorageVolume or storage object. If this parameter is not supplied, SourceGroup is required. Both SourceGroup and SourceElement shall not be supplied.
- SourceAccessPoint: Reference to source access point information. If null, service does not need access information to access the source elements/group.
- TargetGroup:
  - As an input, refers to a target group to use.
  - As an output, refers to the created target group (i.e., the replica group). If a job is created, the target group may not be available immediately. If TargetGroup is supplied, TargetElementCount shall be null.
- TargetElementCount: This parameter applies to one-source-to-many-target elements. If TargetGroup is supplied, this parameter shall be null.
- TargetAccessPoint: Reference to target access point information. If null, service does not need access information to access the target elements/group.
- Consistency: This parameter overrides the default group consistency. For example, "No Consistency", "Sequential Consistency".
- ReplicationSettingData: See CreateElementReplica's parameters (26.5.2.1).
- Job: See CreateElementReplica's parameters (26.5.2.1).
- Synchronization: Refers to the created association between the source element (or source replication group) and the target replication group. If a job is created, this parameter may be null, unless the association is actually formed.
- TargetSettingGoal: See CreateElementReplica's parameters (26.5.2.1).
- TargetPool: See CreateElementReplica's parameters (26.5.2.1).
- WaitForCopyState: See CreateElementReplica's parameters (26.5.2.1).

Method Notes:

- Creates storage elements of the same type as the source element(s).
- If the TargetGroup or the TargetAccessPoint are not specified, the TargetGroup is created on the system hosting the replication service, via the HostedService association.
- Creates StorageSynchronized and GroupSynchronized associations.
- Creates SystemDevice, AllocatedFromStoragePool, and ElementSettingData associations to the newly created target elements.
- May create BasedOn and ReplicaPoolForStorage associations.

Table 517 shows selected optional parameters that can interact.

Table 517 - Selected CreateGroupReplica optional parameters

TargetGroup	TargetElementCount	TargetSettingGoal	TargetPool	Comment
Null	Null	Null	Null	Implementation locates/ creates target elements*
Supplied	Null	Null	Null	
Supplied	Supplied	Null	Null	An illegal combination.
Null	Supplied	Null	Null	Implementation locates/ creates target elements*
Null	Supplied	Supplied	Null	Goal is used to locate/ create target elements*
Null	Supplied	Supplied	Supplied	Goal is used to locate/ create target elements* in the supplied Pool
Null	Null	Supplied	Null	Goal is used to locate/ create target elements*
Null	Null	Supplied	Supplied	Goal is used to locate/ create target elements in the supplied Pool
Null	Null	Null	Supplied	Pool is used to locate/ create target elements* in Pool. Implementation determines the Goal

Note: * See capabilities (Table 534, "Target Element Suppliers") for whether implementation locates/creates target elements.

26.5.2.3 CreateListReplica

```
uint32 ReplicationService.CreateListReplica(
   [IN]  string ElementNames[],
   [IN, Required]  uint16 SyncType,
   [IN]  uint16 Mode,
   [IN, Required]  CIM_LogicalElement REF SourceElements[],
   [IN]  CIM_ServiceAccessPoint REF SourceAccessPoint,
   [IN, OUT]  CIM_LogicalElement REF TargetElements[],
   [IN]  CIM_ServiceAccessPoint REF TargetAccessPoint,
   [IN, EmbeddedInstance("CIM_ReplicationSettingData")]
        string ReplicationSettingData,
   [OUT] CIM_ConcreteJob REF Job,
   [OUT] CIM_Synchronized REF Synchronizations[],
   [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) new storage objects which are a replica of the corresponding specified source storage object (an element of the SourceElements). The parameters are as follows:

- ElementNames: An array of end user relevant names for the elements being created. If null, then a system supplied name is used. The value will be stored in the 'ElementName' property for the created element. The first element of the array ElementNames is assigned to the first replica, the second element to the second replica and so on. If there are more SourceElements entries than ElementNames, the system supplied name is used.
- SyncType: Describes the type of copy that will be made. For example, Mirror, Snapshot, and Clone. The same SyncType is applied to all SourceElements entries.
- Mode: Describes whether the target elements will be updated synchronously or asynchronously. The same Mode is applied to all SourceElements entries.
- SourceElements: An array of source storage objects which may be StorageVolumes or storage objects. All the source elements shall be of the same type -- for example, all StorageVolumes.
- SourceAccessPoint: Reference to source access point information. If null, service does not need access information to access the source element. The same SourceAccessPoint applies to all SourceElements entries.
- TargetElements:
  - As an input, refers to an array of target elements to use. If specified, the elements will match one to one with SourceElements[]. If a target elements are not supplied, the implementation may locate or create a suitable target elements. See 26.5.3.17 "GetSupportedReplicationSettingData".
  - As an output, refers to the created target storage elements (i.e., the replicas). If a job is created, the target elements may not be available immediately.
- TargetAccessPoint: Reference to target access point information. If null, service does not need access information to access the target element. The same TargetAccessPoint applies to all TargetElements entries.
- 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. The same ReplicationSettingData applies to SourceElements entries.
- 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).

- Synchronizations: Refers to an array of created associations between the source and the target elements. If a job is created, this parameter may be null, unless the associations are 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. The same TargetSettingGoal applies to all TargetElements entries.
- 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. The same TargetPool applies to all TargetElement entries.
- WaitForCopyState: Before returning, the method shall wait until this CopyState is reached for all Synchronizations. For example, CopyState of Initialized means associations have been established, but there is no data flow. CopyState of Synchronized indicates the replicas are an exact copy of the corresponding source element. CopyState of UnSynchronized means copy operation is in progress (see Table 511 for the CopyStates).

Method Notes:

- Creates a storage elements of the same type as the source elements.
- If the TargetElements, the TargetPool, or the TargetAccessPoint are not specified, the TargetElements are created on the system hosting the replication service, via the HostedService association. Additionally, when required, the created TargetElements will have the applicable associations to the top level ComputerSystem. For example, if the TargetElements are StorageVolumes, the created TargetElements will have SystemDevice associations to the top level computer system.
- Creates the StorageSynchronized associations.
- Creates SystemDevice, AllocatedFromStoragePool, and ElementSettingData associations to the newly created target elements.
- May create BasedOn and ReplicaPoolForStorage associations.

Table 518 shows selected optional parameters that can interact.

Table 518 - Selected CreateListReplica optional parameters

TargetElements	TargetSettingGoal	TargetPool	Comment
Null	Null	Null	Implementation locates/ creates target elements*
Supplied	Null	Null	
Null	Supplied	Null	Goal is used to locate/ create target elements*
Null	Supplied	Supplied	Goal is used to locate/ create target elements* in the supplied Pool
Null	Null	Supplied	Pool is used to locate/ create target elements* in Pool. Implementation determines the Goal

Note: *See capabilities (Table 534, “Target Element Suppliers”) for whether implementation locates/creates target elements.

26.5.2.4 CreateGroupReplicaFromElements

```
uint32 ReplicationService.CreateGroupReplicaFromElements(
   [IN]   string RelationshipName,
   [IN, Required]   uint16 SyncType,
   [IN]   uint16 Mode,
   [IN,OUT]   CIM_ReplicationGroup REF SourceGroup,
   [IN]   CIM_LogicalElement REF SourceElements[],
   [IN, OUT]  string SourceGroupName,
   [IN]  CIM_ServiceAccessPoint REF SourceAccessPoint,
   [IN, OUT]   CIM_ReplicationGroup REF TargetGroup,
   [IN]   CIM_ServiceAccessPoint REF TargetAccessPoint,
   [IN]   uint16 Consistency,
   [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 group of storage objects which are replicas of the specified source storage objects (SourceElements). This method combines the functionality of CreateGroup and CreateGroupReplica in that the methods accepts a list of source elements and creates the source group, and the target group, if not supplied.

The parameter SourceGroupName corresponds to the parameter GroupName as defined in the CreateGroup method.

For the explanation of the parameters, see the methods CreateGroup (26.5.1.1) and CreateGroupReplica (26.5.2.2).

26.5.2.5 CreateSynchronizationAspect

```
uint32 ReplicationService.CreateSynchronizationAspect(
   [IN]   string ElementName,
   [IN, Required]   uint16 SyncType,
   [IN]   uint16 Mode,
   [IN]   CIM_ReplicationGroup REF SourceGroup,
   [IN]   CIM_LogicalElement REF SourceElement,
   [IN]   CIM_ServiceAccessPoint REF SourceAccessPoint,
   [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 (or a group of source elements) 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 CreateElementReplica's parameters (26.5.2.1).
- Mode: See CreateElementReplica's parameters (26.5.2.1).
- SourceGroup: See parameters in 26.5.2.2 "CreateGroupReplica".
- SourceElement: See CreateGroupReplica's parameters (26.5.2.2)
- SourceAccessPoint: Reference to source access point information. If null, service does not need access information to access the source element/group.
- Consistency: See CreateGroupReplica's parameters (26.5.2.2)
- ReplicationSettingData: See CreateElementReplica's parameters (26.5.2.1).
- Job: See CreateElementReplica's parameters (26.5.2.1).
- 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.

26.5.2.6 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 or replication groups. 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/groups relationship that is to be modified.
- ReplicationSettingData: See CreateElementReplica’s parameters (26.5.2.1).
- SyncPair[]: This parameter applies to AddSyncPair/RemoveSyncPair Operations. It allows a client to form a StorageSynchronized association between source and target elements and then add the association to existing source and target groups. Alternatively, a client can remove a StorageSynchronized association from source and target groups.
- Job: See CreateElementReplica’s parameters (26.5.2.1).
- SettingsState: Reference to the association between the source or group 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 CreateElementReplica’s parameters (26.5.2.1).

26.5.2.7 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 or replication groups. 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/groups relationship that is to be modified. All elements of the this array shall of the same concrete class, i.e., StorageSynchronized or GroupSynchronized, and shall have the same SyncType, the same Mode, and the Operation must be valid for the ReplicationType -- SyncType, Mode, Local/Remote.
- ReplicationSettingData: See CreateElementReplica’s parameters (26.5.2.1).
- Job: See CreateElementReplica’s parameters (26.5.2.1).
- 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 CreateElementReplica’s parameters (26.5.2.1). All the supplied synchronization associations must reach at least the specified CopyState before the method returns.

26.5.2.8 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,
[IN]  CIM_ServiceAccessPoint REF TargetAccessPoint,
[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 (or groups).

The parameters are:

- 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. If an associated source element is part of a consistency group, all members of the group shall be paired with the appropriate target elements.
- 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: If TargetGroup is supplied, TargetElement and TargetElementCount shall be null.
  - As an input, refers to a target group to use. If the source has only one element, the presence of a group creates a one-to-many association between the source and the target elements. If TargetGroup is supplied, TargetElement and TargetCount shall be null."
  - As an output, refers to the created target group (i.e., the replica group). If a job is created, the target group may not be available immediately.
- TargetElementCount: This parameter applies to one-source-to-many-target-elements. It is possible to create multiple copies of a source element. If TargetCount is supplied, TargetElement and TargetGroup shall be null.
- TargetAccessPoint: Reference to target access point information. If null, service does not need access information to access the target elements/group.
- Synchronization: The reference to the replication association describing the elements/groups relationship.
- ReplicationSettingData: See CreateElementReplica’s parameters (26.5.2.1).
- Job: See CreateElementReplica’s parameters (26.5.2.1).

- TargetSettingGoal: See CreateElementReplica’s parameters (26.5.2.1).
- TargetPool: See CreateElementReplica’s parameters (26.5.2.1).
- WaitForCopyState: See CreateElementReplica’s parameters (26.5.2.1).

26.5.2.9 ModifyListSettingsDefineState

```
uint32 ReplicationService.ModifyListSettingsDefineState(
  [IN, Required]   uint16 Operation,
  [IN, Required]   CIM_SettingsDefineState REF SettingsStates[],
  [IN, OUT]   CIM_LogicalElement REF TargetElements[],
  [IN, OUT]   CIM_ReplicationGroup REF TargetGroups[],
  [IN]   uint64 TargetElementCount,
  [IN]  CIM_ServiceAccessPoint REF TargetAccessPoint,
  [OUT]  CIM_Synchronized REF Synchronizations[],
  [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 is similar to ReplicationService.ModifySettingsDefineState (26.5.2.8), except that it accepts a list of SettingsDefineState associations.

26.5.2.10 GetAvailableTargetElements

```
uint32 ReplicationService.GetAvailableTargetElements(
   [IN, Required]   CIM_LogicalElement REF SourceElement,
   [IN, Required]   uint16 SyncType,
   [IN, Required]   uint16 Mode,
   [IN, EmbeddedInstance ( "CIM_ReplicationSettingData" )]
       string ReplicationSettingData,
   [IN]   CIM_ComputerSystem REF TargetComputerSystem,
   [IN]   CIM_ServiceAccessPoint REF TargetAccessPoint,
   [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:

- SourceElement: The source storage object which may be a StorageVolume or storage object.
- SyncType: See CreateElementReplica’s parameters (26.5.2.1).
- Mode: See CreateElementReplica’s parameters (26.5.2.1).
- ReplicationSettingData: See CreateElementReplica’s parameters (26.5.2.1). The parameter is useful for requesting a specific combination of thinly and fully provisioned elements.

- TargetComputerSystem: Reference to target computer system. If this parameter and TargetAccessPoint are null, only local targets are returned.
- TargetAccessPoint: Reference to target access point information. If this parameter and TargetComputerSystem are null, only local targets are returned.
- 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 (on the given systems) are examined.
- Job: See CreateElementReplica's parameters (26.5.2.1).
- Candidates[]: The list of the candidate target elements found.

26.5.2.11 GetPeerSystems

```
uint32 ReplicationService.GetPeerSystems(
   [IN]   uint16 Options,
   [OUT]  CIM_ConcreteJob REF Job,
   [OUT]  CIM_ComputerSystem REF Systems[] );
```

This method allows a client to get (or start a job to get) all of the peer systems. A peer system is a system that is known and visible to the Replication Service. Peer systems are discovered through discovery services and/or implementation specific services. If a job is started, once the job completes, examine the AffectedJobElement associations for the peer systems. The parameters are:

- Options: This parameter specifies whether to return all known peer systems or only the systems that are currently reachable. If null, all known systems are returned, whether they are currently reachable or not.
- Job: See CreateElementReplica's parameters (26.5.2.1).
- Systems[]: The list of peer computer systems.

26.5.2.12 GetReplicationRelationships

```
uint32 ReplicationService.GetReplicationRelationships(
   [IN]   uint16 Type,
   [IN]   uint16 SyncType,
   [IN]   uint16 Mode,
   [IN]   uint16 Locality,
   [IN]   uint16 CopyState,
   [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 or GroupSynchronized. If this parameter is not supplied, all such relationships are retrieved.
- SyncType: See CreateElementReplica's parameters (26.5.2.1). If this parameter is not supplied, all SyncTypes are retrieved.
- Mode: See CreateElementReplica's parameters (26.5.2.1). If this parameter is not supplied, all Modes are retrieved.

- Locality: Describes the desired locality. If this parameter is not supplied, all replication relationships are retrieved, regardless of the locality of elements. Choices are: Local only -- Source and target elements are contained in the same system; and Remote only -- Source and target elements are contained in two different systems.
- CopyState: Only retrieve synchronization relationships that currently this CopyState (see Table 511). If this parameter is not supplied, relationships are retrieved regardless of their current CopyState.
- Job: See CreateElementReplica's parameters (26.5.2.1).
- Synchronizations[]: An array of elements found.

26.5.2.13 GetReplicationRelationshipInstances

```
uint32 ReplicationService.GetReplicationRelationshipInstances(
   [IN]   uint16 Type,
   [IN]   uint16 SyncType,
   [IN]   uint16 Mode,
   [IN]   uint16 Locality,
   [IN]   uint16 CopyState,
   [OUT]  CIM_ConcreteJob REF Job,
   [OUT, EmbeddedInstance("CIM_Synchronized")]
     string Synchronizations[] );
```

This method allows a client to get (or start a job to get) all of the synchronization relationship instances known to the processing replication service. If a job is started, once the job completes, examine the AffectedJobElement associations for the synchronization relationships.

The output parameter `Synchronizations` is an array of embedded instances. For the explanation of the remaining parameters, see the method ReplicationService.GetReplicationRelationships (26.5.2.12).

26.5.2.14 GetServiceAccessPoints

```
uint32 ReplicationService.GetServiceAccessPoints(
   [IN]   CIM_ComputerSystem REF System,
   [OUT]  CIM_ConcreteJob REF Job,
   [OUT]  CIM_ServiceAccessPoint REF ServiceAccessPoints[] );
```

This method allows a client to get (or start a job to get) ServiceAccessPoints associated with a peer system. If a job is started, once the job completes, examine the AffectedJobElement associations for the peer system's ServiceAccessPoints. The parameters are as follows:

- System: A reference to the computer system.
- Job: See CreateElementReplica's parameters (26.5.2.1).
- ServiceAccessPoints[]: An array of references to ServiceAccessPoints associated with the supplied system.

26.5.2.15 AddReplicationEntity

```
uint32 ReplicationService.AddReplicationEntity(
      [Required, IN, EmbeddedInstance("CIM_ReplicationEntity")]
         string ReplicationEntity,
      [IN]  boolean Persistent,
      [IN]  string InstanceNamespace,
      [OUT] CIM_ReplicationEntity REF ReplicationEntityPath);
```

This method allows a client to introduce a new instance of ReplicationEntity in the specified Namespace. The parameters are:

- ReplicationEntity: A required parameter containing the information for the ReplicationEntity.
- Persistent: If true, the instance must persist across a Management Server reboot. If null, the value will be based on the default value of the class in the MOF. Use the intrinsic method ModifyInstance to change the Persistency value.
- InstanceNamespace: Namespace of created instance. If null, created instance will be in the same namespace as the service. Namespace must already exist.
- ReplicationEntityPath: A reference to the created instance.

26.5.2.16 AddServiceAccessPoint

```
uint32 ReplicationService.AddServiceAccessPoint(
      [Required, IN, EmbeddedInstance("CIM_ServiceAccessPoint")]
          string ServiceAccessPoint,
      [IN]  string InstanceNamespace,
      [OUT] CIM_ServiceAccessPoint REF ServiceAccessPointPath);
```

This method allows a client to introduce a new instance of ServiceAccessPoint in the specified Namespace. The parameters are:

- ServiceAccessPoint: A required parameter containing the information for the ServiceAccessPoint, or a subclass of the class ServiceAccessPoint, for example, a RemoteServiceAccessPoint.
- InstanceNamespace: Namespace of created instance. If null, created instance will be in the same namespace as the service. Namespace must already exist.
- ServiceAccessPointPath: A reference to the created instance.

26.5.2.17 AddSharedSecret

```
uint32 ReplicationService.AddSharedSecret(
      [Required, IN, EmbeddedInstance("CIM_SharedSecret")]
         string SharedSecret,
      [IN]  CIM_ServiceAccessPoint REF ServiceAccessPoint,
      [IN]  string InstanceNamespace,
      [OUT] CIM_SharedSecret REF SharedSecretPath);
```

This method allows a client to introduce a new instance of SharedSecret in the specified Namespace and optionally associate it to an instance of a ServiceAccessPoint. The parameters are:

- SharedSecret: A required parameter containing the information for the SharedSecret.
- ServiceAccessPoint: Associate created instance to this ServiceAccessPoint. If null, no such association is established.
- InstanceNamespace: Namespace of created instance. If null, created instance will be in the same namespace as the service. Namespace must already exist.
- SharedSecretPath: A reference to the created instance.

26.5.2.18 CreateRemoteReplicationCollection

```
uint32 ReplicationService..CreateRemoteReplicationCollection(
         [IN]   string ElementName,
         [IN]   CIM_ServiceAccessPoint REF LocalAccessPoints[],
         [IN]   CIM_ServiceAccessPoint REF RemoteAccessPoints[],
         [IN]   CIM_ComputerSystem REF RemoteComputerSystem,
         [IN]   boolean Active,
         [IN]   boolean DeleteOnUnassociated,
         [OUT]  CIM_ConcreteJob REF Job,
         [OUT]  CIM_ConnectivityCollection REF ConnectivityCollection  );
```

A method to create (or start a job to create) a new instance of RemoteReplicationCollection, and optionally supply the remote system and the paths (i.e. ProtocolEndpoints) that are used to perform replication operations to/from the remote system. The parameters are:

- ElementName: A end user relevant name for the element being created. If NULL, then a system supplied default name will be used. The value will be stored in the 'ElementName' property for the created element.
- LocalAccessPoints: An array of references to local ServiceAccessPoints (for example, ProtocolEndpoints) that allow communication to the remote system.
- RemoteAccessPoints: An array of references to remote ServiceAccessPoints (for example, ProtocolEndpoints) that allow communication to the remote system.
- RemoteComputerSsystem: A reference to the remote system.
- Active: If true, the instance of RemoteReplicationCollection will be enabled and allows replication operations to to the remote system. Use the intrinsic method ModifyInstance to change this property after the RemoteReplicationCollection is created.
- DeleteOnUnAssociated: If true, the instance of RemoteReplicationCollection will be deleted when it is no longer associated to a ServiceAccessPoint. Use the intrinsic method ModifyInstance to change this property after the RemoteReplicationCollection is created.
- Job: Reference to the job (may be NULL if job is completed) doing the work.
- ConnectivityCollection: Reference to the created instance of RemoteReplicationCollection.

26.5.2.19 AddToRemoteReplicationCollection

```
uint32 ReplicationService.AddToRemoteReplicationCollection(
         [IN]   CIM_ServiceAccessPoint REF LocalAccessPoints[],
         [IN]   CIM_ServiceAccessPoint REF RemoteAccessPoints[],
         [IN]   CIM_ComputerSystem REF RemoteComputerSystem,
         [OUT]  CIM_ConcreteJob REF Job,
         [Required, IN]  CIM_ConnectivityCollection REF ConnectivityCollection  );
```

A method to add (or start a job to add) additional service access points (i.e. ProtocolEndpoints) and/or remote systems associations to an existing instance of RemoteReplicationCollection.

Generally, both AccessPoints and RemoteComputerSystem parameters are supplied to establish the access points to a remote ComputerSystem; however, if parameter AccessPoints is NULL, then only the RemoteComputerSystem is added for the existing AccessPoints associated to the RemoteReplicationCollection. If

RemoteComputerSystem is NULL, then only AccessPoints are added for the existing remote ComputerSystems known to the RemoteReplicationCollection.

See the method CreateRemoteReplicationCollection for description of the parameters.

26.5.2.20 RemoveFromRemoteReplicationCollection

```
uint32 ReplicationService.RemoveFromRemoteReplicationCollection(
      [IN]   CIM_ServiceAccessPoint REF LocalAccessPoints[],
      [IN]   CIM_ServiceAccessPoint REF RemoteAccessPoints[],
      [OUT]  CIM_ConcreteJob REF Job,
      [Required, IN]  CIM_ConnectivityCollection REF ConnectivityCollection );
```

A method to remove (or start a job to remove) service access points (i.e. ProtocolEndpoints) and/or remote systems associations from an existing instance of RemoteReplicationCollection.

Generally, both AccessPoints and RemoteComputerSystem parameters are supplied to remove the access points to a remote ComputerSystem; however, if parameter AccessPoints is NULL, then only the remote ComputerSystem is removed for the existing AccessPoints associated to the RemoteReplicationCollection. If ComputerSystem is NULL, then only AccessPoints are removed from the existing remote ComputerSystems known to the RemoteReplicationCollection. See the method CreateRemoteReplicationCollection for description of the parameters.

26.5.3 Capabilities Methods

26.5.3.1 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 519, Table 520, Table 521, and Table 522 show the values for the CovertSyncTypeToReplicationType parameters. These values also appear in the value maps in the appropriate MOF files.

Table 519 - SyncTypes

SyncType	Value
Mirror	6
Snapshot	7
Clone	8

Table 520 - Modes

Mode	Value
Synchronous	2
Asynchronous	3

Table 521 - Local or Remote

LocalOrRemote	Value
Local	2
Remote	3

Table 522 - 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

26.5.3.2 ConvertReplicationTypeToSyncType

```
uint32 ReplicationServiceCapabilities.ConvertReplicationTypeToSyncType(
     [IN]   uint16 ReplicationType,
     [OUT]  uint16 SyncType,
     [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.

26.5.3.3 GetSupportedFeatures

```
uint32 ReplicationServiceCapabilities.GetSupportedFeatures(
     [IN]   uint16 ReplicationType,
     [OUT]  uint16 Features[] );
```

For a given ReplicationType, this method returns the supported features, as listed in Table 523.

Table 523 - Features

Feature	Description
“Replication Groups”	Elements in a replication group are supported in a replication operation.
"Multi-hop element replication"	A target element can also act as the source for another copy operation.
“Each hop must have same SyncType“	In a multi-hop replication, the new hop must have the same SyncType as the previous hop.
“Multi-hop requires advance notice”	The service needs to know when multi-hoping is intended to allow the service to do the appropriate set up. The parameter ReplicationSettingData specifies the number of hops intended.
"Requires full discovery of target ComputerSystem"	Provider requires the remote ComputerSystems to be discovered. The absence of this capability indicates the service supports *undiscovered resources*.
"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 Replication 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 must have a specific Usage value. For example, reserved for "Local Replica Target" (mirror), reserved for "Delta Replica Target" (Snapshot)., etc.

Table 523 - Features

Feature	Description
"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 = *Broken or Partitioned*), 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.
“Target must remain associated to source”	A dependent target element must remain associated to source element at all times.
"Remote resource requires remote CIMOM"	Client is required to interact with two providers: the provider controlling the source element and the provider controlling the target element.
"Synchronized clone target detaches automatically"	The clone target element detaches automatically when the target element becomes synchronized; otherwise, the client needs to explicitly request a detach operation.
"Reverse Roles operation requires Read Only source"	The “Reverse Roles” operation requires the source element to be in the Read Only mode. To change the protection of an element, see Clause 25: "Storage Element Protection SubProfile".
"Reverse Roles operation requires resync"	After the “Reverse Roles” operation completed, it is required to resync the synchronization relationship between the source and the target elements. This is indicated in the property Synchronized.ProgressStatus - “Requires resync“.
"Restore operation requires fracture"	The “Restore from Replica” operation requires the synchronization relationship to be fractured after restore is completed -- indicated in the property Synchronized.ProgressStatus - “Requires fracture”.
"Resync operation requires activate"	For the copy operation to continue, the synchronization relationship must be activated -- indicated in the property Synchronized.ProgressStatus - “Requires activate”.

Table 523 - Features

Feature	Description
"Copy operation requires offline source"	Instrumentation requires the source element to be offline (not-ready) to ensure data does not change before starting the copy operation.
"Adjustable CopyPriority"	Priority of copy operation versus the host I/O can be adjusted.

26.5.3.4 GetSupportedGroupFeatures

```
uint32 ReplicationServiceCapabilities.GetSupportedGroupFeatures(
  [IN]   uint16 ReplicationType,
  [OUT]  uint16 GroupFeatures[] );
```

For a given ReplicationType, this method returns the supported replication group features, as listed in Table 524.

Table 524 - Group Features

GroupFeatures	Description
"One-to-many replication"	One source element can be copied to multiple target elements in a group.
“Many-to-many replication“	One or more elements in the source group and one or more elements in the target group.
“Consistency enabled for all groups”	By default, all groups are *Consistent*
“Empty replication groups allowed”	It is possible to have a replication group with no members; otherwise, an empty group gets deleted automatically.
"Source group must have more than one element"	One members replication groups are not supported.
"Composite Groups"	A replication group can have members from different ComputerSystems.
"Multi-hop group replication"	A target replication group can also act as a source for another copy operation.
“Each hop must have same SyncType”	The SyncType of each hop must be the same, e.g., mirror, snapshot, clone.
"Group can only have one single relationship active"	At any given time, only one relationship in the source group can be active.
“Source element can be removed from group”	A source element can be removed even when the group is associated with another replication group.
“Target element can be removed from group”	A target element can be removed even when the group is associated with another replication group.
"Group can persist"	The replication group can persist across the Provider reboot (group is not temporary).
"Group is nameable"	A user friendly name can be given to a replication group (ElementName)

Table 524 - Group Features

GroupFeatures	Description
"Supports target element count"	It is possible to supply one source element and request more than one target element copies.
"Synchronized clone target detaches automatically"	The clone target element detaches automatically when the target element becomes synchronized; otherwise, the client needs to explicitly request a detach operation.
"Reverse Roles operation requires Read Only source"	The “Reverse Roles” operation requires the source element to be in the Read Only mode. To change the protection of an element, see Clause 25: "Storage Element Protection SubProfile".
"Reverse Roles operation requires resync"	**After** the “Reverse Roles” operation completed, it is required to resync the synchronization relationship between the source and the target elements. This is indicated in the property Synchronized.ProgressStatus - “Requires resync“.
"Restore operation requires fracture"	The “Restore from Replica” operation requires the synchronization relationship to be fractured **after** restore is completed -- indicated in the property Synchronized.ProgressStatus - “Requires fracture”.
"Resync operation requires activate"	For the copy operation to continue, the synchronization relationship must be activated -- indicated in the property Synchronized.ProgressStatus - “Requires activate”.
"Copy operation requires offline source"	Instrumentation requires the source element to be offline (not-ready) to ensure data does not change before starting the copy operation.

26.5.3.5 GetSupportedCopyStates

```
uint32 ReplicationServiceCapabilities.GetSupportedCopyStates(
     [IN]   uint16 ReplicationType,
     [OUT]  uint16 SupportedCopyStates[],
     [OUT]  boolean HostAccessible[] );
```

For a given ReplicationType, this method returns the supported CopyStates (see Table 511) and a parallel array to indicate whether for a given CopyState the target element is host accessible or not (true or false).

26.5.3.6 GetSupportedGroupCopyStates

```
uint32 ReplicationServiceCapabilities.GetSupportedGroupCopyStates(
     [IN]   uint16 ReplicationType,
     [OUT]  uint16 SupportedCopyStates[] );
```

For a given ReplicationType, this method returns the supported replication group CopyStates (see Table 511).

26.5.3.7 GetSupportedWaitForCopyStates

```
   uint32 ReplicationServiceCapabilities.GetSupportedWaitForCopyStates(
     [IN]   uint16 ReplicationType,
     [IN]   unit16 MethodName,
     [OUT]  uint16 SupportedCopyStates[] );
```

This method, for a given ReplicationType and method, returns the supported CopyStates that can be specified in the method's WaitForCopyState parameter.

26.5.3.8 GetSupportedConsistency

```
  uint32 ReplicationServiceCapabilities.GetSupportedConsistency(
    [IN]   uint16 ReplicationType,
    [OUT]  uint16 SupportedConsistency[] );
```

For a given ReplicationType, this method returns the supported Consistency, as listed in Table 525.

Table 525 - Consistency

Consistency	Description
“Sequentially Consistent”	Provider guarantees ordered write consistency.

26.5.3.9 GetSupportedOperations

```
  uint32 ReplicationServiceCapabilities.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. Table 526 shows the possible Operations that an implementation may support.

Refer to Figure 55, “CopyState Transitions” for additional information.

Table 526 - Operations

Operation	Description	Special Consideration
"Abort"	Abort the copy operation if it is possible.	
"Activate Consistency"	Enable consistency.	
“Activate”	Activate an “Inactive” or “Prepared” StorageSynchronized association.	
"AddSyncPair"	Add source and target elements of a StorageSynchronized association to the source and target replication groups. The SyncType of the associations must be the same.	
"Deactivate Consistency"	Disable consistency.	
“Deactivate”	Stop the copy operation. Writes to source element are allowed.	Snapshot: Writes to target element after point-in-time is created are lost (pointers removed).

Table 526 - Operations

Operation	Description	Special Consideration
"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.	
"RemoveSyncPair"	Remove the elements associated via the StorageSynchronized association from the source and the target groups.	
"Resync Replica"	Resynchronize a fractured target element. Or, from a Broken or Aborted relationship.	To continue from the *Broken* state, the problem should be corrected first before resyncing the replica. Also, to continue from the *Aborted* state.
"Restore from Replica"	Copy target element to the source element.	To ensure integrity of data, restoring to a source element which is the source of multiple copy operations, the implementation may impose additional restrictions ranging from not supporting the restore operation to such a source element to preventing multiple restore operations simultaneously. Also, after the operation is completed, it may be necessary to fracture the synchronization relationship. See GetSupportedFeatures in capabilities.

Table 526 - Operations

Operation	Description	Special Consideration
"Resume"	Continue the copy operation of a suspended relationship.	
"Reset To Sync"	Change Mode to Synchronous.	
"Reset To Async"	Change Mode to Asynchronous.	
“Return To ResourcePool”	Delete a Snapshot target.	
"Reverse Roles"	Switch the source and the target element roles.	The source element may need to be Read Only. See GetSupportedFeatures in capabilities.
"Split"	Separate the source and the target elements in a *consistent* manner.	
"Suspend"	Stop the copy operation in such a way that it can be resumed.	
“Unprepare”	Causes the synchronization to be reinitialized and stop in Prepared state.	

Table 527 compares the action of similar Operations.

Table 527 - 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 operation. 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 operation. 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.

Table 527 - Comparison of Similar Operations

Operations	Description
Detach versus Dissolve	Detach: The association between the source and target element must 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 (26.5.3.3) Capabilities.
Unsynchronized versus Skewed	Unsynchronized: The source element contains data that has not been copied to the target element. Most likely, the copy operation is in the process of updating the target element (ProgressStatus=Synchronizing). Skewed: The target element has been updated by a host (e.g. target of a snapshot). Resynchronization is not automatic and requires an explicit "Resync" operation (i.e., ModifySynchronization)

26.5.3.10 GetSupportedGroupOperations

```
uint32 ReplicationServiceCapabilities.GetSupportedGroupOperations(
      [IN]   uint16 ReplicationType,
      [OUT]  uint16 SupportedOperations[] );
```

For a given ReplicationType this method returns the supported replication group Operations (see Table 526) on a GroupSynchronized association that can be supplied to the ModifyReplicaSynchronization method.

26.5.3.11 GetSupportedListOperations

```
uint32 ReplicationServiceCapabilities.GetSupportedListOperations(
      [IN]   uint16 ReplicationType,
      [IN]   uint16 SynchronizationType,
      [OUT]  uint16 SupportedOperations[] );
```

For a given ReplicationType this method returns the supported replication Operations (see Table 526) on a list of associations that can be supplied to the ModifyListSynchronization method. The parameter SynchronizationType specifies the operations as they apply to a list of StorageSynchronized or GroupSynchronized. If SynchronizationType is not specified, StorageSynchronized is assumed.

26.5.3.12 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. Table 528 shows the list of SettingsDefineState operations that an implementation may support.

Table 528 - 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 and GroupSynchronized.	

26.5.3.13 GetSupportedThinProvisioningFeatures

```
uint32 ReplicationServiceCapabilities.GetSupportedThinProvisioningFeatures(
      [IN]   uint16 ReplicationType,
      [OUT]   uint16 SupportedThinProvisioningFeatures[] );
```

For a given ReplicationType this method returns the supported features related to thin provisioning. Table 529 shows the list of the Thin Provisioning Features an implementation may support.

A client can request a specific thin provisioning policy in the ReplicationSettingData parameter of the appropriate method call. See the property ReplicationSettingData.ThinProvisioningPolicy for the supported options for a copy operation.

Table 529 - 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. The implementation then writes zeros in the unused blocks of the target element.
"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.

26.5.3.14 GetSupportedMaximum

```
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 530 shows the list of components that can be specified.

Table 530 - Components

Component	Description
“Number of groups”	Maximum number of groups supported by the replication service.
"Number of elements per source group"	Maximum number of elements in a group that can be used as a source group.
"Number of elements per target group"	Maximum number of elements in a group that can be used as a target group.
"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 peer systems"	Maximum number of peer systems that replication service can communicate with.
"Number of hops in multi-hop replication"	Maximum number of hops in multi-hop replication the service can manage.

26.5.3.15 GetDefaultConsistency

```
uint32 ReplicationServiceCapabilities.GetDefaultConsistency(
      [IN]   uint16 ReplicationType,
      [OUT]  uint16 DefaultConsistency );
```

This method for a given ReplicationType, returns the default consistency value for the replication groups. Table 531 shows the list of possible Default Consistency values that an implementation may offer.

Table 531 - Default Consistency

DefaultConsistency	Description
"No default consistency"	Replication groups are not declared as consistent.
"Sequentially Consistent"	By default, a newly created replication group is declared to be consistent.

26.5.3.16 GetDefaultGroupPersistency

```
uint32 ReplicationServiceCapabilities.GetDefaultGroupPersistency(
     [OUT]  uint16 DefaultGroupPersistency );
```

This method returns the default persistency for a newly created group. Table 532 shows the list of possible Group Persistency values that an implementation may offer.

Table 532 - Group Persistency

DefaultGroupPersistency	Description
"No default persistency"	Replication groups are not declared as persistent across the Provider reboots.
"Persistent"	By default, a newly created replication group is declared to be persistent across the Provider reboot (group is not temporary).

26.5.3.17 GetSupportedReplicationSettingData

```
uint32 ReplicationServiceCapabilities.GetSupportedReplicationSettingData(
     [IN]   uint16 ReplicationType,
     [IN]   uint16 PropertyName,
     [OUT]  uint16 SupportedValues[] );
```

This method, for a given ReplicationType, returns an array of supported settings that can be utilized in an instance of the ReplicationSettingData class. See the MOF for the ReplicationSettingData class for the value map of the properties. Explanation of some of the properties appears below.

Table 533 shows the values for the property ReplicationSettingData.CopyMethodology.

Table 533 - Copy Methodologies

CopyMethodology	Description
"Other"	A methodology not listed in this table.
"Implementation decides"	Implementation determines a suitable methodology.
"Full-Copy"	All data is copied to the target element.
"Incremental-Copy"	Only changed data is copied to the target element.

Table 533 - Copy Methodologies

CopyMethodology	Description
"Differential-Copy"	Only the new writes are copied to the target element.
"Copy-On-Write"	Affected data is copied on the first write to the source or to the target elements.
"Copy-On-Access"	Affected data is copied on the first access to the source element.
“Delta-Update”	Difference based replication where initially the source element is copied to the target element. Then, at regular intervals, only changes to the source element that have taken place since the previous copy operation are incrementally updated to the target element. This copy operation is also referred to as asynchronous mirroring.
“Snap-And-Clone“	The service creates a snapshot of the source element first, then uses the snapshot as the source of the copy operation to the target element.

Table 534 shows the values for the property ReplicationSettingData.TargetElementSuppliers.

Table 534 - Target Element Suppliers

TargetElementSupplier	Description
“Use existing”	Use existing elements only. If appropriate elements are not available, returns an error.
“Create new”	Create new target elements only.
“Use and create“	If appropriate elements are not available, create new target elements.
“Instrumentation decides“	
“Client must supply”	Client must supply target elements.

Table 535 shows the values for the property ReplicationSettingData.ThinProvisioningPolicy.

Table 535 - ThinProvisioningPolicy

Feature	Description
"Copy thin source to thin target"	Thinly provisioned source element is copied to a thinly provisioned target element.
"Copy thin source to full target"	Thinly provisioned source element is copied to a fully provisioned target element.
"Copy full source to thin target"	Fully provisioned source element is copied to a thinly provisioned target element.

Table 535 - ThinProvisioningPolicy

Feature	Description
"Provisioning of target same as source"	Provisioning of the target element is the same as the provisioning of the source element.
"Target pool decides provisioning of target element"	In the call to the CreateElementReplica or CreateGroupReplica method, the storage pool for the target elements is supplied. The supplied storage pool decides the provisioning of the created target elements.
"Implementation decides provisioning of target"	Vendor specific.

26.5.3.18 GetDefaultReplicationSettingData

```
uint32 ReplicationServiceCapabilities.GetDefaultReplicationSettingData(
     [IN]   uint16 ReplicationType,
     [OUT, EmbeddedObject]
          string DefaultInstance );
```

This method, for a given ReplicationType, returns the default ReplicationSettingData as an instance. Use this method to determine the implementation behavior for replication settings that do not have a distinct capability method.

26.5.3.19 GetSupportedConnectionFeatures

```
uint32 ReplicationServiceCapabilities.GetSupportedConnectionFeatures(
      [IN]   CIM_ProtocolEndpoint REF connection,
      [OUT]  uint16 SupporteConnectionFeatures[] );
```

This method accepts a connection reference and returns specific features of that connection. Table 536 shows the list of possible Connection Features that an implementation may support.

Table 536 - Connection Features

ConnectionFeature	
"Unidirectional to ProtocolEndpoint"	Direction of data flow to this ProtocolEndpoint, from a remote system (by default the connection is bi-directional).
"Unidirectional from ProtocolEndpoint"	Direction of data flow from this ProtocolEndpoint to a remote system (by default the connection is bi-directional).

26.5.4 Replication Services and Copy Services Properties and Methods Mapping

To preserve backward compatibility, a few additional properties in the existing classes are introduced instead of changing the semantics of the existing properties. Any action taken by a Replication Services client shall be reflected correctly in the applicable properties visible to a Copy Services client. The reverse is also true in that any action taken by a Copy Services client shall be reflected correctly in the properties visible to a Replication Services client. Keep in mind certain requests that are not supported by Copy Services result in the request failing. For example, passing an instance of StorageSynchronized that contains a remote SyncedElement reference to the Copy Services’ ModifySynchronization method will generate an error.

26.5.4.1 Properties Mapping

See 9.1.6.1.1 "Alignment of StorageSynchronized Properties" to determine the alignment between CopyType and SyncState (from Copy Services) and SyncType, Mode, CopyState, and ProgressStatus (from Replication Services).

26.5.4.2 Method Mapping

Table 537, “Copy Services and Replication Services Methods Mapping” summarizes the method mapping between Copy Services and Replication Services Profiles. Again, use the Replication Services for extended functionality, such as Thin Provisioning.

Table 537 - Copy Services and Replication Services Methods Mapping

Copy Services Method	Corresponding Replication Services Method
CreateReplica()	CreateElementReplica()
AttachReplica()	
ModifySynchronization()	ModifyReplicaSynchronization()
	ModifyListSynchronization()

For description of the Copy Services Methods, see 9.5 "Methods of the Profile".

26.6 Client Considerations and Recipes

26.6.1 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 26.1.9 "SyncTypes".
- Locate the hosted instance of ReplicationService. See 26.1.7.
- 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 26.1.8 "Replication Services Capabilities".
- 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 26.1.28.
- Verify StoragePools have sufficient free capacity for the target elements. See 26.1.29.
- If necessary, use the ReplicationService’s group manipulation methods to create and populate source and target groups. See 26.5 "Methods of the Profile".
- Invoke the appropriate extrinsic method of the ReplicationService to create a replica. See 26.5 "Methods of the Profile".
- 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 26.1.16 "Associations" and 26.1.32 "Indications".
- Invoke the method ReplicationService.ModifyReplicaSynchronization to modify a replica. For example, “split” a replica from its source element. See 26.5 "Methods of the Profile".

26.7 Registered Name and Version

Replication Services version 1.6.0 (Component Profile)

26.8 CIM Elements

Table 538 describes the CIM elements for Replication Services.

Table 538 - CIM Elements for Replication Services

Element Name	Requirement	Description
26.8.1 CIM_ElementCapabilities	Mandatory	Associates StorageReplicationCapabilities and ReplicationService.
26.8.2 CIM_GroupSynchronized	Conditional	Experimental. Conditional requirement: Required if groups are supported. Associates source and target groups, or a source element to a target group.
26.8.3 CIM_HostedAccessPoint (ForProtocolEndpoint)	Conditional	Conditional requirement: Required if remote replication is supported. Associates ProtocolEndpoint to the ComputerSystem on which it is hosted.
26.8.4 CIM_HostedAccessPoint (ForRemoteServiceAccessPoint)	Conditional	Conditional requirement: Required if remote replication is supported. Associates RemoteServiceAccessPoint to the ComputerSystem.
26.8.5 CIM_HostedCollection (Allocated Resources)	Mandatory	This would associate the AllocatedResources collection to the top level system for the Replication Services Profile using Cascading.
26.8.6 CIM_HostedCollection (Between ComputerSystem and RemoteReplicationCollection)	Conditional	Conditional requirement: Required if remote replication is supported. Associates the RemoteReplicationCollection (ConnectivityCollection) to the hosting System.
26.8.7 CIM_HostedCollection (Between ComputerSystem and ReplicationGroup)	Conditional	Conditional requirement: Required if groups are supported. Associates the replication group to the hosting System.
26.8.8 CIM_HostedCollection (Remote Resources)	Conditional	Conditional requirement: This is required if SNIA_RemoteResources is modeled. This would associate the RemoteResources collection to the top level system for the Replication Services Profile in support of Cascading.
26.8.9 CIM_HostedService	Mandatory	
26.8.10 CIM_MemberOfCollection (Allocated Resources)	Mandatory	This supports collecting replication components. This is required to support the AllocatedResources collection for Cascading.

Table 538 - CIM Elements for Replication Services

Element Name	Requirement	Description
26.8.11 CIM_MemberOfCollection (ProtocolEndpoints to RemoteReplicationCollection)	Conditional	Conditional requirement: Required if remote replication is supported. Associates ProtocolEndpoints to RemoteReplicationCollection (ConnectivityCollection).
26.8.12 CIM_MemberOfCollection (Remote Resources)	Optional	This supports collecting all Shadow instances of components that the Replication Service has available to use. This is optional when used to support the RemoteResources collection (the RemoteResources collection is optional).
26.8.13 CIM_OrderedMemberOfCollection	Conditional	Conditional requirement: Required if groups are supported. Associates ReplicationGroup to storage elements.
26.8.14 CIM_ProtocolEndpoint	Conditional	Conditional requirement: Required if remote replication is supported. Special purpose endpoint that represents connections between systems.
26.8.15 CIM_RemoteReplicationCollection	Conditional	Conditional requirement: Required if remote replication is supported. A RemoteReplicationCollection groups together a set of ProtocolEndpoints of the same 'type' (i.e., class) which are able to communicate with each other. The ProtocolEndpoints are used by Replication Services.
26.8.16 CIM_RemoteServiceAccessPoint	Conditional	Conditional requirement: Required if remote replication is supported. A ServiceAccessPoint for replication service.
26.8.17 CIM_ReplicaPoolForStorage	Optional	Associates special storage pool for Snapshots (delta replicas) to a source element.
26.8.18 CIM_ReplicationEntity	Optional	Represents a replication entity such as an entity known by its World Wide Name (WWN).
26.8.19 CIM_ReplicationGroup	Conditional	Experimental. Conditional requirement: Required if groups are supported. Represents a group of elements participating in a replication activity.
26.8.20 CIM_ReplicationService	Mandatory	Experimental. Base class for Replication Services. Methods are described in the Extrinsic Methods clause.
26.8.21 CIM_ReplicationServiceCapabilities	Mandatory	Experimental. A set of properties and methods that describe the capabilities of a replication services provider.
26.8.22 CIM_ReplicationSettingData	Optional	Experimental. Contains special options for use by methods of Replication Services.

Table 538 - CIM Elements for Replication Services

Element Name	Requirement	Description
26.8.23 CIM_SAPAvailableForElement	Conditional	Conditional requirement: Required if remote replication is supported. This association identifies the element that is serviced by the ServiceAccessPoint.
26.8.24 CIM_ServiceAffectsElement (Between ReplicationService and RemoteReplicationCollection)	Conditional	Conditional requirement: Required if remote replication is supported. Associates Replication Service to RemoteReplicationCollection (ConnectivityCollection).
26.8.25 CIM_ServiceAffectsElement (Between ReplicationService and ReplicationEntity)	Optional	Associates Replication Service to ReplicationEntity.
26.8.26 CIM_ServiceAffectsElement (Between ReplicationService and ReplicationGroup)	Conditional	Conditional requirement: Required if groups are supported. Associates Replication Service to Replication Group.
26.8.27 CIM_SettingsDefineState (Between ReplicationGroup and SynchronizationAspect)	Optional	Associates a replication group to an instance of SynchronizationAspect.
26.8.28 CIM_SettingsDefineState (Between storage object and SynchronizationAspect)	Optional	Associates a storage object to an instance of SynchronizationAspect.
26.8.29 CIM_SharedSecret	Conditional	Conditional requirement: Required if remote replication is supported.
26.8.30 CIM_StorageSynchronized	Mandatory	Experimental. Associates replica target element to source element. Property definitions and descriptions are identical to those for LogicalDisk usage.
26.8.31 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.
26.8.32 SNIA_AllocatedResources	Optional	This is a SystemSpecificCollection for collecting components that are being used by the Replication Services profile (e.g., StorageVolumes, LogicalDisks, etc.) that supports Cascading.
26.8.33 SNIA_RemoteResources	Optional	This is a SystemSpecificCollection for collecting components that may be allocated by the Replication Services profile (e.g., StorageVolume) that supports Cascading.
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_StorageSynchronized	Mandatory	All instance creation indications for StorageSynchronized.

Table 538 - CIM Elements for Replication Services

Element Name	Requirement	Description
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_GroupSynchronized	Conditional	Conditional requirement: Required if groups are supported. All instance creation indications for GroupSynchronized.
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_SynchronizationAspect	Optional	All instance creation indications for SynchronizationAspect.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_StorageSynchronized AND OBJECTPATH(SourceInstanceModelpath) = OBJECTPATH('string-key-of-storage-synchronized')	Conditional	Conditional requirement: Required if semi-fixed indication filters are supported. CQL -Instance deletion indications for a specific StorageSynchronized.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_StorageSynchronized	Optional	All instance deletion indications for StorageSynchronized.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_GroupSynchronized AND OBJECTPATH(SourceInstanceModelpath) = OBJECTPATH('string-key-of-group-synchronized')	Conditional	Conditional requirement: Required if groups and semi-fixed indication filters are supported. CQL -Instance deletion indications for a specific GroupSynchronized.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_GroupSynchronized	Optional	All instance deletion indications for GroupSynchronized.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_SynchronizationAspect	Optional	All instance deletion indications for SynchronizationAspect.
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')	Conditional	Conditional requirement: Required if semi-fixed indication filters are supported. CQL -Synchronization state transition for a specific replica association.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_StorageSynchronized AND SourceInstance.CIM_StorageSynchronized::CopyState <> PreviousInstance.CIM_StorageSynchronized::CopyState	Optional	CQL -Synchronization state transition for replica associations.

Table 538 - CIM Elements for Replication Services

Element Name	Requirement	Description
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_StorageSynchronized AND SourceInstance.CIM_StorageSynchronized::ProgressStatus <> PreviousInstance.CIM_StorageSynchronized::ProgressStatus AND OBJECTPATH(SourceInstanceModelpath) = OBJECTPATH('string-key-of-storage-synchronized')	Optional	CQL -Progress status transition for a specific replica association.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_StorageSynchronized AND SourceInstance.CIM_StorageSynchronized::ProgressStatus <> PreviousInstance.CIM_StorageSynchronized::ProgressStatus	Optional	CQL -Progress status transition for replica associations.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_GroupSynchronized AND SourceInstance.CIM_GroupSynchronized::CopyState <> PreviousInstance.CIM_GroupSynchronized::CopyState AND OBJECTPATH(SourceInstanceModelpath) = OBJECTPATH('string-key-of-group-synchronized')	Conditional	Conditional requirement: Required if groups and semi-fixed indication filters are supported. CQL -Synchronization state transition for a specific replication group association.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_GroupSynchronized AND SourceInstance.CIM_GroupSynchronized::CopyState <> PreviousInstance.CIM_GroupSynchronized::CopyState	Optional	CQL -Synchronization state transition for replication group associations.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity = 'SNIA' AND AlertingManagedElement ISA CIM_StorageSynchronized	Optional	Be notified when CopyState is set to Broken.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity = 'SNIA' AND AlertingManagedElement ISA CIM_GroupSynchronized	Optional	Be notified when CopyState is set to Broken.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity = 'SNIA' AND AlertingManagedElement ISA CIM_StoragePool	Optional	Remaining pool space either below warning threshold set for the pool or there is no remaining space in the pool.

Table 538 - CIM Elements for Replication Services

Element Name	Requirement	Description
SELECT * FROM CIM_AlertIndication WHERE OwningEntity = 'SNIA' AND AlertingManagedElement ISA CIM_RemoteReplicationCollection	Optional	Be notified of changes in RemoteReplicationCollection (ConnectivityCollections).
SELECT * FROM CIM_AlertIndication WHERE OwningEntity = 'SNIA' AND AlertingManagedElement ISA CIM_ProtocolEndpoint	Optional	Be notified of changes in ProtocolEndpoints.

26.8.1 CIM_ElementCapabilities

Associates StorageReplicationCapabilities and ReplicationService.

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

Table 539 describes class CIM_ElementCapabilities.

Table 539 - SMI Referenced Properties/Methods for CIM_ElementCapabilities

Properties	Flags	Requirement	Description & Notes
Capabilities		Mandatory	
ManagedElement		Mandatory	

26.8.2 CIM_GroupSynchronized

Experimental. Associates source and target groups, or a source element to a target group.

Created By: Extrinsic: CreateGroupReplica
Modified By: Extrinsic: ModifyReplicaSynchronization
Deleted By: Extrinsic: ModifyReplicaSynchronization
Requirement: Required if groups are supported.

Table 540 describes class CIM_GroupSynchronized.

Table 540 - SMI Referenced Properties/Methods for CIM_GroupSynchronized

Properties	Flags	Requirement	Description & Notes
RelationshipName		Mandatory	A user relevant name for the relationship between the source and target groups or between a source element and a target group (i.e. one-to-many).
SyncType		Mandatory	Type of association between source and target groups. Values: 6: Mirror 7: Snapshot 8: Clone.
Mode		Mandatory	Specifies when target elements are updated. Values: 2: Synchronous 3: Asynchronous.
RequestedCopyState	N	Optional	Indicates the last requested or desired state for the association. Values: 4: Synchronized 6: Fractured 7: Split 8: Inactive 9: Suspended 10: Failedover 11: Prepared 12: Aborted 15: Not Applicable 16: Partitioned 17: Invalid.

Table 540 - SMI Referenced Properties/Methods for CIM_GroupSynchronized

Properties	Flags	Requirement	Description & Notes
CopyState		Mandatory	State of association between source and target groups, or source element and target group. Values: 2: Initialized 3: UnSynchronized 4: Synchronized 5: Broken 6: Fractured 7: Split 8: Inactive 9: Suspended 10: FailedOver 11: Prepared 12: Aborted 13: Skewed 14: Mixed 15: Not Applicable 16: Partitioned 17: Invalid.

Table 540 - SMI Referenced Properties/Methods for CIM_GroupSynchronized

Properties	Flags	Requirement	Description & Notes
ProgressStatus	N	Optional	Status of association between source and target groups. Values: 2: Completed 3: Dormant 4: Initializing 5: Preparing 6: Synchronizing 7: Resyncing 8: Restoring 9: Fracturing 10: Splitting 11: Failing over 12: Failing back 13: Aborting 14: Mixed 15: Not Applicable 16: Suspending 17: Requires fracture 18: Requires resync 19: Requires activate 20: Pending 21: Detaching.
PercentSynced	N	Optional	Percent of individual elements in the group synched. Values: 0-100.
ConsistencyEnabled		Mandatory	Set to true if consistency is enabled.
ConsistencyType		Conditional	Conditional requirement: Required if group consistency is enabled. Indicates the consistency type used by the groups. Values: 2: Sequential Consistency.

Table 540 - SMI Referenced Properties/Methods for CIM_GroupSynchronized

Properties	Flags	Requirement	Description & Notes
ConsistencyState		Conditional	Conditional requirement: Required if group consistency is enabled. Indicates the current state of consistency. Values: 2: Not Applicable 3: Consistent 4: Inconsistent.
ConsistencyStatus		Conditional	Conditional requirement: Required if group consistency is enabled. Indicates the current status of consistency. Values: 2: Completed 3: Consistency-in-progress 4: Consistency disabled 5: Consistency-error.
WhenEstablished	N	Optional	Specifies when the association was established.
WhenSynchronized	N	Optional	Date and time synchronization of all elements in the group is achieved.
WhenActivated	N	Optional	Specifies when the association was activated.
WhenSuspended	N	Optional	Specifies when the association was suspended.
FailedCopyStopsHostIO	N	Optional	If true, the storage array tells host to stop sending data to source element if copying to a remote element fails. To set this property initially, use ReplicationSettingData parameter in create method. To modify this property, use ModifyInstance intrinsic method.
CopyRecoveryMode	N	Optional	Describes whether the copy operation continues after a broken link is restored. If Manual, the CopyState is set to Suspended after the link is restored. It is required to issue the Resume operation to continue. To set this property initially, use ReplicationSettingData parameter in create method. To modify this property, use ModifyInstance intrinsic method. Values: 2: Automatic 3: Manual 4: Implementation decides.
SyncedElement		Mandatory	
SystemElement		Mandatory	

26.8.3 CIM_HostedAccessPoint (ForProtocolEndpoint)

Associates ProtocolEndpoint to the System on which it is hosted.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Required if remote replication is supported.

Table 541 describes class CIM_HostedAccessPoint (ForProtocolEndpoint).

Table 541 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (ForProtocolEnd-point)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	The Hosting System.
Dependent		Mandatory	The access points that are hosted on this System.

26.8.4 CIM_HostedAccessPoint (ForRemoteServiceAccessPoint)

Associates RemoteServiceAccessPoint to the ComputerSystem.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Required if remote replication is supported.

Table 542 describes class CIM_HostedAccessPoint (ForRemoteServiceAccessPoint).

Table 542 - SMI Referenced Properties/Methods for CIM_HostedAccessPoint (ForRemoteService-AccessPoint)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	The Hosting System.
Dependent		Mandatory	The access points that are hosted on this System.

26.8.5 CIM_HostedCollection (Allocated Resources)

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 Replication Services profile, it is used to associate the Allocated Resources to the top level Computer System of the Replication Services Profile in support of Cascading.

CIM_HostedCollection is subclassed from CIM_HostedDependency.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 543 describes class CIM_HostedCollection (Allocated Resources).

Table 543 - SMI Referenced Properties/Methods for CIM_HostedCollection (Allocated Resources)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

26.8.6 CIM_HostedCollection (Between ComputerSystem and RemoteReplicationCollection)

Associates the RemoteReplicationCollection (ConnectivityCollection) to the hosting System.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Required if remote replication is supported.

Table 544 describes class CIM_HostedCollection (Between ComputerSystem and RemoteReplicationCollection).

Table 544 - SMI Referenced Properties/Methods for CIM_HostedCollection (Between ComputerSystem and RemoteReplicationCollection)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

26.8.7 CIM_HostedCollection (Between ComputerSystem and ReplicationGroup)

Associates the replication group to the hosting System.

Created By: Extrinsic: CreateGroup
Modified By: Extrinsics: DeleteGroup, RemoveMembers
Deleted By: Extrinsic: DeleteGroup
Requirement: Required if groups are supported.

Table 545 describes class CIM_HostedCollection (Between ComputerSystem and ReplicationGroup).

Table 545 - SMI Referenced Properties/Methods for CIM_HostedCollection (Between ComputerSystem and ReplicationGroup)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

26.8.8 CIM_HostedCollection (Remote Resources)

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 Replication Services Profile, it is used to associate the Remote Resources to the top level Computer System of the Replication Services Profile that supports Cascading.

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 546 describes class CIM_HostedCollection (Remote Resources).

Table 546 - SMI Referenced Properties/Methods for CIM_HostedCollection (Remote Resources)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

26.8.9 CIM_HostedService

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 547 describes class CIM_HostedService.

Table 547 - SMI Referenced Properties/Methods for CIM_HostedService

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	The hosting System.
Dependent		Mandatory	The Replication Service hosted on the System.

26.8.10 CIM_MemberOfCollection (Allocated Resources)

This use of MemberOfCollection is to collect all allocated shadow component instances (in the AllocatedResources collection).

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 548 describes class CIM_MemberOfCollection (Allocated Resources).

Table 548 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Allocated Resources)

Properties	Flags	Requirement	Description & Notes
Member		Mandatory	
Collection		Mandatory	

26.8.11 CIM_MemberOfCollection (ProtocolEndpoints to RemoteReplicationCollection)

Associates ProtocolEndpoints to RemoteReplicationCollection (ConnectivityCollection).

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Required if remote replication is supported.

Table 549 describes class CIM_MemberOfCollection (ProtocolEndpoints to RemoteReplicationCollection).

Table 549 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (ProtocolEndpoints to RemoteReplicationCollection)

Properties	Flags	Requirement	Description & Notes
Collection		Mandatory	
Member		Mandatory	

26.8.12 CIM_MemberOfCollection (Remote Resources)

This use of MemberOfCollection is to collect all shadow components (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 550 describes class CIM_MemberOfCollection (Remote Resources).

Table 550 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Remote Resources)

Properties	Flags	Requirement	Description & Notes
Member		Mandatory	
Collection		Mandatory	

26.8.13 CIM_OrderedMemberOfCollection

Associates ReplicationGroup to storage elements.

Created By: Extrinsic: CreateGroup
Modified By: Extrinsics: AddMembers, RemoveMembers
Deleted By: Extrinsics: DeleteGroup, RemoveMembers
Requirement: Required if groups are supported.

Table 551 describes class CIM_OrderedMemberOfCollection.

Table 551 - SMI Referenced Properties/Methods for CIM_OrderedMemberOfCollection

Properties	Flags	Requirement	Description & Notes
AssignedSequence		Mandatory	Indicates relative position of members within a group.
Collection		Mandatory	
Member		Mandatory	

26.8.14 CIM_ProtocolEndpoint

Special purpose endpoint that represents connections between systems.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Required if remote replication is supported.

Table 552 describes class CIM_ProtocolEndpoint.

Table 552 - SMI Referenced Properties/Methods for CIM_ProtocolEndpoint

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

Table 552 - SMI Referenced Properties/Methods for CIM_ProtocolEndpoint

Properties	Flags	Requirement	Description & Notes
ProtocolIFType		Mandatory	Value always reflects protocol type. Values: 1: Other 6: Ethernet CSMA/CD 7: ISO 802.3 CSMA/CD 8: ISO 802.4 Token Bus 15: FDDI 56: Fibre Channel 117: Gigabit Ethernet 4096: IPv4 4097: IPv6 4098: IPv4/IPv6 4111: TCP.
OtherTypeDescription	N	Optional	String identifying the Other connection protocol.
OperationalStatus		Mandatory	An array containing the operational status of protocol end point.

26.8.15 CIM_RemoteReplicationCollection

Collects the ProtocolEndpoints/ServiceAccessPoints used by Replication Services.

Created By: Extrinsic: CreateRemoteReplicationCollection
Modified By: Extrinsics: AddToRemoteReplicationCollection, RemoveFromRemoteReplicationCollection
Deleted By: Extrinsic: Static
Requirement: Required if remote replication is supported.

Table 553 describes class CIM_RemoteReplicationCollection.

Table 553 - SMI Referenced Properties/Methods for CIM_RemoteReplicationCollection

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	Opaque.
ElementName		Optional	User Friendly name.
ConnectivityStatus		Mandatory	An enumeration describing the current or potential connectivity between endpoints in this collection. Values: 2: Connectivity - Up 3: No Connectivity - Down 4: Partitioned - Partial connectivity.

Table 553 - SMI Referenced Properties/Methods for CIM_RemoteReplicationCollection

Properties	Flags	Requirement	Description & Notes
Active	N	Optional	Indicates that this collection is currently active and allows replication activities to the remote elements.
DeleteOnUnassociated	N	Optional	If true, this instance of RemoteReplicationCollection will be deleted when it is no longer associated with an access point.

26.8.16 CIM_RemoteServiceAccessPoint

Created By: Extrinsic: Static
Modified By: Static
Deleted By: Static
Requirement: Required if remote replication is supported.

Table 554 describes class CIM_RemoteServiceAccessPoint.

Table 554 - SMI Referenced Properties/Methods for CIM_RemoteServiceAccessPoint

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
CreationClassName		Mandatory	
SystemName		Mandatory	
Name		Mandatory	
ElementName		Optional	User Friendly name.
AccessInfo		Mandatory	Access or addressing information or a combination of this information for a remote connection. This information can be a host name, network address, or similar information.
InfoFormat		Mandatory	The format of the Management Address (i.e. AccessInfo). For example: "Host Name", "IPv4 Address", "IPv6 Address", "URL". See MOF for the complete list and values.

26.8.17 CIM_ReplicaPoolForStorage

Associates special storage pool for Snapshots (delta replicas) to a source element.

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

Table 555 describes class CIM_ReplicaPoolForStorage.

Table 555 - SMI Referenced Properties/Methods for CIM_ReplicaPoolForStorage

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

26.8.18 CIM_ReplicationEntity

Represents a replication entity such as an entity known by its World Wide Name (WWN).

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

Table 556 describes class CIM_ReplicationEntity.

Table 556 - SMI Referenced Properties/Methods for CIM_ReplicationEntity

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	Key.
Type		Mandatory	Indicates how to interpret the information appearing in EntityID. Values: 2: StoragePool 3: StorageExtent 4: StorageVolume 5: LogicalDisk 6: Filesystem 7: WWN 8: URI 9: Objectpath 10: Encoded in EntityID.
EntityID		Mandatory	An ID representing the resource identified by this entity. For example, the WWN or the URI of an element. The property Type is to be used to interpret the ID.
OtherTypeDescriptio n	N	Optional	Populated when Type has the value of Other.
Persistent	MN	Optional	If false, the instance of this object, not the element represented by this entity, may be deleted at the completion of a copy operation.

26.8.19 CIM_ReplicationGroup

Experimental. Represents a group of elements participating in a replication activity.

Created By: Extrinsic: CreateGroup
Modified By: Extrinsics: AddMembers, RemoveMembers
Deleted By: Extrinsic: DeleteGroup
Requirement: Required if groups are supported.

Table 557 describes class CIM_ReplicationGroup.

Table 557 - SMI Referenced Properties/Methods for CIM_ReplicationGroup

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	Within the scope of an array, the InstanceID opaquely and uniquely identifies an instance of this class.
Persistent	MN	Optional	If false, the group, not the elements associated with the group, may be deleted at the completion of a copy operation.
DeleteOnEmptyElement	M	Mandatory	If true and empty groups are allowed, the group will be deleted when the last element is removed from the group. If empty groups are not allowed, the group will be deleted automatically when the group becomes empty.
DeleteOnUnassociated	M	Mandatory	If true, the group will be deleted when the group is no longer associated with another group. This can happen if all synchronization associations to the individual elements of the group are dissolved.
ConsistentPointInTime	N	Optional	If it is true, it means the point-in-time was created at an exact time with no I/O activities in such a way the data is consistent among all the elements of the group. This property is only valid when the group is a target of a copy operation.

26.8.20 CIM_ReplicationService

Experimental. Base class for Replication Services.

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

Table 558 describes class CIM_ReplicationService.

Table 558 - SMI Referenced Properties/Methods for CIM_ReplicationService

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
Name		Mandatory	
CreateElementReplica()		Mandatory	
CreateGroupReplica()		Conditional	Conditional requirement: Required if groups are supported.
CreateListReplica()		Optional	
CreateSynchronizationAspect()		Optional	
ModifyReplicaSynchronization()		Mandatory	
ModifyListSynchronization()		Optional	
ModifySettingsDefineState()		Optional	
CreateGroup()		Conditional	Conditional requirement: Required if groups are supported.
DeleteGroup()		Conditional	Conditional requirement: Required if groups are supported.
AddMembers()		Conditional	Conditional requirement: Required if groups are supported.
RemoveMembers()		Conditional	Conditional requirement: Required if groups are supported.
GetAvailableTargetElements()		Optional	
GetPeerSystems()		Optional	
GetReplicationRelationships()		Optional	
GetServiceAccessPoints()		Optional	
AddReplicationEntity()		Optional	
AddServiceAccessPoint()		Optional	
AddSharedSecret()		Optional	

Table 558 - SMI Referenced Properties/Methods for CIM_ReplicationService

Properties	Flags	Requirement	Description & Notes
CreateGroupReplicaFromElements()		Optional	
GetReplicationRelationshipInstance()		Optional	
ModifyListSettingsDefineState()		Optional	
CreateRemoteReplicationCollection()		Optional	
AddToRemoteReplicationCollection()		Optional	
RemoveFromRemoteReplicationCollection()		Optional	

26.8.21 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: Mandatory

Table 559 describes class CIM_ReplicationServiceCapabilities.

Table 559 - SMI Referenced Properties/Methods for CIM_ReplicationServiceCapabilities

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
ElementName		Mandatory	User Friendly name.

Table 559 - SMI Referenced Properties/Methods for CIM_ReplicationServiceCapabilities

Properties	Flags	Requirement	Description & Notes
SupportedReplicationTypes		Mandatory	Enumeration indicating the supported SyncType/Mode/Local-or-Remote combinations. Values: 2: Synchronous Mirror Local 3: Asynchronous Mirror Local 4: Synchronous Mirror Remote 5: Asynchronous Mirror Remote 6: Synchronous Snapshot Local 7: Asynchronous Snapshot Local 8: Synchronous Snapshot Remote 9: Asynchronous Snapshot Remote 10: Synchronous Clone Local 11: Asynchronous Clone Local 12: Synchronous Clone Remote 13: Asynchronous Clone Remote.
SupportedStorageObjects		Mandatory	Enumeration indicating the supported storage objects. Values: 2: StorageVolume 3: LogicalDisk.
SupportedAsynchronousActions		Mandatory	Identify replication methods using job control. Values: 2: CreateElementReplica 3: CreateGroupReplica 4: CreateSynchronizationAspect 5: ModifyReplicaSynchronization 6: ModifyListSynchronization 7: ModifySettingsDefineState 8: GetAvailableTargetElements 9: GetPeerSystems 10: GetReplicationRelationships 11: GetServiceAccessPoints 19: CreateListReplica.

Table 559 - SMI Referenced Properties/Methods for CIM_ReplicationServiceCapabilities

Properties	Flags	Requirement	Description & Notes
SupportedSynchronousActions		Mandatory	Identify replication methods not using job control. Values: 2: CreateElementReplica 3: CreateGroupReplica 4: CreateSynchronizationAspect 5: ModifyReplicaSynchronization 6: ModifyListSynchronization 7: ModifySettingsDefineState 8: GetAvailableTargetElements 9: GetPeerSystems 10: GetReplicationRelationships 11: GetServiceAccessPoints 12: CreateGroup 13: DeleteGroup 14: AddMembers 15: RemoveMembers 16: AddReplicationEntity 17: AddServiceAccessPoint 18: AddSharedSecret 19: CreateListReplica.
ConvertSyncTypeToReplicationType()		Mandatory	
ConvertReplicationTypeToSyncType()		Mandatory	
GetSupportedCopyStates()		Mandatory	
GetSupportedGroupCopyStates()		Conditional	Conditional requirement: Required if groups are supported.
GetSupportedWaitForCopyStates()		Optional	
GetSupportedFeatures()		Mandatory	
GetSupportedGroupFeatures()		Conditional	Conditional requirement: Required if groups are supported.

Table 559 - SMI Referenced Properties/Methods for CIM_ReplicationServiceCapabilities

Properties	Flags	Requirement	Description & Notes
GetSupportedConsistency()		Conditional	Conditional requirement: Required if groups are supported.
GetSupportedOperations()		Mandatory	
GetSupportedGroupOperations()		Conditional	Conditional requirement: Required if groups are supported.
GetSupportedListOperations()		Optional	
GetSupportedSettingsDefineStateOperations()		Optional	
GetSupportedThinProvisioningFeatures()		Optional	
GetSupportedMaximum()		Optional	
GetDefaultConsistency()		Conditional	Conditional requirement: Required if groups are supported.
GetDefaultGroupPersistency()		Conditional	Conditional requirement: Required if groups are supported.
GetSupportedReplicationSettingData()		Optional	
GetDefaultReplicationSettingData()		Optional	
GetSupportedConnectionFeatures()		Optional	

26.8.22 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 560 describes class CIM_ReplicationSettingData.

Table 560 - SMI Referenced Properties/Methods for CIM_ReplicationSettingData

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
ElementName		Mandatory	User Friendly name.

Table 560 - SMI Referenced Properties/Methods for CIM_ReplicationSettingData

Properties	Flags	Requirement	Description & Notes
Pairing	MN	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).
UnequalGroupsAction	MN	Optional	Indicates what should happen if number of elements in source and target are unequal. Values: 2: Return an error 3: Allow larger source group 4: Allow larger target group.
DesiredCopyMethodology	MN	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 9: Snap-And-Clone.
TargetElementSupplier	MN	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 5: Client must supply.

Table 560 - SMI Referenced Properties/Methods for CIM_ReplicationSettingData

Properties	Flags	Requirement	Description & Notes
ThinProvisioningPolicy	MN	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	MN	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	MN	Optional	If non-zero, it specifies the interval between the snapshots of source element, for example, every 23 minutes (00000000002300.000000:000). If zero or NULL, the implementation decides.
Multihop	MN	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.
OnGroupOrListError	MN	Optional	This property applies to group or list operations. It specifies what the implementation should do if an error is encountered before all entries in the group or list are processed. Default is to Stop. 2: Continue 3: Stop.
CopyPriority	MN	Optional	This property sets the StorageSynchronized.CopyPriority property. CopyPriority allows the priority of background copy operation to be managed relative to host I/O operations during a sequential background copy operation. 0: Not Managed 1: Low 2: Same (as host I/O) 3: High 4: Urgent.
FailedCopyStopsHostIO	MN	Optional	If true, the storage array tells host to stop sending data to source element if copying to a remote element fails.

Table 560 - SMI Referenced Properties/Methods for CIM_ReplicationSettingData

Properties	Flags	Requirement	Description & Notes
CopyRecoveryMode	MN	Optional	Describes whether the copy operation continues after a broken link is restored. If Manual, the CopyState is set to Suspended after the link is restored. It is required to issue the Resume operation to continue. Values: 2: Automatic 3: Manual 4: Implementation decides.
UnequalListsAction	MN	Optional	Indicates what should happen if number of elements in source and target lists are unequal. Values: 2: Return an error 3: Allow source list to be larger 4: Allow target list to be larger.
DeltaUpdateBlocks	MN	Optional	This property applies to Delta-Update copy methodology. If non-zero, it specifies the snapshots of source element should be created after this number of blocks have been modified. If both DeltaUpdateBlocks and DeltaUpdateInterval are specified the snapshot is created based on which criterion occurs first. If NULL or 0, the implementation decides the number of blocks.
ReadOnly	MN	Optional	This property specifies whether the source, the target, or both elements should be read only to the host. Values: 2: SystemElement (source) 3: SyncedElement (target) 4: Both.

26.8.23 CIM_SAPAvailableForElement

This association identifies the element that is serviced by the ProtocolEndpoint.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Required if remote replication is supported.

Table 561 describes class CIM_SAPAvailableForElement.

Table 561 - SMI Referenced Properties/Methods for CIM_SAPAvailableForElement

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	The managed element.
AvailableSAP		Mandatory	The servicing protocol end point.

26.8.24 CIM_ServiceAffectsElement (Between ReplicationService and RemoteReplicationCollection)

Associates Replication Service to RemoteReplicationCollection (ConnectivityCollection).

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Required if remote replication is supported.

Table 562 describes class CIM_ServiceAffectsElement (Between ReplicationService and RemoteReplicationCollection).

Table 562 - SMI Referenced Properties/Methods for CIM_ServiceAffectsElement (Between ReplicationService and RemoteReplicationCollection)

Properties	Flags	Requirement	Description & Notes
AffectingElement		Mandatory	Replication Service.
AffectedElement		Mandatory	Remote Replication Collection.

26.8.25 CIM_ServiceAffectsElement (Between ReplicationService and ReplicationEntity)

Associates Replication Service to ReplicationEntity.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 563 describes class CIM_ServiceAffectsElement (Between ReplicationService and ReplicationEntity).

Table 563 - SMI Referenced Properties/Methods for CIM_ServiceAffectsElement (Between ReplicationService and ReplicationEntity)

Properties	Flags	Requirement	Description & Notes
AffectingElement		Mandatory	Replication Service.
AffectedElement		Mandatory	Replication Entity.

26.8.26 CIM_ServiceAffectsElement (Between ReplicationService and ReplicationGroup)

Associates Replication Service to Replication Group.

Created By: Extrinsic: CreateGroup

Modified By: Extrinsics: DeleteGroup, RemoveMembers

Deleted By: Extrinsic: DeleteGroup

Requirement: Required if groups are supported.

Table 564 describes class CIM_ServiceAffectsElement (Between ReplicationService and ReplicationGroup).

Table 564 - SMI Referenced Properties/Methods for CIM_ServiceAffectsElement (Between ReplicationService and ReplicationGroup)

Properties	Flags	Requirement	Description & Notes
AffectingElement		Mandatory	Replication Service.
AffectedElement		Mandatory	Replication Group.

26.8.27 CIM_SettingsDefineState (Between ReplicationGroup and SynchronizationAspect)

Associates a replication group to an instance of SynchronizationAspect.

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

Table 565 describes class CIM_SettingsDefineState (Between ReplicationGroup and SynchronizationAspect).

Table 565 - SMI Referenced Properties/Methods for CIM_SettingsDefineState (Between ReplicationGroup and SynchronizationAspect)

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	Storage Element.
SettingData		Mandatory	Synchronization Aspect.

26.8.28 CIM_SettingsDefineState (Between storage object and SynchronizationAspect)

Associates a storage object to an instance of SynchronizationAspect.

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

Table 566 describes class CIM_SettingsDefineState (Between storage object and SynchronizationAspect).

Table 566 - SMI Referenced Properties/Methods for CIM_SettingsDefineState (Between storage object and SynchronizationAspect)

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	Storage Element.
SettingData		Mandatory	Synchronization Aspect.

26.8.29 CIM_SharedSecret

Created By: Extrinsic: AddSharedSecret
Modified By: Static
Deleted By: Static
Requirement: Required if remote replication is supported.

Table 567 describes class CIM_SharedSecret.

Table 567 - SMI Referenced Properties/Methods for CIM_SharedSecret

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	Key.
SystemName		Mandatory	Key.
ServiceCreationClassName		Mandatory	Key.
ServiceName		Mandatory	Key.
RemoteID		Mandatory	Key, The identity of the client as known on the remote system.
Secret		Mandatory	A secret.

26.8.30 CIM_StorageSynchronized

Experimental. Associates replica target element to source element.

Created By: Extrinsics: CreateElementReplica, CreateGroupReplica, CreateListReplica
Modified By: Extrinsic: ModifyReplicaSynchronization
Deleted By: Extrinsic: ModifyReplicaSynchronization
Requirement: Mandatory

Table 568 describes class CIM_StorageSynchronized.

Table 568 - SMI Referenced Properties/Methods for CIM_StorageSynchronized

Properties	Flags	Requirement	Description & Notes
WhenSynced	N	Optional	Date and time synchronization of the elements is achieved.
WhenEstablished	N	Optional	Specifies when the association was established.
WhenSynchronized	N	Optional	Specifies when the CopyState has a value of Synchronized.
WhenActivated	N	Optional	Specifies when the association was activated.
WhenSuspended	N	Optional	Specifies when the association was suspended.
SyncMaintained		Mandatory	Boolean indicating whether synchronization is maintained.

Table 568 - SMI Referenced Properties/Methods for CIM_StorageSynchronized

Properties	Flags	Requirement	Description & Notes
SyncType		Mandatory	Type of association between source and target groups. Values: 6: Mirror 7: Snapshot 8: Clone.
Mode		Mandatory	Specifies when target elements are updated. Values: 2: Synchronous 3: Asynchronous.
RequestedCopyState		Optional	Indicates the last requested or desired state for the association. Values: 4: Synchronized 6: Fractured 7: Split 8: Inactive 9: Suspended 10: Failedover 11: Prepared 12: Aborted 15: Not Applicable 16: Partitioned 17: Invalid.
ReplicaType		Optional	

Table 568 - SMI Referenced Properties/Methods for CIM_StorageSynchronized

Properties	Flags	Requirement	Description & Notes
CopyState		Mandatory	State of association between source and target groups. Values: 2: Initialized 3: Unsynchronized 4: Synchronized 5: Broken 6: Fractured 7: Split 8: Inactive 9: Suspended 10: FailedOver 11: Prepared 12: Aborted 13: Skewed 14: Mixed 15: Not Applicable 16: Partitioned 17: Invalid.

Table 568 - SMI Referenced Properties/Methods for CIM_StorageSynchronized

Properties	Flags	Requirement	Description & Notes
ProgressStatus	N	Optional	Status of association between source and target groups. Values: 2: Completed 3: Dormant 4: Initializing 5: Preparing 6: Synchronizing 7: Resyncing 8: Restoring 9: Fracturing 10: Splitting 11: Failing over 12: Failing back 13: Aborting 14: Mixed 15: Not Applicable 16: Suspending 17: Requires fracture 18: Requires resync 19: Requires activate 20: Pending 21: Detaching.
PercentSynced	N	Optional	Specifies the percent of the work completed to reach synchronization. For synchronized associations (e.g. SyncType Mirror), while fractured, the percent difference between source and target elements can derived by subtracting PercentSynched from 100.

Table 568 - SMI Referenced Properties/Methods for CIM_StorageSynchronized

Properties	Flags	Requirement	Description & Notes
CopyPriority	MN	Optional	CopyPriority allows the priority of background copy engine I/O to be managed relative to host I/O operations during a sequential background copy operation. Values: 0: Not Managed 1: Low 2: Same (as host I/O) 3: High 4: Urgent.
UndiscoveredElement	N	Optional	Specifies whether the source, the target, or both elements involved in a copy operation are undiscovered. If NULL both source and target elements are considered discovered. Values: 2: SystemElement 3: SyncedElement 4: Both.
FailedCopyStopsHostIO	N	Optional	If true, the storage array tells host to stop sending data to source element if copying to a remote element fails. To set this property initially, use ReplicationSettingData parameter in create method. To modify this property, use ModifyInstance intrinsic method.
CopyRecoveryMode	N	Optional	Describes whether the copy operation continues after a broken link is restored. If Manual, the CopyState is set to Suspended after the link is restored. It is required to issue the Resume operation to continue. To set this property initially, use ReplicationSettingData parameter in create method. To modify this property, use ModifyInstance intrinsic method. Values: 2: Automatic 3: Manual 4: Implementation decides.
ReadOnly	N	Optional	This property specifies whether the source, the target, or both elements are "read only" to the host. Values: 2: SystemElement 3: SyncedElement 4: Both.
SyncedElement		Mandatory	
SystemElement		Mandatory	

26.8.31 CIM_SynchronizationAspect

Experimental. Keeps track of source of a copy operation and point-in-time.

Created By: Extrinsics: CreateElementReplica, CreateListReplica, CreateSynchronizationAspect
Modified By: Extrinsic: ModifyReplicaSynchronization
Deleted By: Extrinsics: ModifyReplicaSynchronization, ModifySettingsDefineState
Requirement: Optional

Table 569 describes class CIM_SynchronizationAspect.

Table 569 - SMI Referenced Properties/Methods for CIM_SynchronizationAspect

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
SyncType		Mandatory	Type of association between source and target elements. Values: 6: Mirror 7: Snapshot 8: Clone.
ConsistencyEnabled		Conditional	Conditional requirement: Required if groups are supported. Set to true if consistency is enabled.
ElementName		Mandatory	An end user relevant name. The value will be stored in the ElementName property of the created SynchronizationAspect.
ConsistencyType		Conditional	Conditional requirement: Required if group consistency is enabled. Indicates the consistency type used by the groups. Values: 2: Sequential Consistency.
CopyStatus	N	Optional	Describes the status of copy operation. Values: 2: Not Applicable 3: Operation In Progress 4: Operation Completed.

Table 569 - SMI Referenced Properties/Methods for CIM_SynchronizationAspect

Properties	Flags	Requirement	Description & Notes
CopyMethodology	N	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 9: Snap-And-Clone.
WhenPointInTime	N	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.

26.8.32 SNIA_AllocatedResources

An instance of a default SNIA_AllocatedResources defines the set of components that are allocated and in use by the Replication Services Profile.

SNIA_AllocatedResources is subclassed from CIM_SystemSpecificCollection.

At least one instance of the SNIA_AllocatedResources shall exist for the Replication Services 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: Optional

Table 570 describes class SNIA_AllocatedResources.

Table 570 - SMI Referenced Properties/Methods for SNIA_AllocatedResources

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
ElementName		Mandatory	A user-friendly name for the AllocatedResources collection (e.g., Allocated StorageVolumes).

Table 570 - SMI Referenced Properties/Methods for SNIA_AllocatedResources

Properties	Flags	Requirement	Description & Notes
ElementType		Mandatory	The type of remote resources collected by the AllocatedResources collection. For this version of SMI-S, the only value supported is '2' (Any Type).
CollectionDiscriminator		Mandatory	Experimental. This is an array of values that shall contain one or more values from the list: 'SNIA:Target Volumes', 'SNIA:Source Volumes', 'SNIA:Target Volume Group', 'SNIA:Source Volume Group'.

26.8.33 SNIA_RemoteResources

An instance of a default SNIA_RemoteResources defines the set of shadow components that are available to be used by the Replication Services Profile that supports Cascading.

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 Replication Services Profile that supports Cascading.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 571 describes class SNIA_RemoteResources.

Table 571 - SMI Referenced Properties/Methods for SNIA_RemoteResources

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
ElementName		Mandatory	A user-friendly name for the RemoteResources collection (e.g., Remote StorageVolumes).
ElementType		Mandatory	The type of remote resources collected by the RemoteResources collection. This shall be '2' (Any Type).
CollectionDiscriminator		Mandatory	Experimental. This is an array of values that shall contain one or more values from the list: 'SNIA:Target Volumes', 'SNIA:Source Volumes', 'SNIA:Target Volume Group', 'SNIA:Source Volume Group', 'SNIA:Remote Storage Pools'.

EXPERIMENTAL

EXPERIMENTAL

Clause 27: Pools from Volumes Profile

27.1 Description

27.1.1 Overview

The Pools from Volumes Profile defines how a pool may be created from StorageVolumes. The Block Services Package defines how to create a StoragePool from unallocated storage. However, there are some devices that allow the user to create storage pools from already allocated volumes, necessitating this profile. This is a similar concept to Volume Composition, in that the volumes are combined into a larger entity and are no longer available for use as regular volumes. The specific use cases that have been identified for these kinds of pools are for snapshot replica pools and thin provisioned volume pools.

27.1.2 Terminology

This profile uses the following terms to help distinguish between the different uses of StoragePool and StorageVolume. This is done to help distinguish which kind of StorageVolume or StoragePool is being referred to.

Constituent Volume -- StorageVolumes used to create a concrete StoragePool.

Pool Volume -- StorageVolume created from a Constituent Pool.

Constituent Pool -- A concrete StoragePool created from constituent volumes.

27.1.3 Relationship to Block Services Package

The Pools from Volumes Profile extends the Block Services Package with additional descriptions and definitions showing how such pools may be created and how to model the constituent volumes. The existing Block Services classes, properties, and methods are used.

27.1.4 Relationship to Extent Composition

This profile shall not require Extent Composition. Some of the examples make use of Extent Composition but only to demonstrate the relationship of the volumes to the underlying extents.

This profile shall not require any BasedOn association to any underlying extents from the volumes created from the constituent pool, even if Extent Composition is supported by the instrumentation.

27.1.5 Class Model

Figure 140 shows the classes used in this profile. These are the same classes used in the autonomous profile and Block Services Package.

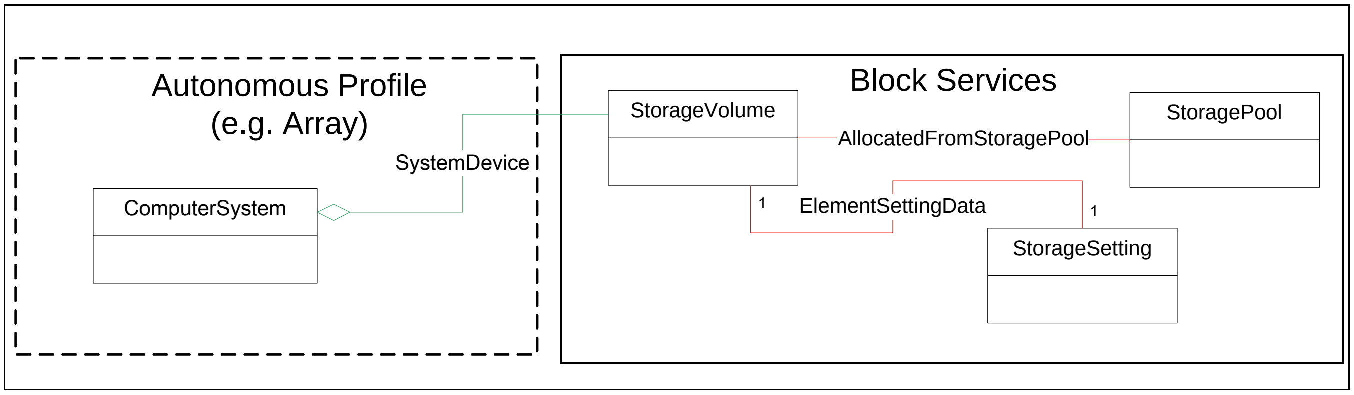

Figure 140 - Class Model

27.1.6 Model Elements

27.1.6.1 StorageVolume

StorageVolume is used in three different contexts in this profile. The first is the normal usage as described in Clause 5: Block Services Package. The second is as a “constituent volume.” These StorageVolumes are normal volumes that have been used to create a concrete StoragePool. This volume has the same associations as normal StorageVolumes with its underlying elements, namely the AllocatedFromStoragePool association to the concrete StoragePool and the BasedOn association to the StorageExtent. Once used to create a StoragePool, this volume can be identified by its Usage property. The third type of StorageVolume is a volume created from the constituent pool. This is referred to as a “pool volume.” This acts as a normal StorageVolume, with the exception that it does not have a BasedOn association to any antecedent StorageExtent, even if Extent Composition is supported.

27.1.6.1.1 Volume Visibility

In some implementations, these volumes may still be visible in a list of volumes reported by the array after pool creation. In this profile, these volumes are called "constituent volumes" to distinguish them from volumes allocated from the pool. Even though these volumes are visible, they are not usable as normal volumes. Some instrumentation may need the ability to “see” a constituent volume in order to perform copy operation or to resize (e.g., shrink) the constituent pool.

27.1.6.2 StoragePool

StoragePool is used in two contexts in this profile. The first is the regular concrete StoragePool. The second is the constituent pool that is created by the constituent volumes.

27.1.7 Example

This example will show the model changes that occur when a constituent StoragePool is created from StorageVolumes. Figure 141 shows the starting conditions. There are two normal StorageVolumes allocated from a concrete pool, labeled ConcretePool in the diagram. This example follows the Extent Composition model, so each volume has a BasedOn association to an underlying StorageExtent that is a ConcreteComponent of the concrete StoragePool. Depending upon the instrumentation, there may be intermediate extents between the volume and extent (e.g. if the instrumentation follows the Volume Composition model, there may be an intermediate CompositeExtent between the StorageVolume and StorageExtent). Although not shown in the diagram, for each

ConcreteComponent association, there is also an AssociatedComponentExtent association between the same two instances the ConcreteComponent associates.

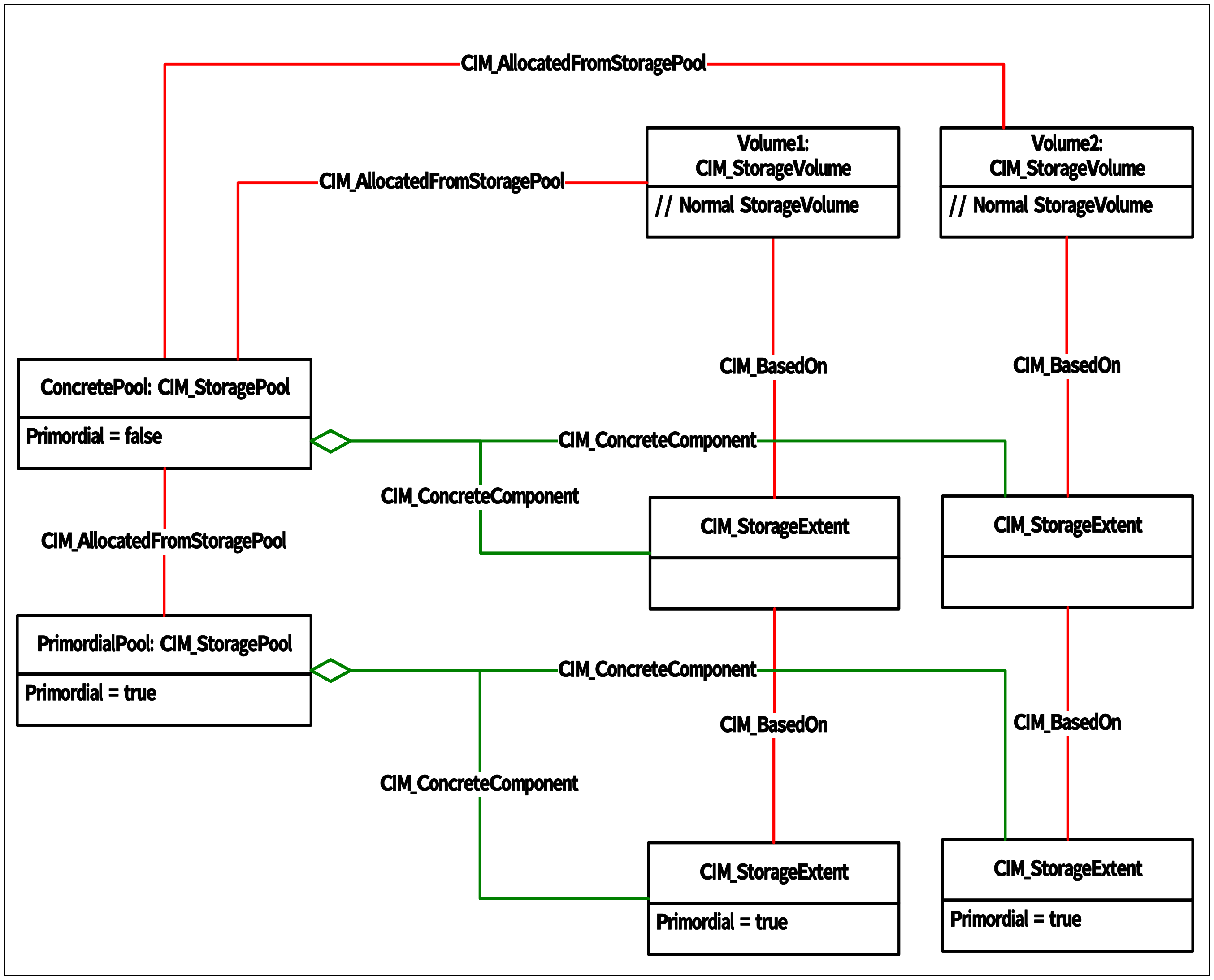

Figure 141 - Before Pool Creation

The next figure, Figure 142, shows the changes that would occur in the model after creation of the StoragePool from the StorageVolumes (e.g. as a result of a invocation of he CreateOrModifyStoragePool method). In this diagram, both of the volumes have been used to create a constituent StoragePool, labeled CreatedPool in the diagram. The following model changes occur:

- AllocatedFromStoragePool.SpaceConsumed value goes to 0 for the constituent volumes (Volume1 and Volume2). This is needed to prevent double counting the storage in the newly created StoragePool
- A ConcreteDependency association is added between the newly created StoragePool (CreatedPool) and each of the StorageVolumes used to create the pool to show that they represent the same piece of storage
- A CompositeExtent is created for each created volume (PoolVolume) and associated to the created pool via ConcreteComponent and AssociatedRemainingExtent (not shown in figure)
- A one-to-one BasedOn association from the created volume to the created CompositeExtent is created BasedOn associations are created to associate each of these created CompositeExtents to all of the extents

(StorageExtent or CompositeExtent) that have a BasedOn association to the StorageVolume that is a constituent volume of the created StoragePool

These changes are consistent with the Extent Composition Profile.

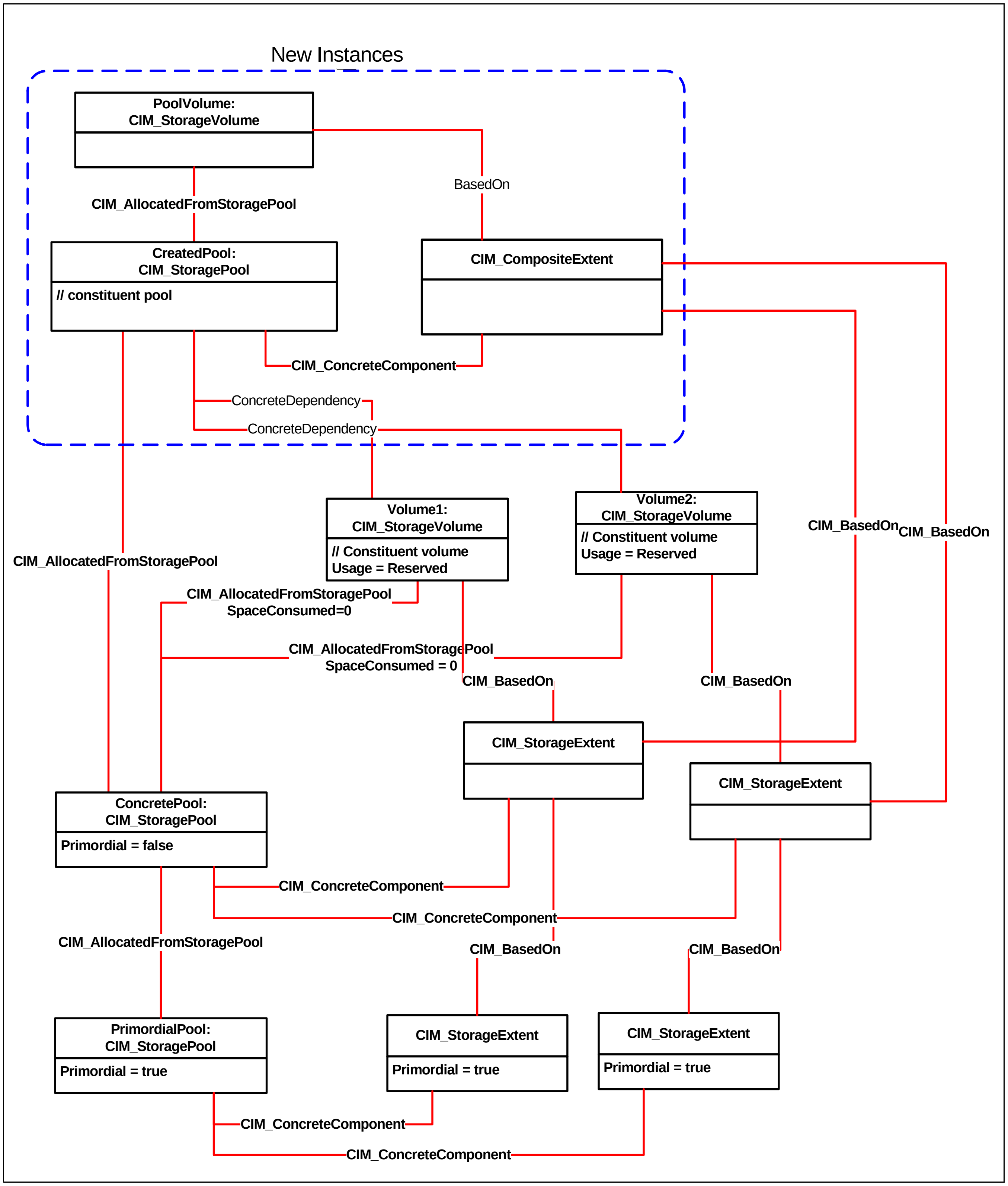

Figure 142 - After Pool Creation

If Extent Composition is not implemented, the model changes are much simpler. The following figure, Figure 143, shows what model changes occur, summarized below.

- AllocatedFromStoragePool.SpaceConsumed value goes to 0 for the constituent volumes (Volume1 and Volume2). This is needed to prevent double counting the storage in the newly created StoragePool
- A ConcreteDependency association is added between the newly created StoragePool (CreatedPool) and each of the StorageVolumes used to create the pool to show that they represent the same piece of storage

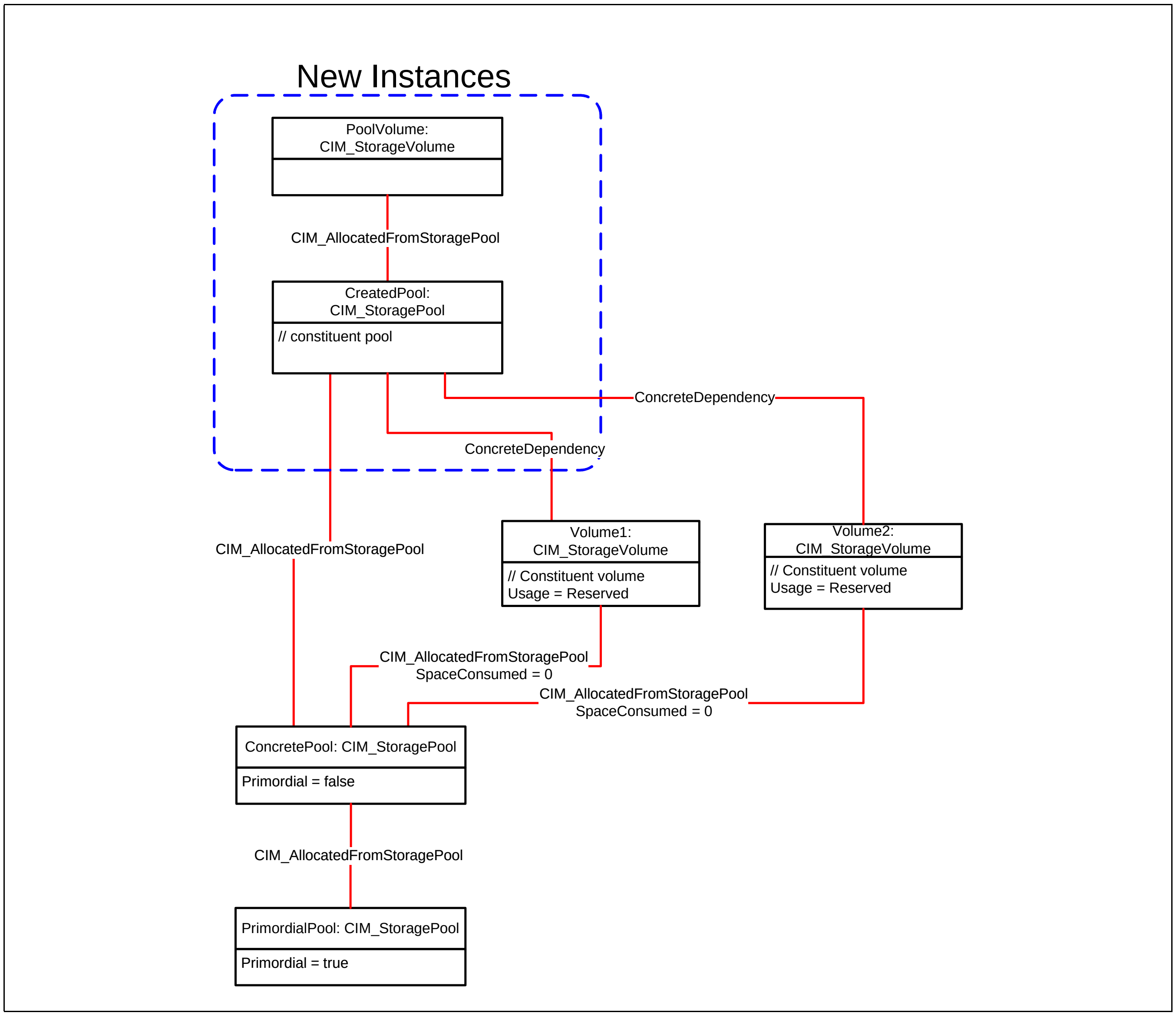
New Instances
PoolVolume: CIM_StorageVolume
CIM_AllocatedFromStoragePool
CreatedPool: CIM_StoragePool
// constituent pool
ConcreteDependency
ConcreteDependency
CIM_AllocatedFromStoragePool
Volume1: CIM_StorageVolume
// Constituent volume
Usage = Reserved
Volume2: CIM_StorageVolume
// Constituent volume
Usage = Reserved
CIM_AllocatedFromStoragePool
SpaceConsumed = 0
CIM_AllocatedFromStoragePool
SpaceConsumed = 0
ConcretePool: CIM_StoragePool
Primordial = false
CIM_AllocatedFromStoragePool
PrimordialPool: CIM_StoragePool
Primordial = true

Figure 143 - After Pool Creation without Extent Composition

27.2 Block Services Enhancements

The following classes, methods, and properties from Block Services are enhanced as follows.

27.2.1 StoragePool Manipulation Methods

See 5.1.6.2 "StoragePool Manipulation Methods".

Possible inputs to CreateOrModifyStoragePool shall also allow StorageVolumes. More details may be found in 27.6 "Methods of the Profile".

27.2.2 Declaring Storage Configuration Options

See 5.1.7 "Declaring Storage Configuration Options".

SNIA_StorageConfigurationCapabilities.SupportedStoragePoolFeatures is enhanced to allow “StorageVolumes” as one of the valid options.

27.2.3 The Usage Property

See 5.1.13 "The Usage Property".

The constituent volume can be identified by its Usage property. The value to use is Reserved for Computer System.

27.3 Health and Fault Management Considerations

The same Health and Fault Management Considerations from Block Services apply here.

27.4 Cascading Considerations

Not defined in this specification

27.5 Supported Profiles, Subprofiles, and Packages

This profile requires and extends the Block Services Package.

Use of the Extent Composition Profile is optional in this profile.

27.6 Methods of the Profile

No new methods are defined. Methods from Block Services are enhanced as follows.

27.6.1 CreateOrModifyStoragePool

See 5.5.3.3 "CreateOrModifyStoragePool".

In the context of the Pools from Volumes Profile, a list of StorageVolumes shall be the only allowed type for the InExtents[] parameter used to build the constituent pool. Use of StorageExtents is already allowed by the Block Services Package and this profile shall not change that. The CreateOrModifyStoragePool method signature is listed below with a description of the parameters used when creating a StoragePool from StorageVolumes.

```
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);
```

The parameters are as follows:

- ElementName: If the instrumentation supports naming of StoragePools this parameter may be used to assign a name to the StoragePool

- 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: Null should be used for the Size parameter as all the passed in capacity (as specified by InExtents) shall be used to create the StoragePool. Size may be specified, but is not recommended, as it may not be possible to accurately estimate the resulting pool size ahead of time, due capacity being reserved for StoragePool overhead.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.
- InPools[]: This shall be null when creating a pool. When modifying a pool, there shall be exactly one entry, corresponding to the pool being modified.
- InExtents[]: This is an array of strings containing Object references (see 4.11.5 of *DMTF DSP00200 CIM Operations over HTTP* for format) to source StorageVolumes.
- TheElement: If the method completes without creating a Job, then the TheElement is the object path of the StoragePool that is created. Otherwise, TheElement shall be null. When the TheElement is null, then the storage element created can be determined by using the Job model.

27.6.2 DeleteStoragePool

See 5.5.3.5 "DeleteStoragePool".

When deleting the constituent pool, the constituent volume’s AllocatedFromStoragePool.SpaceConsumed value returns to the value it had before it was used to build the constituent pool. The RemainingManagedStorage of the associated parent StoragePool will not change, as the same amount of storage is still in use, albeit in the formerly constituent volumes instead of the constituent pool. The former constituent volumes will have their Usage value reset to that of a normal volume,

The parameters and their meanings are the same as in Block Services DeleteStoragePool.

27.6.3 Storage Element Modification

See 5.5.4.4.1 "Storage Element Modification".

For a constituent pool, the capacity may be expandable by providing the references to existing component StorageVolumes of the StoragePool and additional references to normal StorageVolumes. A constituent pool’s capacity may be reducible by providing references to some, but not all, of the current constituent volumes of the StoragePool. If the summary of the capacity of the referenced input StorageVolumes 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.

What this means in relation to the CreateOrModifyStoragePool method is that the InPools[] parameter shall have exactly one entry, that of the StoragePool being modified. This specification shall only define the case where the StoragePool being modified shall have been created from StorageVolumes.

27.7 Client Considerations and Recipes

27.7.1 Client Considerations

Not included in this standard.

27.7.2 Recipe 1: Create StoragePool

```
// DESCRIPTION
// The goal of this recipe is to create a StoragePool from StorageVolumes
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
// 1.A reference to a CIM_ComputerSystem storage array is previously
//      defined in the $BlockServer-> variable
// 2.#Size is set to the size for the new Storage Pool in bytes
//    #Size = total size of the volumes in bytes
// 3.#StorageElementClass is set to the class name of the element being created
//    like CIM_StorageVolume or CIM_LogicalDisk.
// 4. The volumes to use to create the pool have been identified and
//    their object paths stored in $Volumes->[]


// 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]

// See if the service supports thin provisioned pool creation
// There should be zero or only one StorageConfigurationCapabilities instance
$ServiceCapabilities[] = Associators(
     $StorageConfigurationService->,
     “CIM_ElementCapabilities”,
     “SNIA_StorageConfigurationCapabilities”,
     null, null, false, false, null)

if ($ServiceCapabilities[] == null ||
    $ServiceCapabilities[].length == 0) {
   <ERROR! Creation of thin provisioned pools not supported>
}

if (contains(2, // Storage Pool Creation
        $ServiceCapabilities[0].SupportedSynchronousActions[])
  || contains(2, // Storage Pool Creation
     $ServiceCapabilities[0].SupportedAsynchronousActions[]))
{
  #SupportsPoolCreation = true
}

if (contains(
     2, // Storage Pool Creation
     $ServiceCapabilities[0].SupportedAsyncronousActions[]))
{
  #PoolCreationProducesJob = true
}

if (contains(5, // 5 = StorageVolumes
     $ServiceCapabilities[0].SupportedStoragePoolFeatures[]))
{
  #SupportsPoolsFromVolumes = true
}

// Return if thin provisioned pools cannot be created
if ((#SupportesPoolCreation == false) &&
    (#SupportsPoolsFromVolumes == false)) {
   <ERROR! Creation of thin provisioned pools not supported>
}
$StorageCapabilitiesOffered = $ServiceCapabilities[0]


// Step 5. Register for indications on configuration jobs
if(#PoolCreationProducesJob == true)
{
     // ‘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
%InArguments["ElementName"] = NULL// we do not care what the name is
%InArguments["Goal"]        = NULL  // use default setting
%InArguments["Size"]        = #Size
%InArguments["InExtents"]   = $Volumes->[]
%InArguments["InPools"]     = null

#ReturnValue = InvokeMethod(
        $StorageConfigurationService->,
        "CreateOrModifyStoragePool",
        %InArguments, %OutArguments)

if(#ReturnValue != 0 && #ReturnValue != 4096)
{   // Storage Pool was not created
        <ERROR! Failed >
}

if(#PoolCreationProducesJob == true && $PoolCreationJob-> != null)
{
  $PoolCreationJob-> = %OutArguments["Job"]

     //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
}
else {
  $NewPool-> = %OutArguments["Pool"]
}

// Use the new pool
```

27.8 Registered Name and Version

Pools from Volumes version 1.4.0 (Component Profile)

27.9 CIM Elements

Table 572 describes the CIM elements for Pools from Volumes.

Table 572 - CIM Elements for Pools from Volumes

Element Name	Requirement	Description
27.9.1 CIM_AllocatedFromStoragePool (Volume from Pool)	Mandatory	AllocatedFromStoragePool as defined in Block Services.
27.9.2 CIM_ElementCapabilities	Mandatory	Associates StorageCapabilities or StorageConfiguationCapabilities with StoragePool.
27.9.3 CIM_StorageCapabilities	Mandatory	Capabilities class used to generate StorageSettings. Also associated to StoragePools via ElementCapabilities.
27.9.4 CIM_StorageVolume	Mandatory	StorageVolume as defined in Block Services.
27.9.5 CIM_SystemDevice	Mandatory	Associates top level system from Array, Virtualizer, ... to StorageVolume.
27.9.6 SNIA_StorageConfigurationCapabilities	Mandatory	SupportedStoragePoolFeatures as defined in SNIA_StorageConfigurationCapabilities, with the addition of support for StorageVolumes as inputs to pool creation.
27.9.7 SNIA_StoragePool	Mandatory	StoragePool as defined in Block Services.
27.9.8 SNIA_StorageSetting	Optional	StorageSetting as defined in Block Services.

27.9.1 CIM_AllocatedFromStoragePool (Volume from Pool)

AllocatedFromStoragePool as defined in Block Services.

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

Table 573 describes class CIM_AllocatedFromStoragePool (Volume from Pool).

Table 573 - SMI Referenced Properties/Methods for CIM_AllocatedFromStoragePool (Volume from Pool)

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

27.9.2 CIM_ElementCapabilities

Associates StorageCapabilities or StorageConfiguationCapabilities with StoragePool.

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

Table 574 describes class CIM_ElementCapabilities.

Table 574 - SMI Referenced Properties/Methods for CIM_ElementCapabilities

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

27.9.3 CIM_StorageCapabilities

Capabilities class used to generate StorageSettings. Also associated to StoragePools via ElementCapabilities.

Created By: Extrinsic: StorageCapabilities.CreateSetting
Modified By: Static
Deleted By: Static
Requirement: Mandatory

27.9.4 CIM_StorageVolume

StorageVolume as defined in Block Services.

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

27.9.5 CIM_SystemDevice

Associates top level system from Array, Virtualizer, ... to StorageVolume.

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

Table 575 describes class CIM_SystemDevice.

Table 575 - SMI Referenced Properties/Methods for CIM_SystemDevice

Properties	Flags	Requirement	Description & Notes
PartComponent		Mandatory	
GroupComponent		Mandatory	

27.9.6 SNIA_StorageConfigurationCapabilities

SupportedStoragePoolFeatures as defined in SNIA_StorageConfigurationCapabilities, with the addition of support for StorageVolumes as inputs to pool creation.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 576 describes class SNIA_StorageConfigurationCapabilities.

Table 576 - SMI Referenced Properties/Methods for SNIA_StorageConfigurationCapabilities

Properties	Flags	Requirement	Description & Notes
SupportedStoragePoolFeatures		Mandatory	Lists the types of storage elements that are supported by pool creation/modification. To support Pools from Volumes, this list shall include 5 (StorageVolumes).

27.9.7 SNIA_StoragePool

StoragePool as defined in Block Services.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

27.9.8 SNIA_StorageSetting

StorageSetting as defined in Block Services.

Created By: Extrinsic: StorageCapabilities.CreateSetting

Modified By: Static

Deleted By: Static

Requirement: Optional

EXPERIMENTAL

EXPERIMENTAL

Clause 28: Group Masking and Mapping Profile

28.1 Description

28.1.1 Synopsis

Profile Name: Group Masking and Mapping Profile

Version: 1.5.0

Organization: SNIA

CIM schema version: 2.23

Central Class: GroupMaskingMappingService

Scoping Class: ComputerSystem

Table 577 describes the supported profiles for Group Masking and Mapping Profile.

Table 577 - Supported Profiles for Group Masking and Mapping Profile

Profile Name	Organization	Version	Requirement	Description
Generic Target Ports	SNIA	1.4.0	Mandatory	
Block Services	SNIA	1.6.0	Mandatory	
Job Control	SNIA	1.5.0	Optional	

28.1.2 Overview

The Group Masking and Mapping Profile specializes Clause 18: "Masking and Mapping Subprofile". The Group Masking and Mapping Profile is a component profile (subprofile) that allows the masking and mapping operations based on groups of initiator ports (StorageHardwareIDs), target ports, and devices. The profile contains the necessary methods to manipulate masking groups and create or delete masking “views”. Additionally, the group features are advertised by the instance of GroupMaskingMappingCapabilities.

Because the Group Masking and Mapping Profile is specialization of Clause 18: "Masking and Mapping Subprofile", all the classes of Clause 18 (including properties, methods, indications, and capabilities) are inherited (and are available) in this profile.

A masking view created in this profile is modeled as SCSIProtocolController. This is consistent with the views created by methods of the Masking and Mapping Subprofile.

A major goal of the profile is to simplify the masking and mapping operations as much as it is possible. For example, once a masking view is created, to expose additional volumes to the initiators of the masking view, all a client needs to do is to add the additional volumes to the device group belonging to the masking view. Similarly, to remove access to one or more volumes exposed through a masking view, the client simply removes the volumes from the device group associated with the masking view.

The Group Masking and Mapping Profile facilitates provisioning of storage to clustered systems by exposing a group of storage volumes to the same host systems connected to the storage array.

28.1.3 Model Elements

In addition to the model elements inherited from Clause 18: "Masking and Mapping Subprofile", this profile uses the following classes and associations:

MaskingGroup - This class represents a collection of storage masking objects, such as a group of InitiatorPorts, TargetPorts or Volumes (i.e., Devices). The masking group has properties to facilitate "self cleaning" of the groups no longer in use. For example,

- DeleteOnEmpty -- delete the group if all its members are removed.
- DeleteWhenBecomesUnassociated -- delete the group if it no longer is associated to a view.

InitiatorMaskingGroup - A class inherited from MaskingGroup. It represents a collection of StorageHardwareID object paths.

TargetMaskingGroup - A class inherited from MaskingGroup. It represents a collection of ProtocolEndpoint object paths.

DeviceMaskingGroup - A class inherited from MaskingGroup. It represents a collection of StorageVolume object paths.

An implementation may allow empty masking groups to be associated to a masking view; however, an empty associated masking group may indicate “no access” to the elements associated with the masking view. For example, an empty associated InitiatorMaskingGroup indicates none of the initiators have access to the LogicalDevices associated to the masking view. Refer to the group capabilities in 28.7 "CIM Elements", which may indicate “Associated empty group indicates no access“. The absence of this value conversely indicates “all access”. In other words, an empty associated InitiatorMaskingGroup indicates all initiators have access to the elements associated with the masking view.

The masking groups are associated to the “view” via the following associations:

AssociatedInitiatorMaskingGroup - Associates an InitiatorMaskingGroup to a SCSIProtocolController.

AssociatedTargetMaskingGroup - Associates an TargetMaskingGroup to a SCSIProtocolController.

AssociatedDeviceMaskingGroup - Associates an DeviceMaskingGroup to a SCSIProtocolController.

Figure 144 depicts the complete model for a masking view that includes the masking groups. The gray classes are from this profile; whereas, the remaining classes are from Clause 18: "Masking and Mapping Subprofile".

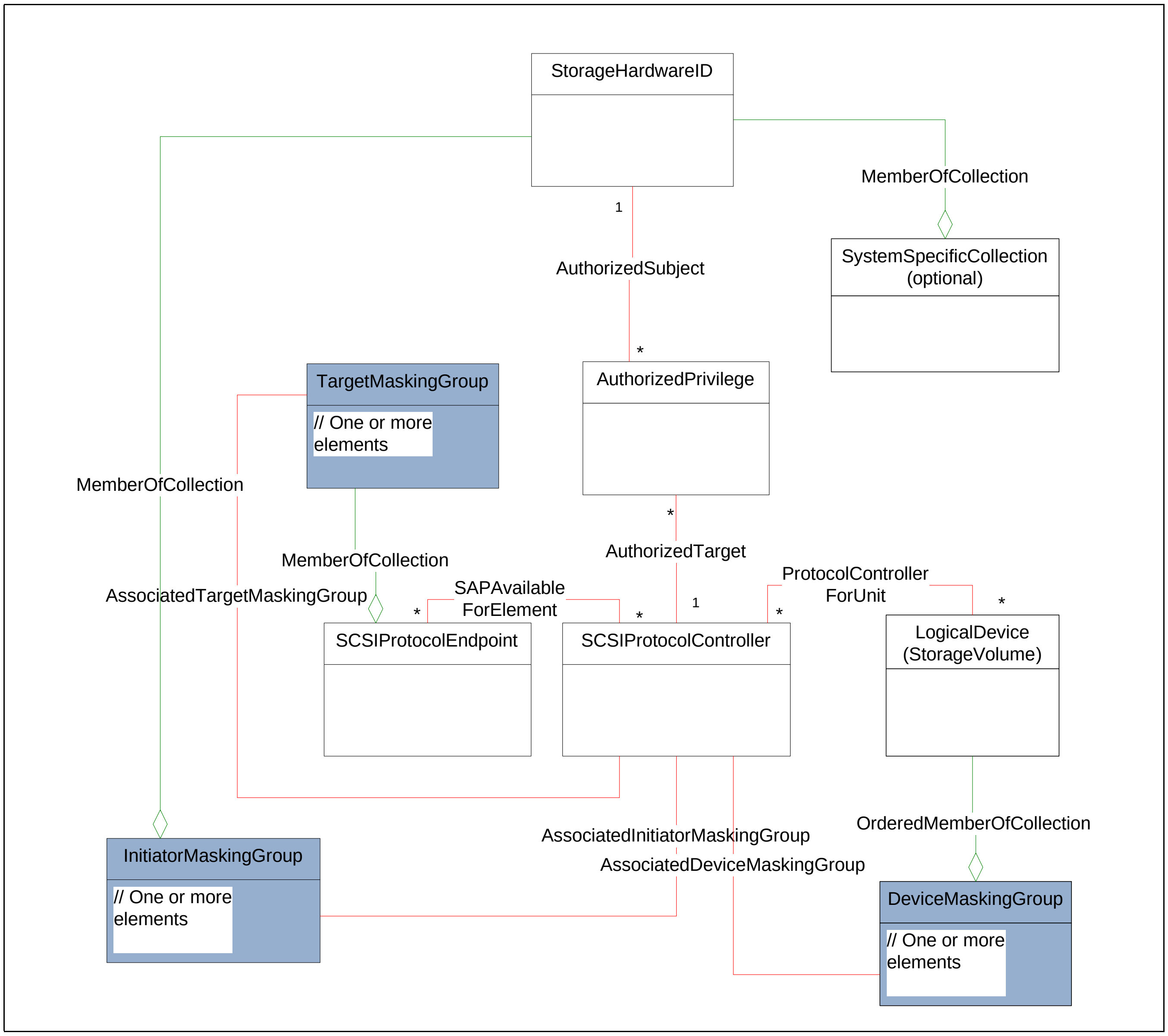

Figure 144 - Group Masking and Mapping Model

Figure 145 shows the masking groups and their associated masking objects. The association between the DeviceMaskingGroup and StorageVolumes is OrderedMemberOfCollections because the method CreateGroup, which creates this association, needs to maintain the order of the StorageVolumes as each StorageVolume is assigned a unique device number. Assigning unique device numbers may be done when a device masking group is created or when a masking view is created. If device numbers are supplied, the implementation shall assign the appropriate device numbers in the order in which the devices are ordered in the device masking group, hence the requirement to have an OrderedMemberOfCollection association between the logical devices and the device masking group.

A SCSIProtocolController shall be associated to no more than one InitiatorMaskingGroup, one TargetMaskingGroup, and one DeviceMaskingGroup. If any of the masking groups is nested, the child groups are indirectly participating in the masking view. However, the nested masking groups are not associated directly to the same masking view.

The profile includes the optional indications for when a masking group is created, deleted, or modified. Additionally, the profile includes an alert message indicating that the associated membership of a masking group has changed. The related standard message is:

```
There is a change in membership of masking group with identifier <InstanceID>.
```

See 28.7 "CIM Elements" for the supported indication filters.

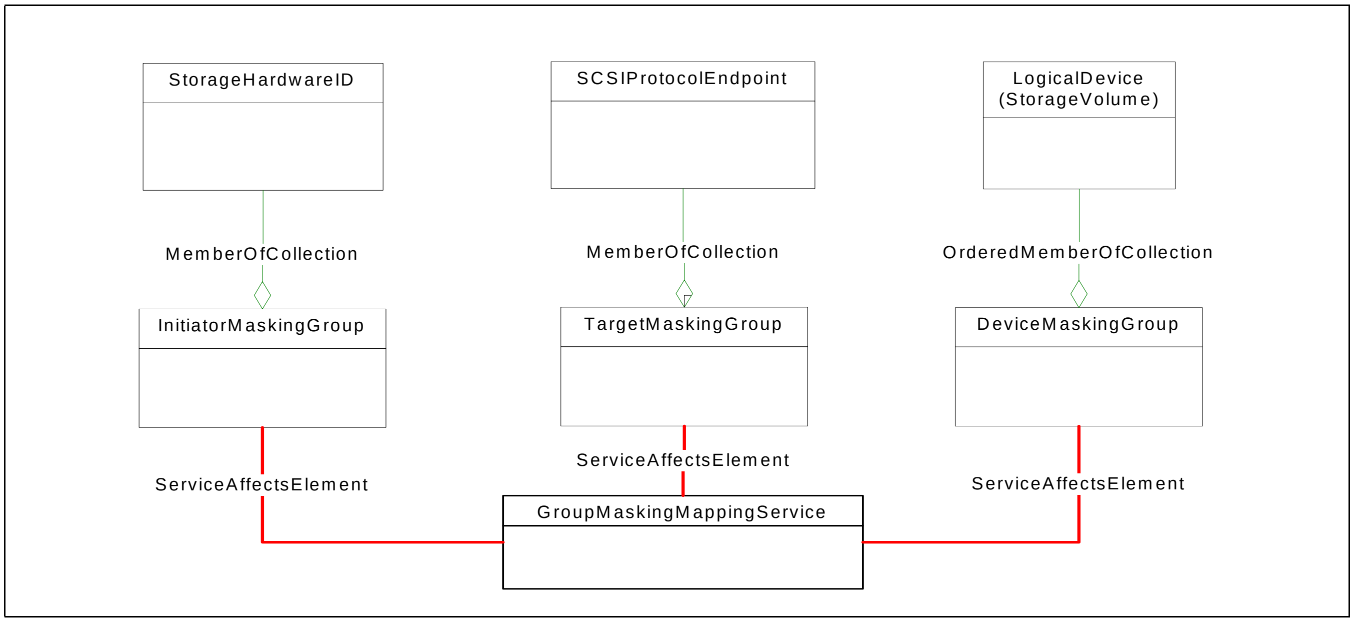

Figure 145 - Masking Groups

28.1.3.1 Nested Groups

Masking groups, depending on the capabilities of the implementation, may be nested to an arbitrary depth. The nested groups shall be of the same type, for example, nested initiator masking groups, or nested target port groups. A masking group may contain a combination of masking objects (initiators, target ports, devices), and the like masking groups. For example, a "top level" initiator masking group may contain zero or more StorageHardwareIDs and zero or more initiator masking groups. The "nested" initiator masking groups may in turn contain additional StorageHardwareIDs and initiator masking groups.

To create a masking view, a client may supply the "top level" masking group and the appropriate target port and device masking groups to the CreateMaskingView method.

See the instance of GroupMaskingMappingCapabilities (in 28.7 "CIM Elements") for whether the implementation supports nested masking groups, and whether the depth of nested groups is limited to one.

Figure 146 shows nest masking groups and an example of a nested initiator masking group.

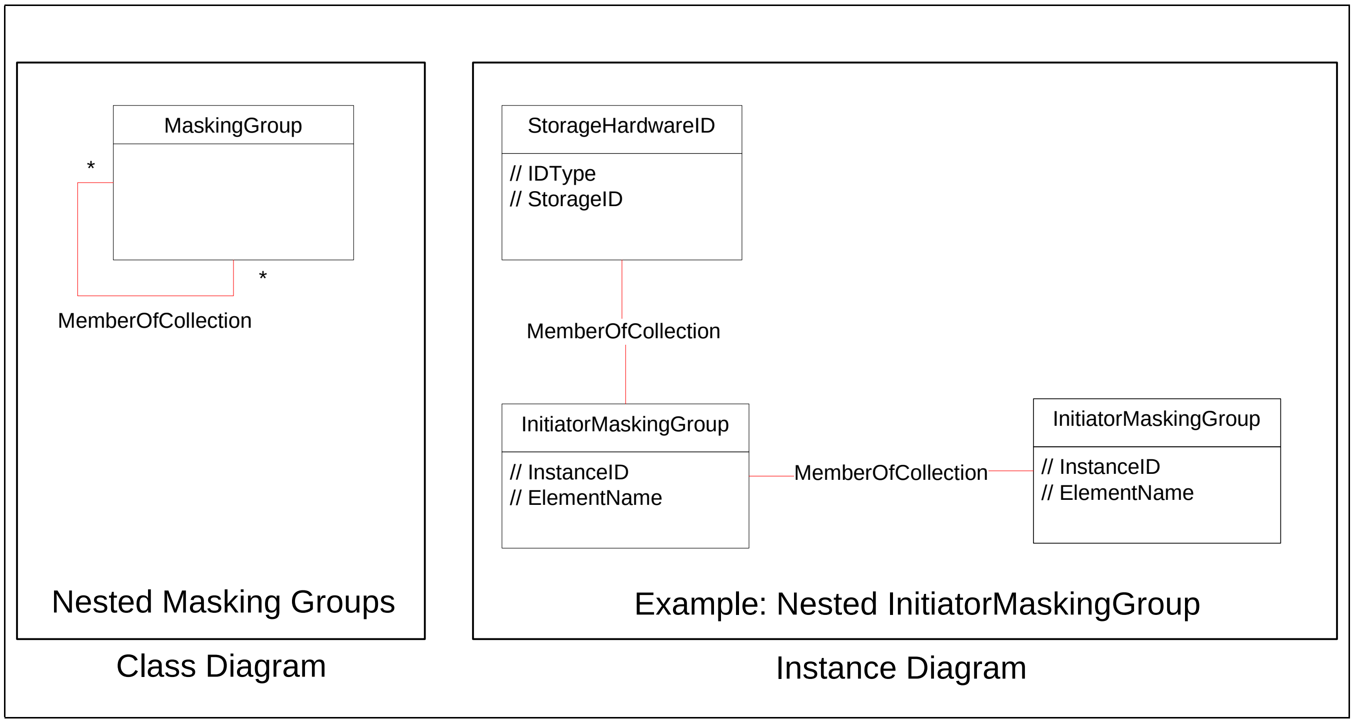

Figure 146 - Nested Masking Groups

A use case for nested group is in a simple cluster environment (see Figure 147, “Nested Masking Group Example,”), where there is a set of HBAs that belong to one host and a set of HBAs that belong to another host. Assume each host’s HBA ports are in their own Initiator Masking Group which is participating in some other masking views. A nested parent group (Engineering, in the example) is then created to contain both of these child groups (HBA0 and HBA1, in the example). This new parent group can then be directly associated to a new masking view, furthermore, the child groups still remain associated to some other masking views.

In this example, when the Engineering InitiatorMaskingGroup is associated to a new masking view, the child groups HBA0 and HBA1 are indirectly participating in the new masking view, however, they are not associated directly to the new masking view. Note that HBA0 and HBA1 will have access to the devices exposed to the Engineering InitiatorMaskingGroup by the masking view.

With nested masking groups, only the outer (i.e., the parent) initiator masking group needs to be associated to a masking view. The inner (i.e., the children) initiator masking groups will implicitly have access to the same storage devices made available in the masking view associated to the parent. However, the inner masking groups (i.e., the children) can be associated to a different masking view in order to have access to storage devices participating in a different masking view.

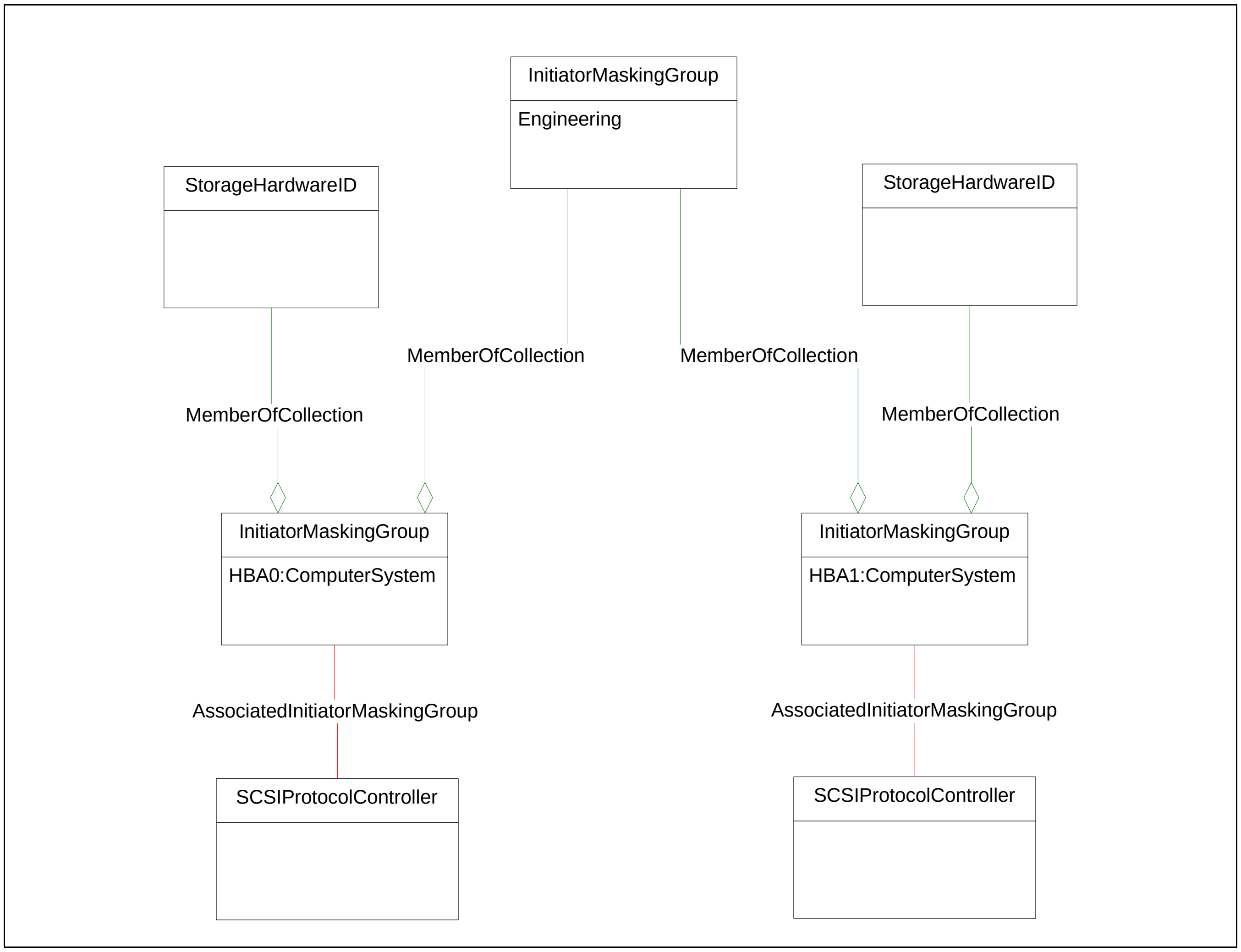

Figure 147 - Nested Masking Group Example

28.1.4 Device Numbers

A LogicalDevice is exposed to an initiator with a Device Number, also known as a Logical Unit Number (LUN). Clients may supply the desired Logical Unit Numbers. If clients do not supply the desired Logical Units Numbers, the instrumentation decides on the Logical Unit Numbers. There is a boolean property in the InitiatorMaskingGroup class called ConsistentLogicalUnitNumber to indicate whether device numbers for a LogicalDevice (volume) visible to the same initiator must be same.

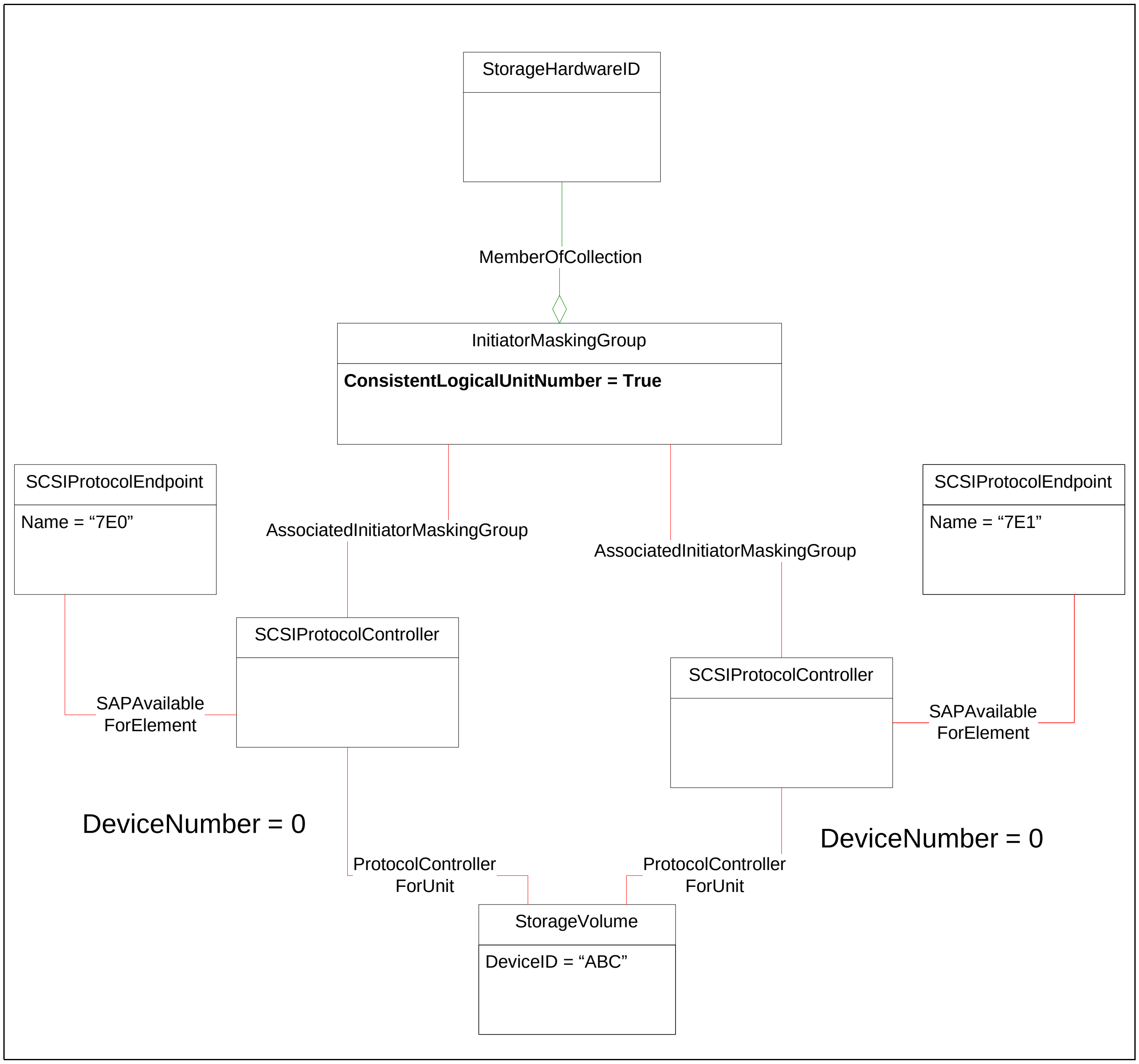

Figure 148 - Example ConsistentLogicalUnitNumber set to true

Figure 148 shows an example configuration with the InitiatorMaskingGroup.ConsistentLogicalUnitNumber property set to true. In this case, the storage volume “ABC” shall have the same DeviceNumber value exposed to the same initiator regardless of the path.

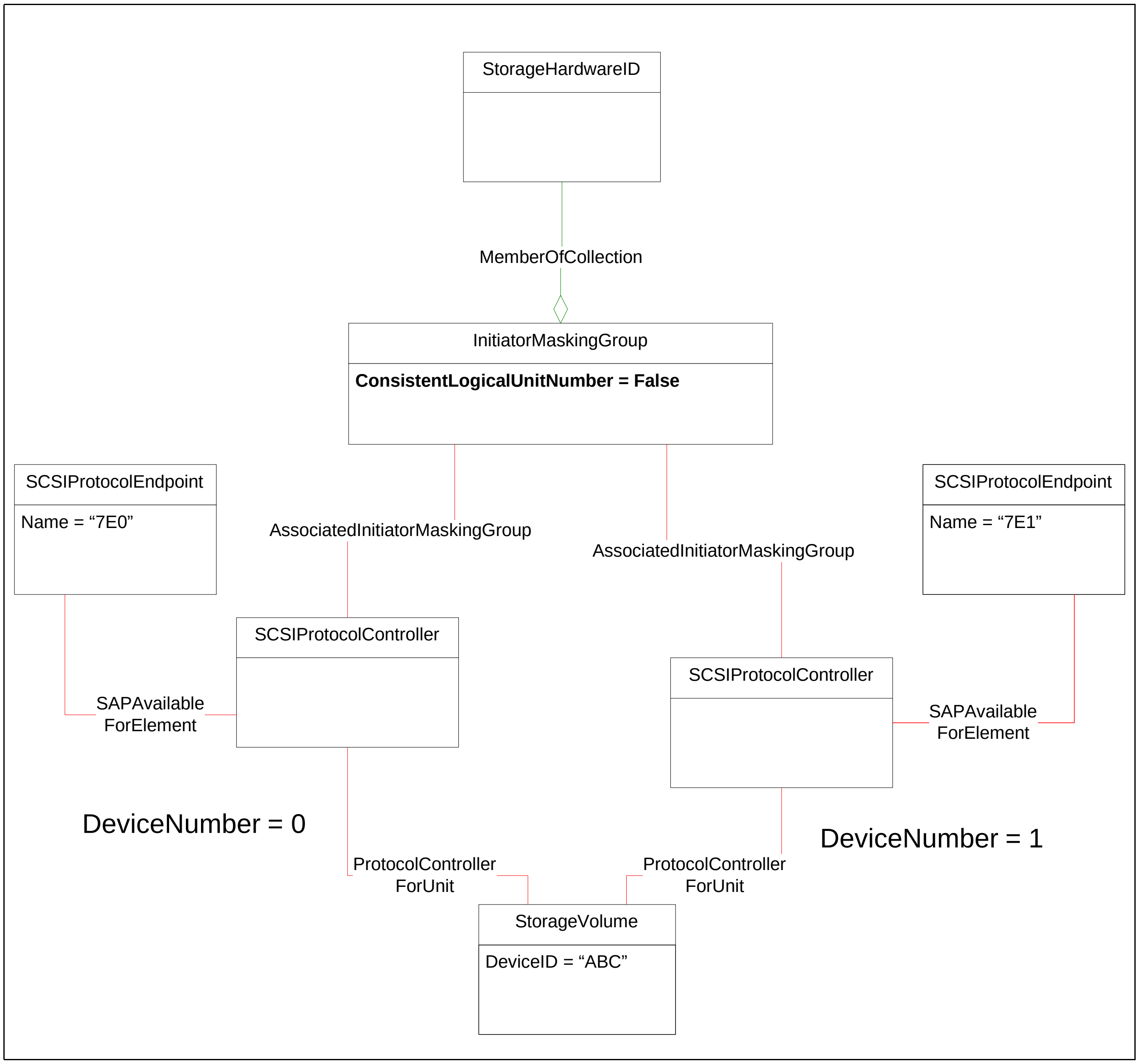

Figure 149 - Example ConsistentLogicalUnitNumber set to false

Figure 149 shows an example configuration with the InitiatorMaskingGroup.ConsistentLogicalUnitNumber property set to false. In this case, depending on the path, the storage volume “ABC” may have different DeviceNumber values exposed to the same initiator.

If the instrumentation only supports ConsistentLogicalUnitNumber, the capabilities method SupportedInitiatorGroupFeatures shall indicate “ConsistentLogicalUnitNumber must be true“. In this case, clients can not create an InitiatorMaskingGroup with the value of the property ConsistentLogicalUnitNumber set to false.

28.1.5 Group Masking and Mapping Capabilities

The class GroupMaskingMappingCapabilities contains the properties that advertise the capabilities of the group masking and mapping implementation. For example, the property SupportedFeatures indicates capabilities of a masking view that uses groups, and the property SupportedInitiatorGroupFeatures indicates the capabilities specific to an initiator group.

Refer to 28.7 "CIM Elements" for all the capabilities details.

28.2 Health and Fault Management Consideration

None

28.3 Cascading Considerations

None

28.4 Methods of the Profile

The Group Masking and Mapping Profile has extrinsic methods for group management and for managing masking view.

The Profile relies on a number of intrinsic methods ModifyInstance and DeleteInstance for changing the property values and deleting instances and that do not require special consideration such as the “force” option.

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 578 summarizes the extrinsic methods for group management (class GroupMaskingMappingService):

Table 578 - Extrinsic Methods for Masking Group Management

Method	Described in
CreateGroup	See 28.4.1
DeleteGroup	See 28.4.2
AddMembers	See 28.4.3
RemoveMembers	See 28.4.4

Table 579 summarizes the extrinsic methods for creating and deleting group masking views (class GroupMaskingMappingService):

Table 579 - Extrinsic Methods for Masking Views Management

Method	Described in
CreateMaskingView	See 28.4.5

Table 579 - Extrinsic Methods for Masking Views Management

Method	Described in
DeleteMaskingView	See 28.4.6
ModifyMaskingView	See 28.4.7

28.4.1 CreateGroup

```
uint32 GroupMaskingMappingService.CreateGroup(
      [IN]  string GroupName,
      [IN]  uint16 Type,
      [IN]  CIM_ManagedElement REF Members[],
      [OUT] CIM_ConcreteJob REF Job,
      [IN]  boolean DeleteOnEmpty,
      [IN]  boolean DeleteWhenBecomesUnassociated,
      [IN]  boolean ConsistentLogicalUnitNumber,
      [OUT] CIM_MaskingGroup REF MaskingGroup);
```

This method allows a client to create a new masking group. Any required associations (such as ServiceAffectsElement) are created in addition to the instance of the group. The parameters are as follows:

- GroupName: If nameable, an end user relevant name for the group being created. If NULL or not nameable, then system assigns a name. If nameable, the name shall be unique for given ComputerSystem. If not nameable and a group name is supplied, the method returns an error and aborts the method call.
- Type: The type of masking group to create. Possible choices are InitiatorMaskingGroup, TargetMaskingGroup, and DeviceMaskingGroup. Any other type or a masking group type not supported by the instrumentation are rejected.
- Members[]: A list of elements to add to the masking group. For device masking groups the order is maintained. If NULL, the group shall be empty -- if empty groups are supported. All the supplied elements shall be of type appropriate for the type of masking group being created.
- 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).
- DeleteOnEmpty: If true, the group shall be deleted when the last element is removed from the group. If false, the group shall not be deleted when the last element is removed from the group. If an implementation does not allow empty groups, the group shall be deleted when it becomes empty regardless of the value of this parameter. See the GetSupported*GroupFeatures() method of the GroupMaskingMappingCapabilities to determine whether empty groups are allowed.
- DeleteWhenBecomesUnassociated: If true, the group shall be deleted when the group is no longer associated to any SCSIProtocolController (i.e., a masking view).
- ConsistentLogicalUnitNumber: If true, it indicates the device numbers for a volume visible to the same initiator through different paths must be same.
- MaskingGroup: A reference to the created group.

28.4.2 DeleteGroup

```
uint32 GroupMaskingMappingService.DeleteGroup(
      [IN, Required]  CIM_MaskingGroup REF MaskingGroup,
      [OUT]  CIM_ConcreteJob REF Job,
      [IN]   boolean Force );
```

This method allows a client to delete a masking group. Deleting a masking group does not delete its associated members. The parameters are as follows:

- MaskingGroup: Reference to a masking group which would be deleted.
- 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).
- Force: Attempt to delete the masking group even though it is associated to a masking view, or the group is not empty. The intent of the Force parameter is to reduce the chance of accidental deletion of a masking group.

28.4.3 AddMembers

```
            uint32 GroupMaskingMappingService.AddMembers(
                  [IN, Required]  CIM_MaskingGroup REF MaskingGroup,
                  [IN]  CIM_ManagedElement REF Members[],
                  [IN]  string DeviceNumbers[],
                  [OUT] CIM_ConcreteJob REF Job  );
```

This method allows a client to add members to an existing masking group. The parameters are as follows:

- MaskingGroup: Reference to an existing masking group.
- Members[]: List of elements to add to the group. New members are added, in the order supplied, to the end of the existing members of the group. It is not an error, if a new member is already in the group. All the supplied elements shall be of type appropriate for the type of masking group supplied.
- DeviceNumbers[]: List of device numbers that correspond to Members. This property is applicable when the group consists of storage volumes.
- 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).

28.4.4 RemoveMembers

```
            uint32 GroupMaskingMappingService.RemoveMembers(
                  [IN, Required]  CIM_MaskingGroup REF MaskingGroup,
                  [IN]   CIM_ManagedElement REF Members[],
                  [OUT]  CIM_ConcreteJob REF Job,
                  [IN]   boolean DeleteOnEmpty );
```

This method allows a client to remove members from a masking group. The parameters are as follows:

- MaskingGroup: Reference to an existing masking group.
- Members[]: List of elements to remove from the group. Deleting all members of a group is equivalent to deleting the group if empty groups are not supported by the implementation.
- 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).
- DeleteOnEmpty: If true and removal of the members causes the group to become empty, the group shall be deleted. Note, if empty groups are not allowed, the group shall be deleted automatically when the group becomes empty. If this parameter is not NULL, it overrides the group's property DeleteOnEmpty.

28.4.5 CreateMaskingView

```
uint32 GroupMaskingMappingService.CreateMaskingView(
      [IN]  string ElementName,
      [IN]  CIM_MaskingGroup REF InitiatorMaskingGroup,
      [IN]  CIM_MaskingGroup REF TargetMaskingGroup,
      [IN]  CIM_MaskingGroup REF DeviceMaskingGroup,
      [IN]  string DeviceNumbers[],
      [OUT] CIM_ConcreteJob REF Job,
      [OUT] CIM_SCSIProtocolController REF ProtocolController);
```

This method allows a client to expose a group of SCSI logical units (such as RAID volumes or tape drives) to a group of initiators through a group of target ports, through one or more SCSIProtocolControllers (SPCs). If 0 is returned, the function completed successfully and no ConcreteJob instance is created. If 4096/0x1000 is returned, a ConcreteJob is started, a reference to which is returned in the Job output parameter. The parameters are as follows:

- ElementName: A user relevant name for the masking view. If NULL, the implementation assigns a name.
- InitiatorMaskingGroup: Reference to a group of StorageHardwareIDs.
- TargetMaskingGroup: Reference to a group of SCSIProtocolEndpoints.
- DeviceMaskingGroup: Reference to a group of StorageVolumes.
- DeviceNumbers[]: List of device numbers that correspond to the elements of DeviceMaskingGroup. If this parameter is NULL, device numbers are assigned by the instrumentation.
- 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).
- ProtocolController: A reference to the created SCSIProtocolController, which represents the masking view.

28.4.6 DeleteMaskingView

```
uint32 GroupMaskingMappingService.DeleteMaskingView(
      [IN, Required]  CIM_SCSIProtocolController REF ProtocolController,
      [OUT]  CIM_ConcreteJob REF Job,
      [IN]   boolean DeleteWhenBecomesUnassociated );
```

This method allows a client to delete a masking view, i.e., a SCSIProtocolController. Deleting a masking view may also delete the associated masking groups -- see the applicable capabilities and group properties in 28.7 "CIM Elements". The parameters are as follows:

- ProtocolController: A reference to the SCSIProtocolController to delete. The masking group associated with the view may also get deleted depending on the groups' applicable properties.
- 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).
- DeleteWhenBecomesUnassociated: Override the setting of the masking groups' property DeleteWhenBecomesUnassociated with the value of this parameter.

28.4.7 ModifyMaskingView

```
uint32 GroupMaskingMappingService.ModifyMaskingView(
      [IN, Required]   uint16 Operation,
      [IN, Required]   CIM_SCSIProtocolController REF ProtocolController,
      [IN]   CIM_MaskingGroup REF MaskingGroup,
      [IN]   string DeviceNumbers[],
      [OUT]  CIM_ConcreteJob REF Job,
      [IN]   Force );
```

This method allows a client to modify a masking view by adding a masking group or by removing a masking group from the masking view. The parameters are as follows:

- Operation: It describes the type of modification to be made to the masking view. Possible values: "Add Group", "Remove Group", and “Replace Group”. Adding a masking group to a masking view which already is associated to the same type of masking group is an error condition. For example, if a masking view is already associated to an InitiatorMaskingGroup, attempting to add another InitiatorMaskingGroup to the same masking view results in an error return (or an exception is thrown). The “Replace Group” operation replaces an existing masking group of the same type (i.e., Initiator, Target Port, or Device). However, if the masking view is not already associated to a masking group of the type supplied, the instrumentation shall create the appropriate association between the supplied masking view and masking group; in other words, the “Replace Group” operation behaves the same as the “Add Group” operation.
- MaskingGroup: A reference to the masking group.
- DeviceNumbers: This parameter applies to an "Add Group" operation. It is a list of device numbers that correspond to the elements of a DeviceMaskingGroup. If device numbers are not supplied, the instrumentation may assign the appropriate device numbers to the supplied logical devices.
- 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).
- Force: If true, the client is not warned that the operation may render the masking view unusable.

28.5 Client Considerations and Recipes

28.5.1 Using Groups in Masking and Mapping

In general, the Masking and Mapping operations using groups involve the following steps:

- Create the masking groups (initiators, target port, and storage volumes), using the CreateGroup method call.
- Create the masking view using the CreateMaskingView method call.

Depending on the implementation, it may be necessary to supply DeviceNumbers when creating a DeviceMaskingGroup or the actual masking view -- refer to the group capabilities (in 28.7 "CIM Elements"). If DeviceNumbers are not required, the implementation shall assign the appropriate device numbers.

Once a masking view is created, to expose additional storage volumes to the same initiator ports, the client only needs to add the additional storage volumes to the DeviceMaskingGroup using the AddMembers method call. Alternatively, to remove access to certain storage volumes exposed through a masking view, the client needs only to use the RemoveMembers method call to removed the intended storage volumes from the DeviceMaskingGroup associated with the masking view.

An implementation may initially allow a client to create a masking view with fewer than all three masking groups (initiators, target ports, and devices) or even empty masking groups (see the capabilities in 28.7 "CIM Elements" to determine which groups are required for the creation of a masking view). Subsequently, the client may use the appropriate methods (ModifyMaskingView) to add the necessary masking groups and/or to add members (AddMembers) to the empty masking groups.

Assuming the methods ExposePaths and HidePaths methods are supported by the implementation, changes made to an masking view via the ExposePaths and HidePaths methods shall appear correctly to a client using the Group Masking and Mapping Profile. For example, if a client utilizes the HidePaths method to remove a device associated to a masking view, the instrumentation shall remove the device from the device masking group associated to the same masking view. However, if the device masking group is associated to multiple masking views, the instrumentation return an error. Similarly, if a client utilizes the AddMembers method to add a device to a device masking group associated to an existing masking view, the end result shall be as if the client used the ExposePaths method to expose the device. In summary, any changes made to a masking view by a 1.5 client shall appear correct to the pre-1.5 client and vice versa.

28.6 Registered Name and Version

Group Masking and Mapping Profile version 1.5.0 (Component Profile)

CIM Schema Version: 2.23.0

Specializes SNIA Masking and Mapping version 1.6.0

28.7 CIM Elements

Table 580 describes the CIM elements for Group Masking and Mapping Profile.

Table 580 - CIM Elements for Group Masking and Mapping Profile

Element Name	Requirement	Description
28.7.1 CIM_AssociatedDeviceMaskingGroup	Conditional	Conditional requirement: Required if device masking groups are supported. Associates SCSIProtocolController to an DeviceMaskingGroup.
28.7.2 CIM_AssociatedInitiatorMaskingGroup	Conditional	Conditional requirement: Required if initiator masking groups are supported. Associates SCSIProtocolController to an InitiatorMaskingGroup.
28.7.3 CIM_AssociatedPrivilege	Mandatory	
28.7.4 CIM_AssociatedTargetMaskingGroup	Conditional	Conditional requirement: Required if target masking groups are supported. Associates SCSIProtocolController to a TargetMaskingGroup.
28.7.5 CIM_AuthorizedPrivilege	Mandatory	Deprecated.
28.7.6 CIM_AuthorizedSubject	Mandatory	Deprecated.
28.7.7 CIM_AuthorizedTarget	Mandatory	Deprecated.
28.7.8 CIM_ConcreteDependency (Associates ControllerConfigiurationService and ProtocolController)	Mandatory	
28.7.9 CIM_ConcreteDependency (Associates PrivilegeManagementService and AuthorizedPrivilege)	Mandatory	Deprecated.

Table 580 - CIM Elements for Group Masking and Mapping Profile

Element Name	Requirement	Description
28.7.10 CIM_ConcreteDependency (Associates StorageHardwareIDManagementService and StorageHardwareID)	Mandatory	
28.7.11 CIM_ConcreteDependency (Associates StorageHardwareIDManagementService and SystemSpecificCollection)	Conditional	Conditional requirement: Implementation support for collections of StorageHardwareIDs.
28.7.12 CIM_DeviceMaskingGroup	Mandatory	Represents a group of Devices (StorageVolumes).
28.7.13 CIM_ElementCapabilities (EnabledLogicalElementCapabilities to ControllerConfigurationService)	Optional	Associates EnabledLogicalElementCapabilities with ControllerConfigurationService.
28.7.14 CIM_ElementCapabilities (EnabledLogicalElementCapabilities to ProtocolController)	Optional	Expressed the ability for the element to be named or have its state changed.
28.7.15 CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StorageHardwareID)	Optional	Associates EnabledLogicalElementCapabilities to StorageHardwareID.
28.7.16 CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StorageHardwareIDManagementService)	Optional	Associates EnabledLogicalElementCapabilities with StorageHardwareIDManagementService.
28.7.17 CIM_ElementCapabilities (EnabledLogicalElementCapabilities to SystemSpecificCollection)	Conditional	Conditional requirement: Implementation support for collections of StorageHardwareIDs. Associates EnabledLogicalElementCapabilities and SystemSpecificCollection.
28.7.18 CIM_ElementCapabilities (System to ProtocolControllerMaskingCapabilities)	Mandatory	
28.7.19 CIM_ElementSettingData (Associates ComputerSystem and StorageClientSettingData)	Mandatory	
28.7.20 CIM_ElementSettingData (Associates Port and StorageClientSettingData)	Optional	
28.7.21 CIM_ElementSettingData (Associates ProtocolController and StorageClientSettingData)	Optional	
28.7.22 CIM_ElementSettingData (Associates StorageHardwareID and StorageClientSettingData)	Optional	
28.7.23 CIM_EnabledLogicalElementCapabilities	Optional	This class is used to express the naming and possible requested state change possibilities for storage elements.

Table 580 - CIM Elements for Group Masking and Mapping Profile

Element Name	Requirement	Description
28.7.24 CIM_GroupMaskingMappingCapabilities	Mandatory	A set of properties that describe the capabilities of a group masking and mapping provider.
28.7.25 CIM_GroupMaskingMappingService	Mandatory	Central class for Group Masking and Mapping Profile. Methods are described in the Extrinsic Methods clause.
28.7.26 CIM_HostedCollection	Conditional	Conditional requirement: Implementation support for collections of StorageHardwareIDs.
28.7.27 CIM_HostedService (Associates ComputerSystem and ControllerConfigurationService)	Mandatory	
28.7.28 CIM_HostedService (Associates ComputerSystem and PrivilegeManagementService)	Mandatory	Deprecated.
28.7.29 CIM_HostedService (Associates ComputerSystem and StorageHardwareIDManagementService)	Mandatory	
28.7.30 CIM_InitiatorMaskingGroup	Mandatory	Represents a group of initiator ports (StorageHardwareIDs).
28.7.31 CIM_MemberOfCollection	Conditional	Conditional requirement: Implementation support for collections of StorageHardwareIDs.
28.7.32 CIM_PrivilegeManagementService	Mandatory	Deprecated.
28.7.33 CIM_ProtocolController	Mandatory	
28.7.34 CIM_ProtocolControllerForUnit	Mandatory	
28.7.35 CIM_SAPAvailableForElement	Mandatory	
28.7.36 CIM_ServiceAffectsElement (Between GroupMaskingMappingService and MaskingGroup)	Conditional	Conditional requirement: Required if device masking groups are supported or Required if initiator masking groups are supported or Required if target masking groups are supported. Associates Group Masking Mapping Service to Masking Group.
28.7.37 CIM_StorageClientSettingData	Mandatory	
28.7.38 CIM_StorageHardwareID	Mandatory	
28.7.39 CIM_StorageHardwareIDManagementService	Mandatory	
28.7.40 CIM_SystemSpecificCollection	Conditional	Conditional requirement: Implementation support for collections of StorageHardwareIDs.

Table 580 - CIM Elements for Group Masking and Mapping Profile

Element Name	Requirement	Description
28.7.41 CIM_TargetMaskingGroup	Mandatory	Represents a group of target ports (ProtocolEndpoints).
28.7.42 SNIA_ProtocolControllerMaskingCapabilities	Optional	An experimental subclass of CIM_ProtocolControllerMaskingCapabilities.
28.7.43 SNIA_StorageHardwareID	Optional	Experimental SNIA class adding SAS Address IDs.
28.7.44 SNIA_StorageHardwareIDManagementService	Optional	Experimental subclass with support for SAS StorageHardwareIDs.
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_AssociatedPrivilege	Mandatory	Creation of an AssociatedPrivilege association.
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_InstDeletion WHERE SourceInstance ISA CIM_AssociatedPrivilege	Mandatory	Deletion of an AssociatedPrivilege association.
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_AuthorizedSubject	Mandatory	Deprecated. Creation of an AuthorizedSubject association.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_AuthorizedSubject	Mandatory	Deprecated. Deletion of an AuthorizedSubject association.
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_MaskingGroup	Optional	Creation of a MaskingGroup.
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_MaskingGroup	Optional	Deletion of a MaskingGroup.
SELECT * FROM CIM_InstModification WHERE SourceInstance ISA CIM_MaskingGroup	Optional	Modification of properties of a MaskingGroup.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity='SNIA' and MessageID='DRM31'	Optional	There is a change in the membership of a masking group.

28.7.1 CIM_AssociatedDeviceMaskingGroup

Associates SCSIProtocolController to an DeviceMaskingGroup.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Required if device masking groups are supported.

Table 581 describes class CIM_AssociatedDeviceMaskingGroup.

Table 581 - SMI Referenced Properties/Methods for CIM_AssociatedDeviceMaskingGroup

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	SCSIProtocolController.
Dependent		Mandatory	DeviceMaskingGroup.

28.7.2 CIM_AssociatedInitiatorMaskingGroup

Associates SCSIProtocolController to an InitiatorMaskingGroup.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Required if initiator masking groups are supported.

Table 582 describes class CIM_AssociatedInitiatorMaskingGroup.

Table 582 - SMI Referenced Properties/Methods for CIM_AssociatedInitiatorMaskingGroup

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	SCSIProtocolController.
Dependent		Mandatory	InitiatorMaskingGroup.

28.7.3 CIM_AssociatedPrivilege

Created By: Extrinsics: CIM_ControllerConfigurationService.ExposePaths, CIM_ControllerConfigurationService.HidePaths

Modified By: Extrinsics: CIM_ControllerConfigurationService.ExposePaths, CIM_ControllerConfigurationService.HidePaths

Deleted By: Extrinsics: CIM_ControllerConfigurationService.ExposePaths, CIM_ControllerConfigurationService.HidePaths

Requirement: Mandatory

Table 583 describes class CIM_AssociatedPrivilege.

Table 583 - SMI Referenced Properties/Methods for CIM_AssociatedPrivilege

Properties	Flags	Requirement	Description & Notes
UseKey		Mandatory	Opaque and unique identifer.
PrivilegeGranted		Mandatory	Indicates if the privilege is granted or not.
Activities		Mandatory	For SMI-S, shall be 5,6 ('Read' and Write').
Subject		Mandatory	The StorageHardwareID having access to a target element.
Target		Mandatory	The ProtocolController accessible to subject element.

28.7.4 CIM_AssociatedTargetMaskingGroup

Associates SCSIProtocolController to an TargetMaskingGroup.

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Required if target masking groups are supported.

Table 584 describes class CIM_AssociatedTargetMaskingGroup.

Table 584 - SMI Referenced Properties/Methods for CIM_AssociatedTargetMaskingGroup

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	SCSIProtocolController.
Dependent		Mandatory	TargetMaskingGroup.

28.7.5 CIM_AuthorizedPrivilege

Deprecated.

Created By: Extrinsics: CIM_ControllerConfigurationService.ExposePaths, CIM_ControllerConfigurationService.HidePaths

Modified By: Extrinsics: CIM_ControllerConfigurationService.ExposePaths, CIM_ControllerConfigurationService.HidePaths

Deleted By: Extrinsics: CIM_ControllerConfigurationService.ExposePaths, CIM_ControllerConfigurationService.HidePaths

Requirement: Mandatory

Table 585 describes class CIM_AuthorizedPrivilege.

Table 585 - SMI Referenced Properties/Methods for CIM_AuthorizedPrivilege

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	Opaque and unique identifer.
ElementName		Optional	User friendly name.

Table 585 - SMI Referenced Properties/Methods for CIM_AuthorizedPrivilege

Properties	Flags	Requirement	Description & Notes
PrivilegeGranted		Mandatory	Indicates if the privilege is granted or not.
Activities		Mandatory	For SMI-S, shall be 5,6 ('Read' and Write').

28.7.6 CIM_AuthorizedSubject

Deprecated.

Created By: Extrinsics: CIM_ControllerConfigurationService.ExposePaths, CIM_ControllerConfigurationService.HidePaths

Modified By: Extrinsics: CIM_ControllerConfigurationService.ExposePaths, CIM_ControllerConfigurationService.HidePaths

Deleted By: Extrinsics: CIM_ControllerConfigurationService.ExposePaths, CIM_ControllerConfigurationService.HidePaths

Requirement: Mandatory

Table 586 describes class CIM_AuthorizedSubject.

Table 586 - SMI Referenced Properties/Methods for CIM_AuthorizedSubject

Properties	Flags	Requirement	Description & 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.

28.7.7 CIM_AuthorizedTarget

Deprecated.

Created By: Extrinsics: CIM_ControllerConfigurationService.ExposePaths, CIM_ControllerConfigurationService.HidePaths

Modified By: Extrinsics: CIM_ControllerConfigurationService.ExposePaths, CIM_ControllerConfigurationService.HidePaths

Deleted By: Extrinsics: CIM_ControllerConfigurationService.ExposePaths, CIM_ControllerConfigurationService.HidePaths

Requirement: Mandatory

Table 587 describes class CIM_AuthorizedTarget.

Table 587 - SMI Referenced Properties/Methods for CIM_AuthorizedTarget

Properties	Flags	Requirement	Description & Notes
TargetElement		Mandatory	The target set of resources to which the Privilege applies.
Privilege		Mandatory	The Privilege affecting the target resource.

28.7.8 CIM_ConcreteDependency (Associates ControllerConfiguirationService and ProtocolController)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 588 describes class CIM_ConcreteDependency (Associates ControllerConfiguirationService and ProtocolController).

Table 588 - SMI Referenced Properties/Methods for CIM_ConcreteDependency (Associates ControllerConfiguirationService and ProtocolController)

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	
Antecedent		Mandatory	

28.7.9 CIM_ConcreteDependency (Associates PrivilegeManagementService and AuthorizedPrivilege)

Deprecated.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 589 describes class CIM_ConcreteDependency (Associates PrivilegeManagementService and AuthorizedPrivilege).

Table 589 - SMI Referenced Properties/Methods for CIM_ConcreteDependency (Associates PrivilegeManagementService and AuthorizedPrivilege)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

28.7.10 CIM_ConcreteDependency (Associates StorageHardwareIDManagementService and StorageHardwareID)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 590 describes class CIM_ConcreteDependency (Associates StorageHardwareIDManagementService and StorageHardwareID).

Table 590 - SMI Referenced Properties/Methods for CIM_ConcreteDependency (Associates StorageHardwareIDManagementService and StorageHardwareID)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

28.7.11 CIM_ConcreteDependency (Associates StorageHardwareIDManagementService and SystemSpecificCollection)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Implementation support for collections of StorageHardwareIDs.

Table 591 describes class CIM_ConcreteDependency (Associates StorageHardwareIDManagementService and SystemSpecificCollection).

Table 591 - SMI Referenced Properties/Methods for CIM_ConcreteDependency (Associates StorageHardwareIDManagementService and SystemSpecificCollection)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

28.7.12 CIM_DeviceMaskingGroup

Represents a group of Devices (StorageVolumes).

Created By: Extrinsic: CreateGroup

Modified By: Extrinsics: AddMembers, RemoveMembers

Deleted By: Extrinsic: DeleteGroup

Requirement: Mandatory

Table 592 describes class CIM_DeviceMaskingGroup.

Table 592 - SMI Referenced Properties/Methods for CIM_DeviceMaskingGroup

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	Within the scope of an array, the InstanceID opaquely and uniquely identifies an instance of this class.
DeleteOnEmpty	M	Mandatory	If true and empty groups are allowed, the group will be deleted when the last element is removed from the group. If empty groups are not allowed, the group will be deleted automatically when the group becomes empty.
DeleteWhenBecomesUnassociated	M	Mandatory	If true, the group will be deleted when the group is no longer associated with a masking view. This can happen if all masking views associated to this group are deleted.
ElementName		Optional	User Friendly name.

28.7.13 CIM_ElementCapabilities (EnabledLogicalElementCapabilities to ControllerConfigurationService)

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

Table 593 describes class CIM_ElementCapabilities (EnabledLogicalElementCapabilities to ControllerConfigurationService).

Table 593 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (EnabledLogicalElementCapabilities to ControllerConfigurationService)

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

28.7.14 CIM_ElementCapabilities (EnabledLogicalElementCapabilities to ProtocolController)

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

Table 594 describes class CIM_ElementCapabilities (EnabledLogicalElementCapabilities to ProtocolController).

Table 594 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (EnabledLogicalElementCapabilities to ProtocolController)

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

28.7.15 CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StorageHardwareID)

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

Table 595 describes class CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StorageHardwareID).

Table 595 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StorageHardwareID)

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

28.7.16 CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StorageHardwareIDManagementService)

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

Table 596 describes class CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StorageHardwareIDManagementService).

Table 596 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StorageHardwareIDManagementService)

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

28.7.17 CIM_ElementCapabilities (EnabledLogicalElementCapabilities to SystemSpecificCollection)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Implementation support for collections of StorageHardwareIDs.

Table 597 describes class CIM_ElementCapabilities (EnabledLogicalElementCapabilities to SystemSpecificCollection).

Table 597 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (EnabledLogicalElementCapabilities to SystemSpecificCollection)

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

28.7.18 CIM_ElementCapabilities (System to ProtocolControllerMaskingCapabilities)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 598 describes class CIM_ElementCapabilities (System to ProtocolControllerMaskingCapabilities).

Table 598 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (System to ProtocolControllerMaskingCapabilities)

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	
Capabilities		Mandatory	

28.7.19 CIM_ElementSettingData (Associates ComputerSystem and StorageClientSettingData)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 599 describes class CIM_ElementSettingData (Associates ComputerSystem and StorageClientSettingData).

Table 599 - SMI Referenced Properties/Methods for CIM_ElementSettingData (Associates ComputerSystem and StorageClientSettingData)

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	
SettingData		Mandatory	

28.7.20 CIM_ElementSettingData (Associates Port and StorageClientSettingData)

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

Table 600 describes class CIM_ElementSettingData (Associates Port and StorageClientSettingData).

Table 600 - SMI Referenced Properties/Methods for CIM_ElementSettingData (Associates Port and StorageClientSettingData)

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	
SettingData		Mandatory	

28.7.21 CIM_ElementSettingData (Associates ProtocolController and StorageClientSettingData)

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

Table 601 describes class CIM_ElementSettingData (Associates ProtocolController and StorageClientSettingData).

Table 601 - SMI Referenced Properties/Methods for CIM_ElementSettingData (Associates ProtocolController and StorageClientSettingData)

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	
SettingData		Mandatory	

28.7.22 CIM_ElementSettingData (Associates StorageHardwareID and StorageClientSettingData)

Created By: Extrinsic: CIM_StorageHardwareIDManagementService.CreateStorageHardwareID

Modified By: Static

Deleted By: Extrinsic: CIM_StorageHardwareIDManagementService.DeleteStorageHardwareID

Requirement: Optional

Table 602 describes class CIM_ElementSettingData (Associates StorageHardwareID and StorageClientSettingData).

Table 602 - SMI Referenced Properties/Methods for CIM_ElementSettingData (Associates StorageHardwareID and StorageClientSettingData)

Properties	Flags	Requirement	Description & Notes
ManagedElement		Mandatory	
SettingData		Mandatory	

28.7.23 CIM_EnabledLogicalElementCapabilities

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 603 describes class CIM_EnabledLogicalElementCapabilities.

Table 603 - SMI Referenced Properties/Methods for CIM_EnabledLogicalElementCapabilities

Properties	Flags	Requirement	Description & 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.

28.7.24 CIM_GroupMaskingMappingCapabilities

A set of properties that describe the capabilities of a group masking and mapping provider. The class definition specializes the CIM_ProtocolControllerMaskingCapabilities definition in the Masking and Mapping 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 604 describes class CIM_GroupMaskingMappingCapabilities.

Table 604 - SMI Referenced Properties/Methods for CIM_GroupMaskingMappingCapabilities

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	Opaque and unique identifer.
ElementName		Mandatory	User-friendly name.
ValidHardwareIdTypes		Mandatory	A list of the valid values for StrorageHardwareID.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 ProtocolContollers attached to a SCSIProtocolEndpoint can expose the same unit numbers (e.g. multiple LUN 0s) or if the numbers must 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 it the instrumentation supports default view SPCs that exposes logical units to all initiators.

Table 604 - SMI Referenced Properties/Methods for CIM_GroupMaskingMappingCapabilities

Properties	Flags	Requirement	Description & Notes
SupportedFeatures (added)		Mandatory	Enumeration indicating the capabilities of masking and mapping features having to do with masking groups. Values: 2: Supports initiator masking group 3: Supports target masking group 4: Supports device masking group 5: Auto assigns host device numbers 6: Maskview creation requires initiator masking group 7: Maskview creation requires target masking group 8: Maskview creation requires device masking group 9: Maskview requires non-empty initiator masking group 10: Maskview requires non-empty target masking group 11: Maskview requires non-empty device masking group.
SupportedAsynchronousActions (added)		Mandatory	Identify group masking methods using job control. Values: 19: CreateGroup 20: DeleteGroup 21: AddMembers 22: RemoveMembers 23: CreateMaskingView 24: DeleteMaskingView 25: ModifyMaskingView.
SupportedSynchronousActions (added)		Mandatory	Identify group masking methods not using job control. Values: 19: CreateGroup 20: DeleteGroup 21: AddMembers 22: RemoveMembers 23: CreateMaskingView 24: DeleteMaskingView 25: ModifyMaskingView.

Table 604 - SMI Referenced Properties/Methods for CIM_GroupMaskingMappingCapabilities

Properties	Flags	Requirement	Description & Notes
SupportedDeviceGroupFeatures (added)		Conditional	Conditional requirement: Required if device masking groups are supported. Enumeration indicating the capabilities of Initiator groups. Values: 2: Group is nameable 3: Can add to an associated group 4: Empty group is allowed 5: Group associated with view can be empty 6: Nested groups allowed 7: Only one level of nested groups 8: Group can participate in multiple views 9: Maskview deletion deletes unassociated masking group 10: Associated empty group indicates no access 11: Unassociated group rejects device numbers.

Table 604 - SMI Referenced Properties/Methods for CIM_GroupMaskingMappingCapabilities

Properties	Flags	Requirement	Description & Notes
SupportedInitiatorGroupFeatures (added)		Conditional	Conditional requirement: Required if initiator masking groups are supported. Enumeration indicating the capabilities of Initiator groups. Values: 2: Group is nameable 3: Can add to an associated group 4: Empty group is allowed 5: Group associated with view can be empty 6: Nested groups allowed 7: Only one level of nested groups 8: Group can participate in multiple views 9: Maskview deletion deletes unassociated masking group 10: Associated empty group indicates no access 11: ConsistentLogicalUnitNumber must be true.
SupportedTargetGroupFeatures (added)		Conditional	Conditional requirement: Required if target masking groups are supported. Enumeration indicating the capabilities of Initiator groups. Values: 2: Group is nameable 3: Can add to an associated group 4: Empty group is allowed 5: Group associated with view can be empty 6: Nested groups allowed 7: Only one level of nested groups 8: Group can participate in multiple views 9: Maskview deletion deletes unassociated masking group 10: Associated empty group indicates no access.

28.7.25 CIM_GroupMaskingMappingService

Central class for Group Masking and Mapping Profile. The class definition specializes the CIM_ControllerConfigurationService definition in the Masking and Mapping 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 605 describes class CIM_GroupMaskingMappingService.

Table 605 - SMI Referenced Properties/Methods for CIM_GroupMaskingMappingService

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	The scoping System CreationClassName.
SystemName		Mandatory	The scoping System Name.
CreationClassName		Mandatory	The name of the concrete subclass.
Name		Mandatory	Unique identifer for the Service.
ExposePaths()		Mandatory	
HidePaths()		Mandatory	
ExposeDefaultLUs()		Optional	
HideDefaultLUs()		Optional	
DeleteProtocolController()		Optional	
CreateMaskingView() (added)		Mandatory	
DeleteMaskingView() (added)		Optional	
ModifyMaskingView() (added)		Optional	
CreateGroup() (added)		Mandatory	
DeleteGroup() (added)		Optional	
AddMembers() (added)		Mandatory	
RemoveMembers() (added)		Mandatory	

28.7.26 CIM_HostedCollection

Created By: Extrinsic: CIM_StorageHardwareIDManagementService.CreateHardwareIDCollection
Modified By: Static
Deleted By: Static
Requirement: Implementation support for collections of StorageHardwareIDs.

Table 606 describes class CIM_HostedCollection.

Table 606 - SMI Referenced Properties/Methods for CIM_HostedCollection

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

28.7.27 CIM_HostedService (Associates ComputerSystem and ControllerConfigurationService)

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

Table 607 describes class CIM_HostedService (Associates ComputerSystem and ControllerConfigurationService).

Table 607 - SMI Referenced Properties/Methods for CIM_HostedService (Associates Computer-System and ControllerConfigurationService)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

28.7.28 CIM_HostedService (Associates ComputerSystem and PrivilegeManagementService)

Deprecated.

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

Table 608 describes class CIM_HostedService (Associates ComputerSystem and PrivilegeManagementService).

Table 608 - SMI Referenced Properties/Methods for CIM_HostedService (Associates Computer-System and PrivilegeManagementService)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

28.7.29 CIM_HostedService (Associates ComputerSystem and StorageHardwareIDManagementService)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 609 describes class CIM_HostedService (Associates ComputerSystem and StorageHardwareIDManagementService).

Table 609 - SMI Referenced Properties/Methods for CIM_HostedService (Associates ComputerSystem and StorageHardwareIDManagementService)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

28.7.30 CIM_InitiatorMaskingGroup

Represents a group of initiator ports (StorageHardwareIDs).

Created By: Extrinsic: CreateGroup

Modified By: Extrinsics: AddMembers, RemoveMembers

Deleted By: Extrinsic: DeleteGroup

Requirement: Mandatory

Table 610 describes class CIM_InitiatorMaskingGroup.

Table 610 - SMI Referenced Properties/Methods for CIM_InitiatorMaskingGroup

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	Within the scope of an array, the InstanceID opaquely and uniquely identifies an instance of this class.
DeleteOnEmpty	M	Mandatory	If true and empty groups are allowed, the group will be deleted when the last element is removed from the group. If empty groups are not allowed, the group will be deleted automatically when the group becomes empty.
DeleteWhenBecomesUnassociated	M	Mandatory	If true, the group will be deleted when the group is no longer associated with a masking view. This can happen if all masking views associated to this group are deleted.
ConsistentLogicalUnitNumber	M	Mandatory	If true, it indicates the device numbers for a volume visible to the same initiator though different paths must be the same.
ElementName		Optional	User Friendly name.

28.7.31 CIM_MemberOfCollection

Created By: Extrinsics: CIM_StorageHardwareIDManagementService.CreateHardwareIDCollection, CIM_StorageHardwareIDManagementService.AddHardwareIDsToCollection

Modified By: Static

Deleted By: Static

Requirement: Implementation support for collections of StorageHardwareIDs.

Table 611 describes class CIM_MemberOfCollection.

Table 611 - SMI Referenced Properties/Methods for CIM_MemberOfCollection

Properties	Flags	Requirement	Description & Notes
Collection		Mandatory	
Member		Mandatory	

28.7.32 CIM_PrivilegeManagementService

Deprecated.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 612 describes class CIM_PrivilegeManagementService.

Table 612 - SMI Referenced Properties/Methods for CIM_PrivilegeManagementService

Properties	Flags	Requirement	Description & 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	

28.7.33 CIM_ProtocolController

Created By: Extrinsics: CIM_ControllerConfigurationService.ExposePaths, CIM_ControllerConfigurationService.HidePaths

Modified By: Extrinsics: CIM_ControllerConfigurationService.ExposePaths, CIM_ControllerConfigurationService.HidePaths

Deleted By: Extrinsics: CIM_ControllerConfigurationService.ExposePaths, CIM_ControllerConfigurationService.HidePaths

Requirement: Mandatory

Table 613 describes class CIM_ProtocolController.

Table 613 - SMI Referenced Properties/Methods for CIM_ProtocolController

Properties	Flags	Requirement	Description & 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.

28.7.34 CIM_ProtocolControllerForUnit

Created By: Extrinsics: CIM_ControllerConfigurationService.ExposePaths, CIM_ControllerConfigurationService.HidePaths, CIM_ControllerConfigurationService.ExposeDefaultLUs, CIM_ControllerConfigurationService.HideDefaultLUs

Modified By: Extrinsics: CIM_ControllerConfigurationService.ExposePaths, CIM_ControllerConfigurationService.HidePaths, CIM_ControllerConfigurationService.ExposeDefaultLUs, CIM_ControllerConfigurationService.HideDefaultLUs

Deleted By: Extrinsics: CIM_ControllerConfigurationService.ExposePaths, CIM_ControllerConfigurationService.HidePaths, CIM_ControllerConfigurationService.ExposeDefaultLUs, CIM_ControllerConfigurationService.HideDefaultLUs

Requirement: Mandatory

Table 614 describes class CIM_ProtocolControllerForUnit.

Table 614 - SMI Referenced Properties/Methods for CIM_ProtocolControllerForUnit

Properties	Flags	Requirement	Description & Notes
DeviceNumber		Mandatory	Address (e.g. LUN) of the associated Device. Shall be formatted as unseparated uppercase hexadecimal digits, with no leading 0x.
DeviceAccess		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).

28.7.35 CIM_SAPAvailableForElement

Created By: Extrinsics: CIM_ControllerConfigurationService.ExposePaths, CIM_ControllerConfigurationService.HidePaths, CIM_ControllerConfigurationService.ExposeDefaultLUs, CIM_ControllerConfigurationService.HideDefaultLUs

Modified By: Extrinsics: CIM_ControllerConfigurationService.ExposePaths, CIM_ControllerConfigurationService.HidePaths, CIM_ControllerConfigurationService.ExposeDefaultLUs, CIM_ControllerConfigurationService.HideDefaultLUs

Deleted By: Extrinsics: CIM_ControllerConfigurationService.ExposePaths, CIM_ControllerConfigurationService.HidePaths, CIM_ControllerConfigurationService.ExposeDefaultLUs, CIM_ControllerConfigurationService.HideDefaultLUs

Requirement: Mandatory

Table 615 describes class CIM_SAPAvailableForElement.

Table 615 - SMI Referenced Properties/Methods for CIM_SAPAvailableForElement

Properties	Flags	Requirement	Description & Notes
AvailableSAP		Mandatory	
ManagedElement		Mandatory	

28.7.36 CIM_ServiceAffectsElement (Between GroupMaskingMappingService and MaskingGroup)

Associates Group Masking Mapping Service to Masking Group.

Created By: Extrinsic: CreateGroup

Modified By: Extrinsics: DeleteGroup, RemoveMembers

Deleted By: Extrinsic: DeleteGroup

Requirement: Required if device masking groups are supported or Required if initiator masking groups are supported or Required if target masking groups are supported.

Table 616 describes class CIM_ServiceAffectsElement (Between GroupMaskingMappingService and MaskingGroup).

Table 616 - SMI Referenced Properties/Methods for CIM_ServiceAffectsElement (Between Group-MaskingMappingService and MaskingGroup)

Properties	Flags	Requirement	Description & Notes
AffectingElement		Mandatory	Group Masking Mapping Service.
AffectedElement		Mandatory	Masking Group.

28.7.37 CIM_StorageClientSettingData

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 617 describes class CIM_StorageClientSettingData.

Table 617 - SMI Referenced Properties/Methods for CIM_StorageClientSettingData

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	Opaque and unique identifier.
ElementName		Mandatory	A user-friendly name.
ClientTypes		Mandatory	Array of OS names.

28.7.38 CIM_StorageHardwareID

Created By: Extrinsics: CIM_StorageHardwareIDManagementService.CreateStorageHardwareID, CIM_ControllerConfigurationService.ExposePaths

Modified By: Static

Deleted By: Extrinsic: CIM_StorageHardwareIDManagementService.DeleteStorageHardwareID

Requirement: Mandatory

Table 618 describes class CIM_StorageHardwareID.

Table 618 - SMI Referenced Properties/Methods for CIM_StorageHardwareID

Properties	Flags	Requirement	Description & 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).

28.7.39 CIM_StorageHardwareIDManagementService

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 619 describes class CIM_StorageHardwareIDManagementService.

Table 619 - SMI Referenced Properties/Methods for CIM_StorageHardwareIDManagementService

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	The scoping System CreationClassName.
SystemName		Mandatory	The scoping System Name.
CreationClassName		Mandatory	The name of the concrete subclass.

Table 619 - SMI Referenced Properties/Methods for CIM_StorageHardwareIDManagementService

Properties	Flags	Requirement	Description & Notes
Name		Mandatory	Uniquely identifies the Service.
CreateStorageHardwareID()		Mandatory	
DeleteStorageHardwareID()		Mandatory	
CreateHardwareIDCollection()		Optional	
AddHardwareIDsToCollection()		Optional	

28.7.40 CIM_SystemSpecificCollection

Created By: Extrinsic: CIM_StorageHardwareIDManagementService.CreateHardwareIDCollection
Modified By: Static
Deleted By: Static
Requirement: Implementation support for collections of StorageHardwareIDs.

Table 620 describes class CIM_SystemSpecificCollection.

Table 620 - SMI Referenced Properties/Methods for CIM_SystemSpecificCollection

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	Opaque and unique identifier.
ElementName		Mandatory	A user-friendly name.

28.7.41 CIM_TargetMaskingGroup

Represents a group of target ports (ProtocolEndpoints).

Created By: Extrinsic: CreateGroup
Modified By: Extrinsics: AddMembers, RemoveMembers
Deleted By: Extrinsic: DeleteGroup
Requirement: Mandatory

Table 621 describes class CIM_TargetMaskingGroup.

Table 621 - SMI Referenced Properties/Methods for CIM_TargetMaskingGroup

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	Within the scope of an array, the InstanceID opaquely and uniquely identifies an instance of this class.
DeleteOnEmpty	M	Mandatory	If true and empty groups are allowed, the group will be deleted when the last element is removed from the group. If empty groups are not allowed, the group will be deleted automatically when the group becomes empty.
DeleteWhenBecomesUnassociated	M	Mandatory	If true, the group will be deleted when the group is no longer associated with a masking view. This can happen if all masking views associated to this group are deleted.
ElementName		Optional	User Friendly name.

28.7.42 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 622 describes class SNIA_ProtocolControllerMaskingCapabilities.

Table 622 - SMI Referenced Properties/Methods for SNIA_ProtocolControllerMaskingCapabilities

Properties	Flags	Requirement	Description & 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.

28.7.43 SNIA_StorageHardwareID

Created By: Extrinsics: CIM_StorageHardwareIDManagementService.CreateStorageHardwareID, CIM_ControllerConfigurationService.ExposePaths

Modified By: Static

Deleted By: Extrinsic: CIM_StorageHardwareIDManagementService.DeleteStorageHardwareID

Requirement: Optional

Table 623 describes class SNIA_StorageHardwareID.

Table 623 - SMI Referenced Properties/Methods for SNIA_StorageHardwareID

Properties	Flags	Requirement	Description & 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).

28.7.44 SNIA_StorageHardwareIDManagementService

Experimental subclass with support for SAS StorageHardwareIDs.

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

Table 624 describes class SNIA_StorageHardwareIDManagementService.

Table 624 - SMI Referenced Properties/Methods for SNIA_StorageHardwareIDManagementService

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	The scoping System CreationClassName.
SystemName		Mandatory	The scoping System Name.
CreationClassName		Mandatory	The name of the concrete subclass.
Name		Mandatory	Uniquely identifies the Service.
CreateStorageHardwareID()		Mandatory	Experimental: may use SAS Address IDType.
DeleteStorageHardwareID()		Mandatory	
CreateHardwareIDCollection()		Optional	
AddHardwareIDsToCollection()		Optional	

EXPERIMENTAL

EXPERIMENTAL

Clause 29: Storage Relocation Profile

29.1 Overview

Typically, a storage pool is established on a set of storage extents, and a volume is allocated to a storage pool with different performance characteristics (e.g. with different RPM drives, or different drive classes).

However, Storage Relocation can be used to relocate storage (e.g. storage volume, storage pool, logical disk) within the same storage machine or cross different storage machines with different extent allocation methods. Storage Relocation can be used as a solution on data migration; and can also be used as part of a solution on hot spot tuning, while there may be additional backend execution on hot spot tuning which is transparent to customers. The relocation is performed concurrently with IO operations and the host views of the volumes do not change. So, the relocations are transparent to host operations.

This profile defines three types of storage relocation operations defined in this profile: Storage Volume Relocation, Logical Disk Relocation, Storage Pool Relocation.

29.1.1 Storage Volume Relocation

Volume relocation refers to the operation to migrate every extent of a volume to a different set of extents. In this profile, volume relocation is simply done by: 1) relocating volume away from one storage pool and into another; or 2) relocating onto a new group of extents within one storage pool.

The extent migration serves as a basic utility for the volume relocation function. It migrates data from a source extent to another target extent. The source and target extent can be any two distinct extents as long as the migrations can be done through storage controllers (within same storage machine, or cross different storage machines).

29.1.2 Logical Disk Relocation

Similar to storage volume relocation, logical disk relocation refers to the operation to migrate every extent of a logical disk to a different set of extents. In this profile, logical disk relocation can be done by: relocating logical disk onto a new group of extents.

29.1.3 Storage Pool Relocation

Storage pools are container objects which allows user to group extents together. Typically, users consider performance characteristics and/or failure boundaries on grouping extents. Volumes created from the pools inherit the characteristics of the storage in the pools. In this profile, storage pool relocation includes: 1) relocating storage pool onto a new group of extents; 2) merging storage pools.

Relocating storage pool onto a new group of extents needs extent migration which is similar to volume relocation.

Merging storage pools is an operation which permits user to merge multiple storage pools into one, which finally enables user to create volumes on more extents. Merging storage pools is also helpful for users to define storage tiers by grouping extents with similar performance characteristics into one storage pool. Typically pools merge might not involve migrating extents. It might just involve a series of global metadata updates in storage machine. So the time duration for this operation can be relatively short.

29.2 Model

New class ‘StorageRelocationService’ is defined in this profile; and it extends the classes of Block Services, and Job Control.

29.3 Implementation

Figure 150. presents the classes related to this profile:

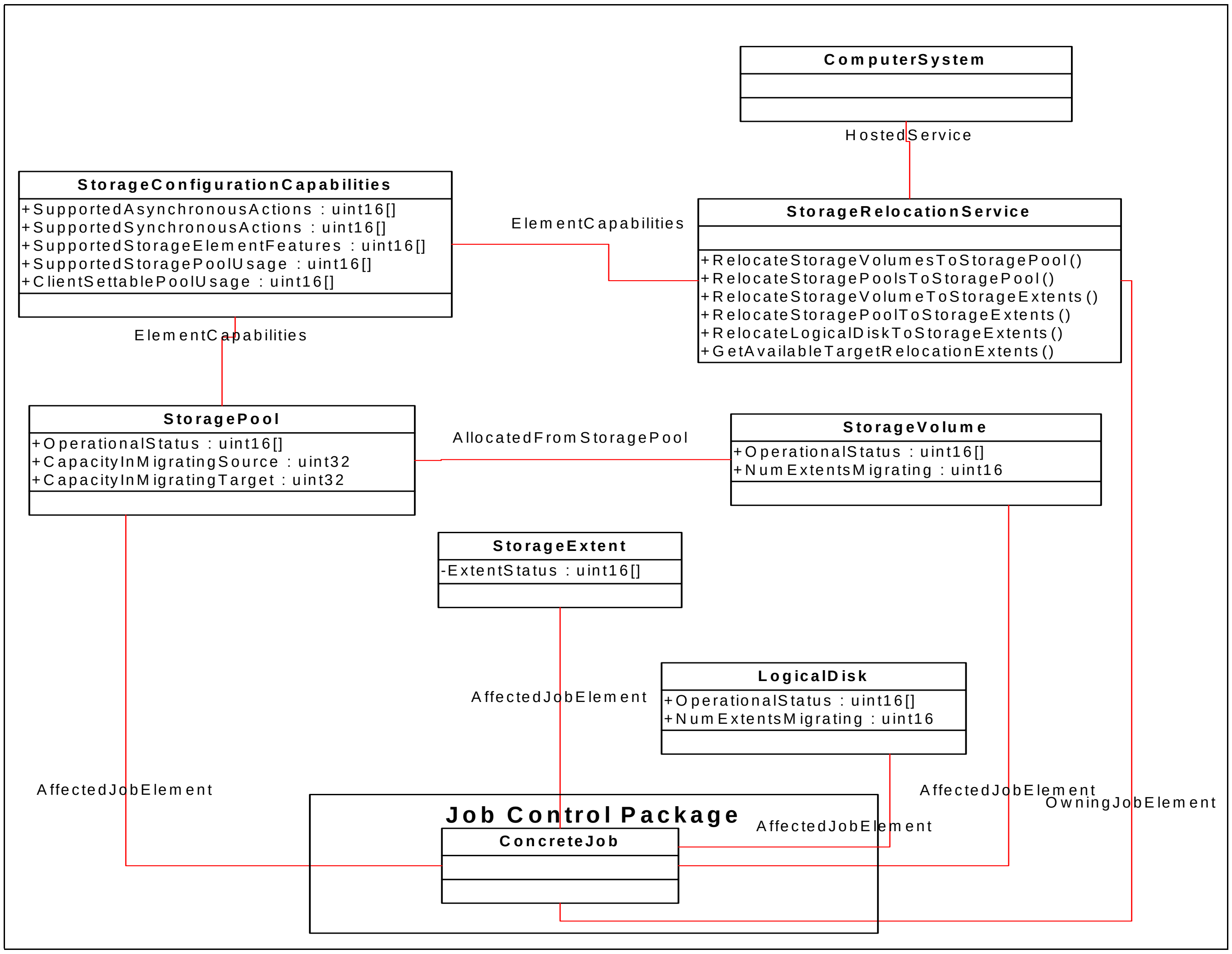
ComputerSystem
HostedService
StorageConfigurationCapabilities
+SupportedAsynchronousActions : uint16[]
+SupportedSynchronousActions : uint16[]
+SupportedStorageElementFeatures : uint16[]
+SupportedStoragePoolUsage : uint16[]
+ClientSettablePoolUsage : uint16[]
ElementCapabilities
StorageRelocationService
+RelocateStorageVolumesToStoragePool()
+RelocateStoragePoolsToStoragePool()
+RelocateStorageVolumeToStorageExtents()
+RelocateStoragePoolToStorageExtents()
+RelocateLogicalDiskToStorageExtents()
+GetAvailableTargetRelocationExtents()
ElementCapabilities
StoragePool
+OperationalStatus : uint16[]
+CapacityInMigratingSource : uint32
+CapacityInMigratingTarget : uint32
AllocatedFromStoragePool
StorageVolume
+OperationalStatus : uint16[]
+NumExtentsMigrating : uint16
StorageExtent
-ExtentStatus : uint16[]
LogicalDisk
+OperationalStatus : uint16[]
+NumExtentsMigrating : uint16
AffectedJobElement
AffectedJobElement
AffectedJobElement
AffectedJobElement
OwningJobElement
Job Control Package
ConcreteJob

Figure 150 - Storage Relocation

StorageConfigurationCapabilities.SupportedStorageElementFeatures shall include a subset of 'StorageVolume Relocation', 'LogicalDisk Relocation', and 'StoragePool Relocation' to indicate support for relocation of StorageVolumes, LogicalDisks, or StoragePools.

StorageConfigurationCapabilities.SupportedAsynchronousActions shall include a subset of 'StorageVolume Relocation', 'LogicalDisk Relocation', and 'StoragePool Relocation' to indicate support for asynchronous relocation control of StorageVolumes, LogicalDisks, or StoragePools.

StorageConfigurationCapabilities.SupportedSynchronousActions shall include a subset of 'StorageVolume Relocation', 'LogicalDisk Relocation', and 'StoragePool Relocation' to indicate support for asynchronous relocation control of StorageVolumes, LogicalDisks, or StoragePools.

StorageConfigurationCapabilities.SupportedStoragePoolUsage shall include a subset of 'Used as source for Relocation Service', and 'Used as target for Relocation Service' to indicate the support for each StoragePool (Primordial or Concrete) in relocation.

StorageConfigurationCapabilities.ClientSettablePoolUsage shall include a subset of 'Used as source for Relocation Service', and 'Used as target for Relocation Service' to indicate the support for each StoragePool (Primordial or Concrete) in relocation.

The 'CapacityInMigratingSource' of StoragePool defines the capacity in bytes of allocated extents in the process of being migrated out of this storage pool when volume relocation is on going. The 'CapacityInMigratingSource' property is optional if the Storage Relocation profile is not supported.

The 'CapacityInMigratingTarget' of StoragePool defines the capacity in bytes of allocated Extents in the process of being migrated into this storage pool when volume relocation is on going. The 'CapacityInMigratingTarget' property is optional if the Storage Relocation profile is not supported.

The 'Relocating' value for 'OperationalStatus' of both source and target StoragePools defines the status when the relocation operation of source and target storage pools is on going.

The 'NumExtentsMigrating' of StorageVolume defines the number of Extents in the process of migrating for this storage volume when the volume relocation is on going. The 'NumExtentsMigrating' property is optional if the Storage Relocation profile is not supported.

The 'Relocating' value for 'OperationalStatus' of StorageVolume defines the status when the volume relocation is on going.

The 'NumExtentsMigrating' of LogicalDisk defines the number of Extents in the process of migrating for this logical disk when the relocation is on going. The 'NumExtentsMigrating' property is optional if the Storage Relocation profile is not supported.

The 'Relocating' value for 'OperationalStatus' of LogicalDisk defines the status when the volume relocation is on going.

The 'Relocating' value for 'ExtentStatus' defines the status of StorageExtent who belongs to relocation ongoing progress.

StorageRelocationService is defined as the service class for storage relocation methods. Two methods of StorageRelocationService will be used for relocation:

- RelocateStorageVolumesToStoragePool: this method is defined to relocate a group of storage volumes into a target storage pool.
- RelocateStoragePoolsToStoragePool: this method is defined to relocate a group of storage pools into a target storage pool.
- RelocateStorageVolumeToStorageExtents: this method is defined to relocate a storage volume onto a new group of storage extents.
- RelocateStoragePoolToStorageExtents: this method is defined to relocate a storage pool onto a new group of storage extents.
- RelocateLogicalDiskToStorageExtents: this method is defined to relocate a logical disk onto a new group of storage extents.
- GetAvailableTargetRelocationExtents: this method is defined to get available storage extents for relocation.

29.3.1 Capacity of StoragePool after StorageVolume relocation

After relocating StorageVolume away from source StoragePool and onto target StoragePool, the available capacity (RemainingManagedSpace) of source StoragePool should be increased by the capacity

(AllocatedFromStoragePool.SpaceConsumed) of that StorageVolume, while the available capacity (RemainingManagedSpace) of target StoragePool should be decreased by the capacity (AllocatedFromStoragePool.SpaceConsumed) of that StorageVolume. The total capacity of both source StoragePool and target StoragePool won't be changed.

29.3.2 Capacity of StoragePool after StoragePool merge

After merging source StoragePool into target StoragePool, the source StoragePool will never exist, meanwhile the total capacity and remaining capacity of target StoragePool should be increased by that capacity of the source one.

29.3.3 Capacity of StorageElement after be relocated onto new group of StorageExtents

After relocating StorageElement (StorageVolume, StoragePool, LogicalDisk) onto new group of StorageExtents, the capacity will be as same as before.

29.3.4 Track and control of Relocation progress

As one implementation option, Job Control Subprofile can be used to track and control relocation progress. When the Job Control Subprofile is implemented and a client executes the relocation that executes asynchronously, a reference to an instance of ConcreteJob is returned and the return value for the method is set to "Method parameters checked - job started".

The ConcreteJob instance allows the progress of the method to be checked, suspended, resumed, and terminated, etc. And Indications can be used to subscribe for Job completion. For more details, see *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* Clause 26: Job Control Subprofile.

29.3.5 Relocation Capabilities of Storage Pool

StoragePool has a key role in storage relocation. So defining relocation capabilities of each StoragePool (Primordial and Concrete) is recommended:

- If a primordial or concrete StoragePool can be used as an source in relocation, StorageConfigurationCapabilities.SupportedStoragePoolUsage should include 'Used as source for Relocation Service'.
- If a primordial or concrete StoragePool can be used as a target in relocation, StorageConfigurationCapabilities.SupportedStoragePoolUsage should include 'Used as target for Relocation Service'.
- If a primordial or concrete StoragePool can be used as either an source or a target, StorageConfigurationCapabilities.SupportedStoragePoolUsage should include both.
- If the storage system supports client to configure the capabilities of the storage pool, StorageConfigurationCapabilities.ClientSettablePoolUsage should be configured to include 'Used as source for Relocation Service', or 'Used as target for Relocation Service', or both.

29.4 Indications

The implementation of Storage Relocation can be asynchronous operation, so indications can be used to notify multiple clients.

29.4.1 StorageVolume Relocation starts (msgID: DRM32)

This is an alert message indicating that the relocation of a StorageVolume starts. The related standard message can be:

```
Relocation starts for StorageVolume with identifier DeviceID.
```

29.4.2 StorageVolume Relocation ends (msgID: DRM33)

This is an alert message indicating that the relocation of a StorageVolume ends. The related standard message can be:

```
Relocation ends for StorageVolume with identifier DeviceID.
```

29.4.3 StoragePool Relocation starts (msgID: DRM34)

This is an alert message indicating that the relocation of a StoragePool starts. The related standard message can be:

```
Relocation starts for StoragePool with identifier PoolID.
```

29.4.4 StoragePool Relocation ends (msgID: DRM35)

This is an alert message indicating that the relocation of a StoragePool ends. The related standard message can be:

```
Relocation ends for StoragePool with identifier PoolID.
```

29.4.5 LogicalDisk Relocation starts (msgID: DRM36)

This is an alert message indicating that the relocation of a LogicalDisk starts. The related standard message can be:

```
Relocation starts for LogicalDisk with identifier DeviceID.
```

29.4.6 LogicalDisk Relocation ends (msgID: DRM37)

This is an alert message indicating that the relocation of a LogicalDisk ends. The related standard message can be:

```
Relocation ends for LogicalDisk with identifier DeviceID.
```

29.5 Health and Fault Management Consideration

Not defined in this standard.

29.6 Cascading Considerations

Not defined in this standard.

29.7 Mapping & Masking Considerations

For both local and remote relocation, after it’s done, the source should disappear, and the target should be the new replacement.

In local relocation, it should be transparent to client sides, which means every properties of the target volume should be as same as source volume and mapping relationship to host should not be impacted. But it’s not constrained here to have something changed on target, it depends on the real behavior of storage system.

In remote relocation, the source object will disappear from source device, and target object will appear on target device. So the remote relocation may be only permitted when there is relation from the volume to others (e.g. mapping relation to host). But if the storage system has the ability to migrate all the relations together with the relocation action, it can still be achieved.

So to mapping & masking, there should be no impact in local relocation, but it can depend on the behavior of the storage system. And to remote relocation, it depends on the behavior of the storage system.

29.8 Supported Profiles, Subprofiles, and Packages

Table 625 describes the supported profiles for Storage Relocation.

Table 625 - Supported Profiles for Storage Relocation

Profile Name	Organization	Version	Requirement	Description
Block Services	SNIA	1.6.0	Optional	
Job Control	SNIA	1.5.0	Optional	

29.9 Methods of the Profile

This profile defines StorageRelocationService as the service class for storage relocation methods. 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.

29.9.1 RelocateStorageVolumesToStoragePool

```
uint32 RelocateStorageVolumesToStoragePool(
        [IN,OUT]  CIM_StorageVolume REF TheElements[],
        [OUT]     CIM_ConcreteJob REF Job,
        [IN]      CIM_StorageSetting REF TargetSettingGoal,
        [IN,OUT]  CIM_StoragePool REF TargetPool);
```

This method is defined to relocate a group of storage volumes into a target storage pool, by inputting ‘TheElements’ as storage volumes to be relocated, ‘TargetSettingGoal’ as setting goal, and ‘TargetPool’ as target storage pool. The output 'Job' parameter can be used to track the relocation progress. The detailed description for each parameter is as follows:

- TheElements: As an input, TheElements is an array of storage volumes for the source of relocation. As an output, it represents the storage volumes after relocation.
- Job: Reference to the job (may be null if job completed).
- TargetSettingGoal: The requirements for the relocation target pool. If set to a null value, the default configuration associated with the service will be used. This parameter should be a reference to a CIM_StorageSetting which represent the profile appropriate to the relocation target pool. If not NULL, this parameter will supply a new Goal for the target pool.

- TargetPool: A reference to target storage pool instance used for relocation. As an input parameter, TargetPool specifies the storage pool to relocate source onto. As an output parameter, TargetPool represents the pool actually used as the relocation target. It is output only when the relocation succeeds.

29.9.2 RelocateStoragePoolsToStoragePool

```
uint32 RelocateStoragePoolsToStoragePool(
        [IN,OUT]  CIM_StoragePool REF TheElements[],
        [OUT]     CIM_ConcreteJob REF Job,
        [IN]      CIM_StorageSetting REF TargetSettingGoal,
        [IN,OUT]  CIM_StoragePool REF TargetPool);
```

This method is defined to relocate a group of storage pools into a target storage pool, by inputting ‘TheElements’ as source storage pools to be relocated, ‘TargetSettingGoal’ as setting goal, and ‘TargetPool’ as the target storage pool. The output 'Job' parameter can be used to track the relocation progress. The detailed description for each parameter is as follows:

- TheElements: As an input, TheElements is an array of storage pools for the source of relocation. As an output, it represents the storage pools after relocation.
- Job: Reference to the job (may be null if job completed).
- TargetSettingGoal: The requirements for the relocation target pool. If set to a null value, the default configuration associated with the service will be used. This parameter should be a reference to a CIM_StorageSetting which represent the profile appropriate to the relocation target pool. If not NULL, this parameter will supply a new Goal for the target pool.
- TargetPool: A reference to target storage pool instance used for relocation. As an input parameter, TargetPool specifies the storage pool to relocate source onto. As an output parameter, TargetPool represents the pool actually used as the relocation target. It is output only when the relocation succeeds.

29.9.3 RelocateStorageVolumeToStorageExtents

```
uint32 RelocateStorageVolumeToStorageExtents(
        [IN]      CIM_StorageExtent REF InElements[],
        [OUT]     CIM_ConcreteJob REF Job,
        [IN]      CIM_StorageSetting REF TargetSettingGoal,
        [IN,OUT]  CIM_StorageVolume REF TheElement);
```

This method is defined to relocate a storage volume onto a new group of storage extents, by inputting ‘InElements’ as the group of storage extents to move onto, ‘TargetSettingGoal’ as setting goal, and ‘TheElement’ as target storage volume to be relocated. The output 'Job' parameter can be used to track the relocation progress. The detailed description for each parameter is as follows:

- InElements: An array of StorageExtents that 'TheElement' is relocated to.
- Job: Reference to the job (may be null if job completed).
- TargetSettingGoal: The requirements for the relocation target element. If set to a null value, the default configuration associated with the service will be used. This parameter should be a reference to a CIM_StorageSetting which represents the profile appropriate to the relocation target elements. If not NULL, this parameter will supply a new Goal for the target elements.
- TheElement: As an input, TheElement is a storageVolume as the source of relocation. As an output, it represents the storageVolume after relocation.

29.9.4 RelocateStoragePoolToStorageExtents

```
uint32 RelocateStoragePoolToStorageExtents(
        [IN]      CIM_StorageExtent REF InElements[],
        [OUT]     CIM_ConcreteJob REF Job,
        [IN]      CIM_StorageSetting REF TargetSettingGoal,
        [IN,OUT]  CIM_StoragePool REF TheElement);
```

This method is defined to relocate a storage pool onto a new group of storage extents, by inputting ‘InElements’ as the group of storage extents to move onto, ‘TargetSettingGoal’ as setting goal, and ‘TheElement’ as the target storage pool to be relocated. The output 'Job' parameter can be used to track the relocation progress. The detailed description for each parameter is as follows:

- InElements: An array of StorageExtents that 'TheElement' is relocated to.
- Job: Reference to the job (may be null if job completed).
- TargetSettingGoal: The requirements for the relocation target element. If set to a null value, the default configuration associated with the service will be used. This parameter should be a reference to a CIM_StorageSetting which represents the profile appropriate to the relocation target elements. If not NULL, this parameter will supply a new Goal for the target elements.
- TheElement: As an input, TheElement is a storagePool as the source of relocation. As an output, it represents the storagePool after relocation.

29.9.5 RelocateLogicalDiskToStorageExtents

```
uint32 RelocateLogicalDiskToStorageExtents(
        [IN]      CIM_StorageExtent REF InElements[],
        [OUT]     CIM_ConcreteJob REF Job,
        [IN]      CIM_StorageSetting REF TargetSettingGoal,
        [IN,OUT]  CIM_LogicalDisk REF TheElement);
```

This method is defined to relocate a logical disk onto a new group of storage extents, by inputting ‘InElements’ as the group of storage extents to move onto, ‘TargetSettingGoal’ as setting goal, and ‘TheElement’ as target logical disk to be relocated. The output 'Job' parameter can be used to track the relocation progress. The detailed description for each parameter is as follows:

- InElements: An array of StorageExtents that 'TheElement' is relocated to.
- Job: Reference to the job (may be null if job completed).
- TargetSettingGoal: The requirements for the relocation target element. If set to a null value, the default configuration associated with the service will be used. This parameter should be a reference to a CIM_StorageSetting which represents the profile appropriate to the relocation target elements. If not NULL, this parameter will supply a new Goal for the target elements.
- TheElement: As an input, TheElement is a logicalDisk as the source of relocation. As an output, it represents the logicalDisk after relocation.

29.9.6 GetAvailableTargetRelocationExtents

```
uint32 GetAvailableTargetRelocationExtents(
        [IN]      CIM_LogicalElement REF TheElement,
        [IN]      CIM_StorageSetting REF TargetSettingGoal,
        [IN]      CIM_StoragePool REF InPool,
                    [OUT]     CIM_StorageExtent REF AvailableExtents[]);
```

This method is defined to get available group of storage extents for relocation. It supports:

a) Get available target storage extents for storage volume relocation: by inputting ‘TheElement’ as target storage volume to be relocated, ‘TargetSettingGoal’ as setting goal, ‘InPool’ as the source for new group of storage extents. The output ‘AvailableExtents’ parameter will return the candidate extents for relocation.

b) Get available target storage extents for storage pool relocation: by inputting ‘TheElement’ as target storage pool to be relocated, ‘TargetSettingGoal’ as setting goal, ‘InPool’ as the source for new group of storage extents. The output ‘AvailableExtents’ parameter will return the candidate extents for relocation.

c) Get available target storage extents for logical disk relocation: by inputting ‘TheElement’ as target logical disk to be relocated, ‘TargetSettingGoal’ as setting goal, ‘InPool’ as the source for new group of storage extents. The output ‘AvailableExtents’ parameter will return the candidate extents for relocation.

The detailed description for each parameter is as follows:

- TheElement: TheElement is a storage element as the source of relocation.
- TargetSettingGoal: The TargetSettingGoal for which supported extents should be retrieved as available for relocation. If a NULL is passed for the Goal, the method will return all available extents, regardless of the goal. There exists a possibility of error in relocating a Pool, Volume, or LogicalDisk to extents retrieved in this manner.
- InPool: A reference to target storage pool instance used for relocation.
- AvailableExtents: A group of references to available StorageExtents for relocation.

29.10 Client Considerations and Recipes

29.10.1 Client Considerations

29.10.1.1 Relocate StorageVolume to StoragePool for data migration

As one solution for data migration, StorageVolume can be relocated to a new local or remote concrete StoragePool.

- For local relocation, if 1) ‘SupportedStorageElementFeatures’ of StorageConfigurationCapabilities associating to the StorageRelocationService instance supports 'StorageVolume To StoragePool Relocation'; and 2) ‘SupportedStoragePoolUsage’ of StorageConfigurationCapabilities associating to the target StoragePool instance supports 'Used as target for Relocation Service', client can invoke StorageRelocationService.RelocateStorageVolumesToStoragePool to start the StorageVolume relocation onto the new concrete StoragePool.

Figure 151 shows the progress of this relocation.

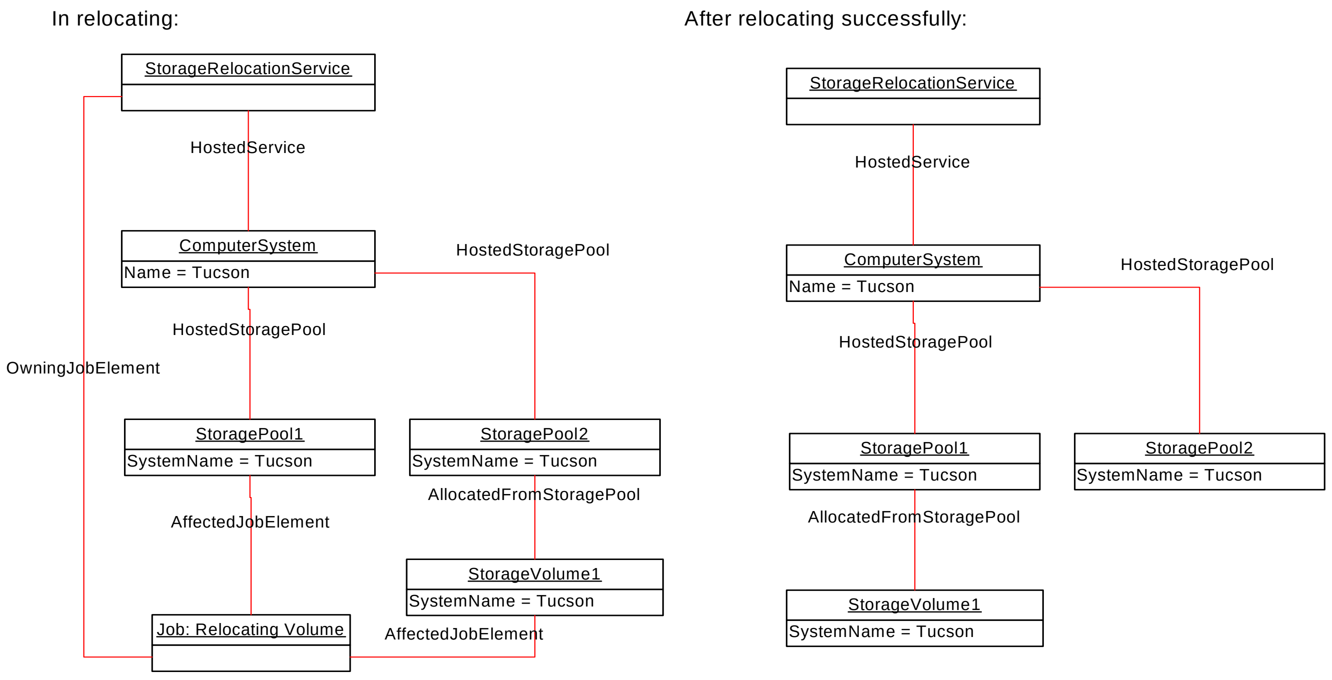

Figure 151 - Relocate StorageVolume to local StoragePool

If ‘SupportedAsynchronousActions’ of StorageConfigurationCapabilities supports 'StorageVolume To StoragePool Relocation', client can use the ‘Job’ output to track the progress of relocation. In the progress, Job.PercentComplete can be queried to track the completion percent. And through association ‘AffectedJobElement’, the target StoragePool instance with its property ‘CapacityInMigratingTarget’, the source StoragePool instance which the StorageVolume originally resides on with its property ‘CapacityInMigratingSource’, and StorageVolume with its property ‘NumExtentsMigrating’ can be queried out continuously to track the detailed progress of relocation. While in relocation, the ‘OperationalStatus’ of the source storage volume (StorageVolume1) will have value ‘Relocating’ to indicate it’s in relocation progress.

As an successful result, ‘CapacityInMigratingTarget’, ‘CapacityInMigratingSource’ and ‘NumExtentsMigrating’ should all be 0. The ‘OperationalStatus’ of the storage volume will change the value from ‘Relocating’ to ‘OK’. The key properties of the StorageVolume should stay the same.

- For remote relocation to remote storage pool, it needs to check the StorageConfigurationCapabilities in both local and remote storage system. So if 1) ‘SupportedStorageElementFeatures’ of StorageConfigurationCapabilities associating to the local StorageRelocationService instance supports 'StorageVolume To StoragePool Relocation'; and 2) ‘SupportedStoragePoolUsage’ of StorageConfigurationCapabilities associating to the remote target StoragePool instance supports 'Used as target for Relocation Service', client can invoke StorageRelocationService.RelocateStorageVolumesToStoragePool to start the StorageVolume relocation onto the remote concrete StoragePool.

Figure 152 shows the progress of this relocation.

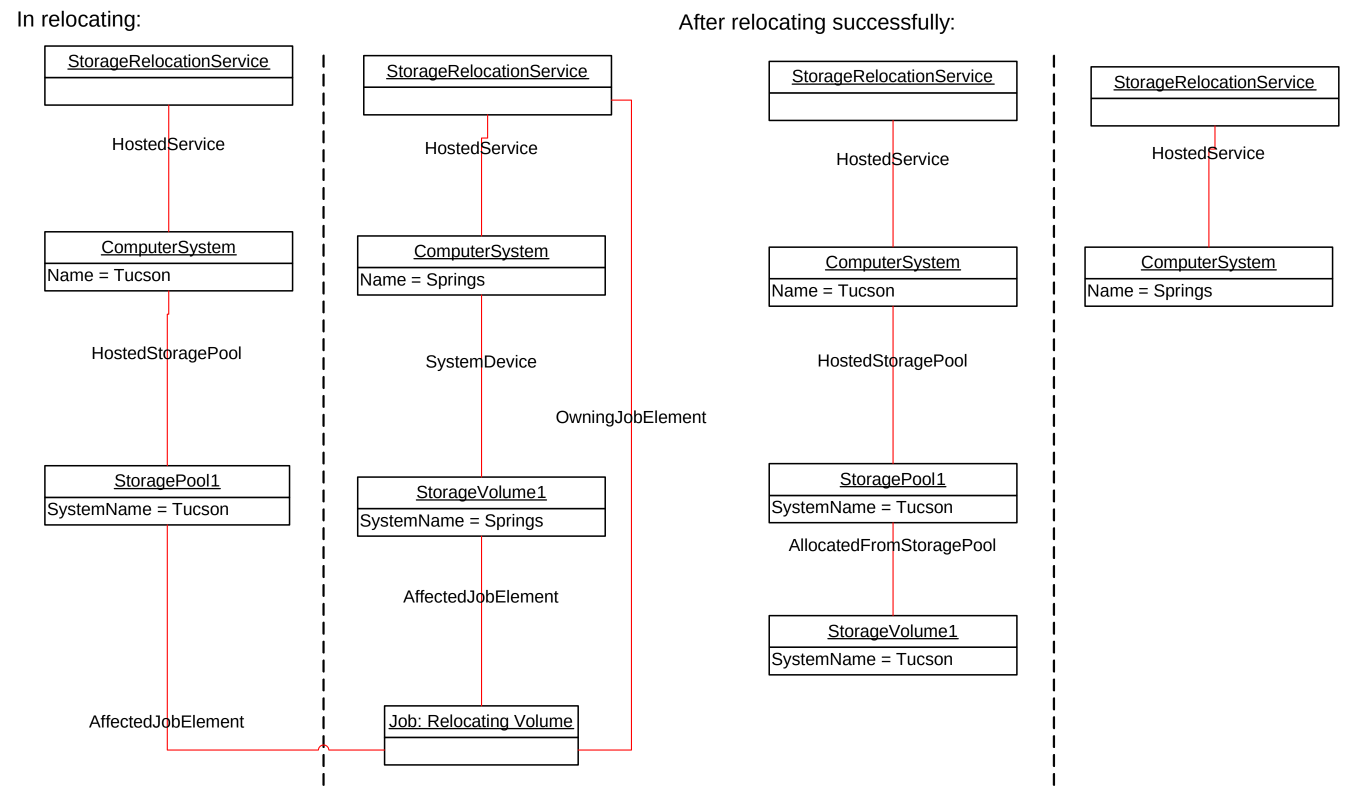

Figure 152 - Relocate StorageVolume to remote StoragePool

As same as the local relocation, the same mechanism can be used to track the relocation progress. And as an successful result, it is different in remote relocation that the key properties of the StorageVolume may not be able to stay the same as before, the source storage volume will disappear from the source storage device and appear on the target storage device, so relocating to a remote location may be a disruptive process.

29.10.1.2 Relocate StoragePool for merge

As one solution for extent merge or capacity expansion, one StoragePool can be relocated into a target local or remote StoragePool.

- For local relocation, if 1) 'SupportedStorageElementFeatures' of StorageConfigurationCapabilities associating to the StorageRelocationService instance supports 'StoragePool To StoragePool Relocation'; 2) 'SupportedStoragePoolUsage' of StorageConfigurationCapabilities associating to the source StoragePool instance supports 'Used as source for Relocation Service'; and 3) 'SupportedStoragePoolUsage' of StorageConfigurationCapabilities associating to the target StoragePool instance supports 'Used as target for Relocation Service', client can invoke StorageRelocationService.RelocateStoragePoolsToStoragePool to start the source StoragePool relocation onto the target StoragePool.

Figure 153 shows the progress of this relocation.

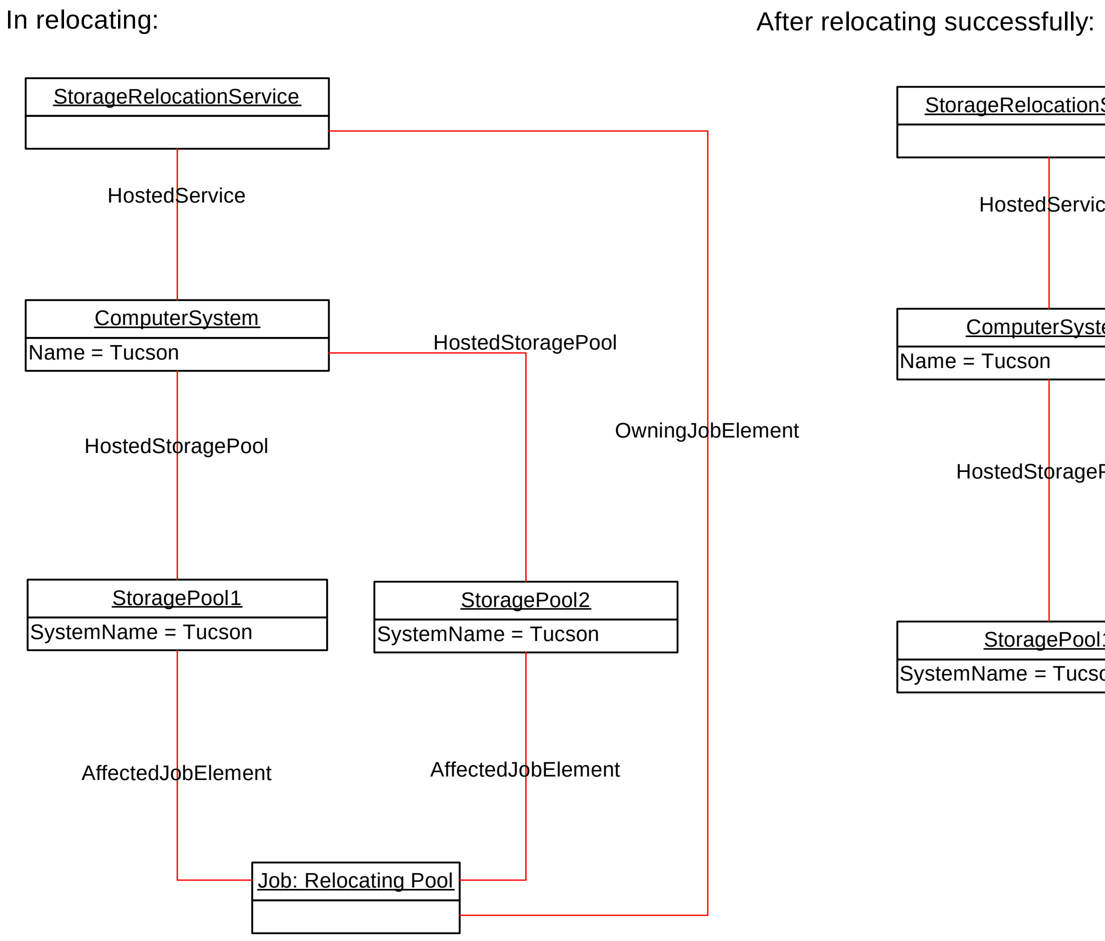

Figure 153 - Relocate StoragePool to local StoragePool

If ‘SupportedAsynchronousActions’ of StorageConfigurationCapabilities supports 'StoragePool To StoragePool Relocation', client can use the ‘Job’ output to track the progress of relocation. In the progress, Job.PercentComplete can be queried to track the completion percent. And through association ‘AffectedJobElement’, the source StoragePool instance with its property ‘CapacityInMigratingSource’, and the target StoragePool instance with its property ‘CapacityInMigratingTarget’ can be queried out continuously to track the detailed progress of relocation. While in relocation, the ‘OperationalStatus’ of the source storage pool (StoragePool2) will have value ‘Relocating’ to indicate it’s in relocation progress.

As an successful result, the source StoragePool won’t exist anymore, and target StoragePool’s ‘CapacityInMigratingTarget’ should be 0. The ‘OperationalStatus’ of the source storage pool will change the value from ‘Relocating’ to ‘OK’. All the children (volumes or sub pools) belonging to the source storage pool will be changed as children of target storage pool.

- For remote relocation, if 1) ‘SupportedStorageElementFeatures’ of StorageConfigurationCapabilities associating to the local StorageRelocationService instance supports 'StoragePool To StoragePool Relocation'; 2) ‘SupportedStoragePoolUsage’ of StorageConfigurationCapabilities associating to the local source StoragePool instance supports 'Used as source for Relocation Service'; and 3) ‘SupportedStoragePoolUsage’ of StorageConfigurationCapabilities associating to the remote target StoragePool instance supports 'Used as target for Relocation Service', client can invoke StorageRelocationService.RelocateStoragePoolsToStoragePool to start the local source StoragePool relocation onto the remote target StoragePool.

Figure 154 shows the progress of this relocation.

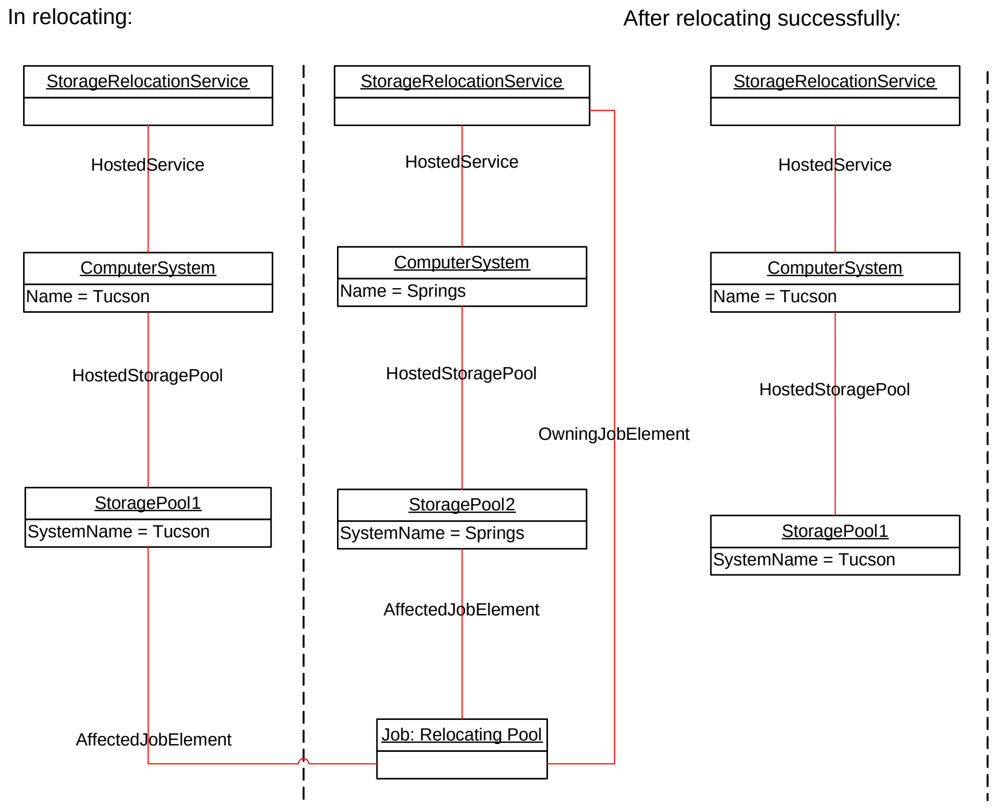

Figure 154 - Relocate StoragePool to remote StoragePool

As same as the local relocation, job instance can be used to track the relocation progress. And as the result of remote relocation, all the children (volumes or sub pools) belonging to the source storage pool will disappear from the source storage device, and appear as the children of target storage pool on remote target storage device. So remote location is a disruptive process.

Note: It is possible to merge a primordial StoragePool into a target StoragePool if the ‘SupportedStoragePoolUsage’ of StorageConfigurationCapabilities associating to the source primordial StoragePool instance supports 'Used as source for Relocation Service'. As implementation, merging primordial StoragePool can result in that all of the concrete StoragePools under that primordial StoragePool will be merged.

29.10.1.3 Relocate StorageVolume to new StorageExtent group for hotspot tuning

If a StorageVolume is identified as one hotspot volume after analyzing performance statistics of it, relocating it from low performance StorageExtent (e.g. HDD extent) group to high performance StorageExtent (e.g.SSD extent) group can serve as an solution on hotspot tuning. And if it becomes cold spot, it can be relocated back to low performance StorageExtent group.

- For local relocation, if 1) ‘SupportedStorageElementFeatures’ of StorageConfigurationCapabilities associating to the StorageRelocationService instance supports 'StorageVolume To StorageExtent Relocation'; 2) the new StorageExtent group is under one concrete StoragePool; and 3) ‘SupportedStoragePoolUsage’ of StorageConfigurationCapabilities associating to the target StoragePool instance supports 'Used as target for Relocation Service', the client can invoke StorageRelocationService.RelocateStorageVolumeToStorageExtents to start the StorageVolume relocation onto the new StorageExtent group.

Figure 155 shows the progress of this relocation.

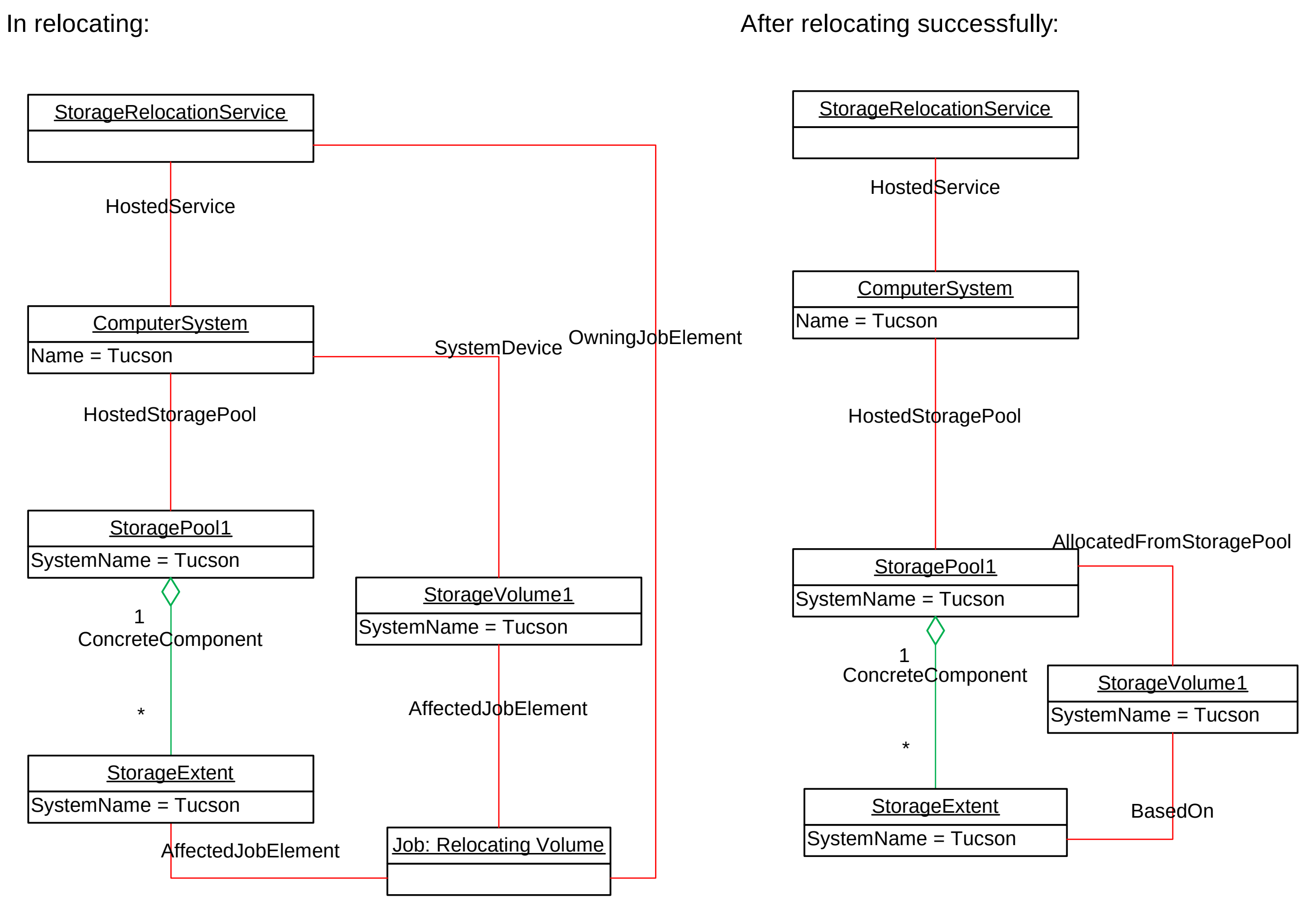

Figure 155 - Relocate StorageVolume to local StorageExtent group

If ‘SupportedAsynchronousActions’ of StorageConfigurationCapabilities supports 'StorageVolume To StorageExtent Relocation', client can use the ‘Job’ output to track the progress of relocation. In the progress, Job.PercentComplete can be queried to track the completion percent. And through association ‘AffectedJobElement’, the StorageVolume instance with its property ‘NumExtentsMigrating’, each associated StorageExtent instance in ‘Relocating’ status, and the target StoragePool instance with its property ‘CapacityInMigratingTarget’ can be queried out continuously to track the detailed progress of relocation. While in relocation, the ‘OperationalStatus’ of the source storage volume (StorageVolume1) will have value ‘Relocating’ to indicate it’s in relocation progress; and the ‘ExtentStatus’ of each new storage extent will have value ‘Relocating’ to indicate it’s in relocation progress.

As an successful result, both the target StoragePool’s ‘CapacityInMigratingTarget’ and the StorageVolume’s ‘NumExtentsMigrating’ should be 0. The ‘OperationalStatus’ of the source storage volume will change the value from ‘Relocating’ to ‘OK’. The ‘ExtentStatus’ of each new storage extent will remove the ‘Relocating’ value. The key properties of the StorageVolume stays the same.

For remote relocation, if 1) ‘SupportedStorageElementFeatures’ of StorageConfigurationCapabilities associating to the local StorageRelocationService instance supports 'StorageVolume To StorageExtent Relocation'; 2) the remote new StorageExtent group is under one remote concrete StoragePool; and 3) ‘SupportedStoragePoolUsage’ of StorageConfigurationCapabilities associating to the remote target StoragePool instance supports 'Used as target for Relocation Service', the client can invoke StorageRelocationService.relocateStorageVolumeToStorageExtents to start the StorageVolume relocation onto the remote new StorageExtent group.

Figure 156 shows the progress of this relocation.

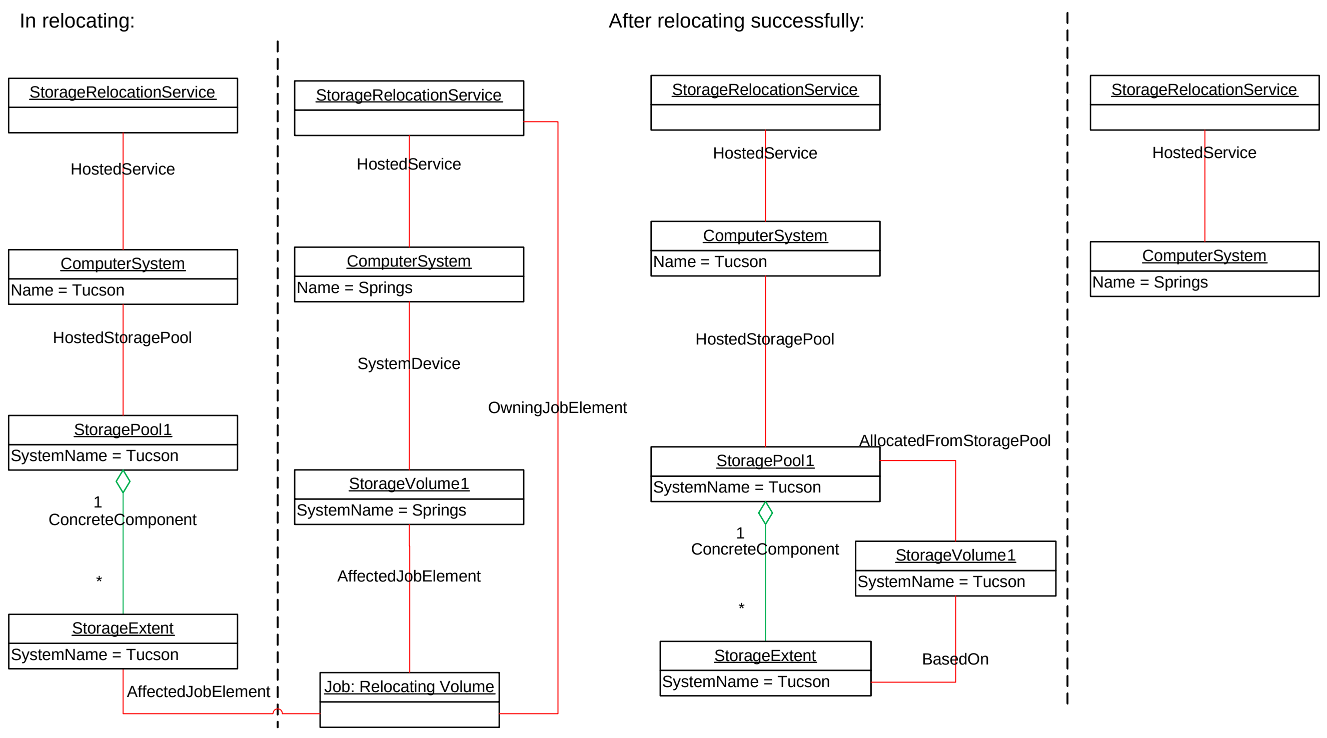

Figure 156 - Relocate StorageVolume to remote StorageExtent group

Like the local relocation, job instance can be used to track the relocation progress. And as the result of remote relocation, the source storage volume will disappear from the source storage device, and appear on the target storage device, so this remote location is a disruptive process.

Note: Relocating StorageVolume to new StorageExtent group can also be used as an solution for data migration, and the progress will be as same as the above.

29.10.2 Recipes

29.10.2.1 Relocate StorageVolumes to StoragePool

```
// DESCRIPTION
// The goal of this recipe is to relocate storage volumes to a local or remote
                          storage pool.
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
// 1.A reference to a CIM_ComputerSystem storage array is previously
//      defined in the $BlockServer-> variable
// 2.$RelocationTargetPool-> is set to reference of the target storage pool for
                          volume relocation
// 3.$VolumesToRelocate->[] is set to reference of the storage volumes to be
                          relocated.


// Step 1. Get the storage relocation services and determine the service
//   capabilities. Note that the device may not support storage relocation
//   so it is possible that the service is not present and
//   the desired operation cannot be performed.

try {
    $Services->[] = AssociatorNames($BlockServer->,
         “CIM_HostedService”,
         “CIM_StorageRelocationService”,
         null,
         null)

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

// There should be only one storage relocation service
// associated with the system
$StorageRelocationService-> = $Services->[0]

// See if the service supports storage volume to storage pool relocation
// There should be zero or only one StorageConfigurationCapabilities instance
$ServiceCapabilities[] = Associators(
     $StorageRelocationService->,
     “CIM_ElementCapabilities”,
     “CIM_StorageConfigurationCapabilities”,
     null, null, false, false, null)

if ($ServiceCapabilities[] == null ||
    $ServiceCapabilities[].length == 0) {
   <ERROR! Storage relocation not supported>
}

if (contains(14, // StorageVolume To StoragePool Relocation
        $ServiceCapabilities[0].SupportedStorageElementFeatures[]))
{
  #SupportsVolumeToStoragePoolRelocation = true
}
else {
  #SupportsVolumeToStoragePoolRelocation = false
  <ERROR! Storage volume to storage pool relocation not supported>
```

```
}

// Only one of SupportedSynchronousActions or SupportedAsynchronousActions
// should contain StorageVolume To StoragePool Relocation
if (contains(17, // StorageVolume To StoragePool Relocation
        $ServiceCapabilities[0].SupportedSynchronousActions[])
 == contains(17, // StorageVolume To StoragePool Relocation
     $ServiceCapabilities[0].SupportedAsynchronousActions[]))
{
  <ERROR! Storage volume to storage pool relocation should be supported
  in either SupportedSynchronousActions or SupportedAsynchronousActions>
}

if (contains(17, // StorageVolume To StoragePool Relocation
     $ServiceCapabilities[0].SupportedAsynchronousActions[]))
{
  #VolumeRelocationProducesJob = true
}

// Step 2. Get the StorageConfigurationCapabilities of the pool and check
// if it can be target pool for volume relocation

$PoolCapabilities[] = Associators(
        $RelocationTargetPool->,
        “CIM_ElementCapabilities”,
        “CIM_StorageConfigurationCapabilities”,
    null, null, false, false, null)

if ($PoolCapabilities[] == null ||
    $PoolCapabilities[].length == 0) {
   <ERROR! The target storage pool does not support storage relocation>
}

if ( contains(10, // Used as target for Relocation Service
     $PoolCapabilities[0].SupportedStoragePoolUsage[]))
{
  #Pool_TargetPoolCapability = true
}
else {
  #Pool_TargetPoolCapability = false
  <ERROR! The target storage pool can’t be used as target for relocation service>
}

if (contains(17, // StorageVolume To StoragePool Relocation
    $PoolCapabilities[0].SupportedAsyncronousActions[]))
{
  #Pool_RelocationProducesJob = true
```

```
}

// Step 3. Register for indications on relocation jobs

if(#Pool_RelocationProducesJob == true)
{
     // ‘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 4. Relocate the volumes to the Storage Pool

%InArguments[“TheElements”] = $VolumesToRelocate->[]
%InArguments[“Goal”]        = null
%InArguments[“TargetPool”]  = $RelocationTargetPool->

#ReturnValue = InvokeMethod(
        $StorageRelocationService->,
        “RelocateStorageVolumesToStoragePool”,
        %InArguments, %OutArguments)

if(#ReturnValue != 0 && #ReturnValue != 4096)
{   // The volume relocation operation failed
        <ERROR! Failed >
}

if(#Pool_RelocationProducesJob == true)
{
  $VolumeRelocationJob-> = %OutArguments[“Job”]

  // Get migrating status of one of the volumes
  $AffectedVolumes[] = Associators(
     $VolumeRelocationJob->,
     “CIM_AffectedJobElement”,
     “CIM_StorageVolume”,
     null, null, false, false, null)

  #NumExtentsMigrating_Vol = $AffectedVolumes[0].NumExtentsMigrating
  // The OperationalStatus of the volume could be 19 (Relocating)
  #OperationalStatus_Vol = $AffectedVolumes[0].OperationalStatus[0]

  // Get migrating status of the target pool
  $AffectedTargetPool[] = Associators(
     $VolumeRelocationJob->,
     “CIM_AffectedJobElement”,
     “CIM_StoragePool”,
     null, null, false, false, null)

  #MigratingIn_TargetPool = $AffectedTargetPool[0].CapacityInMigratingTarget
  // The OperationalStatus of the target pool could be 19 (Relocating)
  #OperationalStatus_TargetPool = $AffectedTargetPool[0].OperationalStatus[0]


     //Wait until the completion of the job
     // using $VolumeRelocationJob-> as a filter

  // Wait for indication from either filters defined in step 3
     // If the indication states the Job is ‘Complete’ and ‘Error’
     // then exit with error: ERROR! Job did not complete successfully
}
else {
  // The volumes that are successfully relocated are output in TheElements
  $RelocatedVolumes-> = %OutArguments[“TheElements”]
}
```

29.10.2.2 Relocate StoragePool to StoragePool

```
// DESCRIPTION
// The goal of this recipe is to merge a storage pool to another (local or remote)
                           storage pool
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
// 1.A reference to a CIM_ComputerSystem storage array is previously
//       defined in the $BlockServer-> variable
// 2.$TargetPool-> is set to reference of the target storage pool
// 3.$PoolsToRelocate->[] is set to reference of the storage pools to be
                           relocated.



// Step 1. Get the storage relocation services and determine the service
//   capabilities. Note that the device may not support storage relocation
//   so it is possible that the service is not present and
//   the desired operation cannot be performed.

try {
    $Services->[] = AssociatorNames($BlockServer->,
         “CIM_HostedService”,
         “CIM_StorageRelocationService”,
         null,
         null)

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

// There should be only one storage relocation service
// associated with the system
$StorageRelocationService-> = $Services->[0]

// See if the service supports storage pool to storage pool relocation
// There should be zero or only one StorageConfigurationCapabilities instance
$ServiceCapabilities[] = Associators(
     $StorageRelocationService->,
     “CIM_ElementCapabilities”,
     “CIM_StorageConfigurationCapabilities”,
     null, null, false, false, null)

if ($ServiceCapabilities[] == null ||
    $ServiceCapabilities[].length == 0) {
   <ERROR! Storage relocation not supported>
}

if (contains(15, // StoragePool To StoragePool Relocation
        $ServiceCapabilities[0].SupportedStorageElementFeatures[]))
{
  #SupportsPoolToPoolRelocation = true
}
else {
  #SupportsPoolToPoolRelocation = false
  <ERROR! Storage pool to storage pool relocation not supported>
}

// Only one of SupportedSynchronousActions or SupportedAsynchronousActions
// should contain StoragePool To StoragePool Relocation
if (contains(18, // StoragePool To StoragePool Relocation
        $ServiceCapabilities[0].SupportedSynchronousActions[])
  == contains(18, // StoragePool To StoragePool Relocation
     $ServiceCapabilities[0].SupportedAsynchronousActions[]))
{
  <ERROR! Storage pool to storage pool relocation should be supported
  in either SupportedSynchronousActions or SupportedAsynchronousActions>
}

if (contains(18, // StoragePool To StoragePool Relocation
     $ServiceCapabilities[0].SupportedAsynchronousActions[]))
{
  #PoolRelocationProducesJob = true
}

// Step 2. Get the StorageConfigurationCapabilities of the pools to be relocated
                           and check
// if they can be source pools in storage pool relocation
for #i in $PoolsToRelocate->[]
{
$SourcePoolCapabilities[] = Associators(
        $PoolsToRelocate->[#i],
        “CIM_ElementCapabilities”,
        “CIM_StorageConfigurationCapabilities”,
    null, null, false, false, null)

if ($SourcePoolCapabilities[] == null ||
    $SourcePoolCapabilities[].length == 0) {
   <ERROR! The source storage pool does not support storage relocation>
}

if ( contains(9, // Used as source for Relocation Service
     $SourcePoolCapabilities[0].SupportedStoragePoolUsage[]))
{
  #Pool_SourcePoolCapability = true
}
else {
  #Pool_SourcePoolCapability = false
  <ERROR! The source storage pool can’t be used as source for relocation service>
}
}

// Step 3. Get the StorageConfigurationCapabilities of the target pool and check
// if it can be target storage pool in storage pool relocation
```

```
$TargetPoolCapabilities[] = Associators(
        $TargetPool->,
        “CIM_ElementCapabilities”,
        “CIM_StorageConfigurationCapabilities”,
    null, null, false, false, null)

if ($TargetPoolCapabilities[] == null ||
    $TargetPoolCapabilities[].length == 0) {
   <ERROR! The target storage pool does not support storage relocation>
}

if ( contains(10, // Used as target for Relocation Service
     $TargetPoolCapabilities[0].SupportedStoragePoolUsage[]))
{
  #Pool_TargetPoolCapability = true
}
else {
  #Pool_TargetPoolCapability = false
  <ERROR! The target storage pool can’t be used as target for relocation service>
}

// Step 4. Register for indications on relocation jobs
if(#PoolRelocationProducesJob == true)
{
     // ‘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 5. Relocate the source pools to the target pool

%InArguments[“TheElements”] = $PoolsToRelocate->[]
%InArguments[“Goal”]        = null
%InArguments[“TargetPool”]  = $TargetPool->
```

```
#ReturnValue = InvokeMethod(
        $StorageRelocationService->,
        “RelocateStoragePoolsToStoragePool”,
        %InArguments, %OutArguments)

if(#ReturnValue != 0 && #ReturnValue != 4096)
{   // The storage pool relocation operation failed
        <ERROR! Failed >
}

if(#PoolRelocationProducesJob == true)
{
  $PoolRelocationJob-> = %OutArguments[“Job”]


  $AffectedStoragePools[] = Associators(
     $PoolRelocationJob->,
     “CIM_AffectedJobElement”,
     “CIM_StoragePool”,
     null, null, false, false, null)
  // Get migrating status of one of the source pools
  for #i in $AffectedStoragePools[]
  {
  if ($AffectedStoragePools[#i] != $TargetPool) {
     $Affected_SourcePool = $AffectedStoragePools[#i]
     break
    }
  }

  #MigratingOut_SourcePool = $Affected_SourcePool.CapacityInMigratingSource
  // The OperationalStatus of the source pool could be 19 (Relocating)
  #OperationalStatus_SourcePool = $Affected_SourcePool.OperationalStatus[0]

  // Get migrating status of the target pool
  #MigratingIn_TargetPool = $TargetPool.CapacityInMigratingTarget
  // The OperationalStatus of the target pool could be 19 (Relocating)
  #OperationalStatus_TargetPool = $TargetPool.OperationalStatus[0]

     //Wait until the completion of the job
     // using $PoolRelocationJob-> as a filter

  // Wait for indication from either filters defined in step 4
     // If the indication states the Job is ‘Complete’ and ‘Error’
     // then exit with error: ERROR! Job did not complete successfully
}
```

29.10.2.3 Relocate StorageVolume to StorageExtents

```
// DESCRIPTION
// The goal of this recipe is to relocate storage volume onto a new group of
                           storage extents
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
// 1.A reference to a CIM_ComputerSystem storage array is previously
//      defined in the $BlockServer-> variable
// 2.$TargetPool-> is set to reference of the target storage pool from which
                           target extents are allocated for volume relocation
// 3.$VolumeToRelocate-> is set to reference of the storage volume to be
                           relocated.


// Step 1. Get the storage relocation services and determine the service
//   capabilities. Note that the device may not support storage relocation
//   so it is possible that the service is not present and
//   the desired operation cannot be performed.

try {
    $Services->[] = AssociatorNames($BlockServer->,
         “CIM_HostedService”,
         “CIM_StorageRelocationService”,
         null,
         null)

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

// There should be only one storage relocation service
// associated with the system
$StorageRelocationService-> = $Services->[0]

// See if the service supports storage volume to storage extent relocation
// There should be zero or only one StorageConfigurationCapabilities instance
$ServiceCapabilities[] = Associators(
     $StorageRelocationService->,
     “CIM_ElementCapabilities”,
     “CIM_StorageConfigurationCapabilities”,
     null, null, false, false, null)

if ($ServiceCapabilities[] == null ||
    $ServiceCapabilities[].length == 0) {
   <ERROR! Storage relocation not supported>
}

if (contains(16, // StorageVolume To StorageExtent Relocation
        $ServiceCapabilities[0].SupportedStorageElementFeatures[]))
{
  #SupportsVolumeToExtentRelocation = true
}
else {
  #SupportsVolumeToExtentRelocation = false
  <ERROR! Storage volume to storage extent relocation not supported>
}

// Only one of SupportedSynchronousActions or SupportedAsynchronousActions
// should contain StorageVolume To StorageExtent Relocation
if (contains(19, // StorageVolume To StorageExtent Relocation
        $ServiceCapabilities[0].SupportedSynchronousActions[])
 == contains(19, // StorageVolume To StorageExtent Relocation
     $ServiceCapabilities[0].SupportedAsynchronousActions[]))
{
  <ERROR! Storage volume to storage extent relocation should be supported
  in either SupportedSynchronousActions or SupportedAsynchronousActions>
}

if (contains(19, // StorageVolume To StorageExtent Relocation
     $ServiceCapabilities[0].SupportedAsynchronousActions[]))
{
  #VolumeToExtentProducesJob = true
}

// Step 2. Get available target extents on target pool

%InArguments[“TheElement”]  = $VolumeToRelocate->
%InArguments[“InPool”]  = $TargetPool->
%InArguments[“TargetSettingGoal”]= null

#ReturnValue = InvokeMethod(
        $StorageRelocationService->,
        “GetAvailableTargetRelocationExtents”,
        %InArguments, %OutArguments)

if( #ReturnValue != 0 )
```

```
{   // GetAvailableTargetRelocationExtents failed
        <ERROR! Failed >
}

if( %OutArguments[“AvailableExtents”] == null )
{   // There is no available relocation extent
        <ERROR! Failed >
}
$TargetExtents->[] = %OutArguments[“AvailableExtents”]

// Step 3. Register for indications on relocation jobs

if(#VolumeToExtentProducesJob == true)
{
     // ‘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 4. Relocate the volume to the extents

%InArguments[“InElements”]  = $TargetExtents->[]
%InArguments[“Goal”]        = null
%InArguments[“TheElement”]  = $VolumeToRelocate->

#ReturnValue = InvokeMethod(
        $StorageRelocationService->,
        “RelocateStorageVolumeToStorageExtents”,
        %InArguments, %OutArguments)

if(#ReturnValue != 0 && #ReturnValue != 4096)
{   // The volume relocation operation failed
        <ERROR! Failed >
}
```

```
if(#VolumeToExtentProducesJob == true)
{
  $VolumeRelocationJob-> = %OutArguments["Job"]

  // Get migrating status of the source volume
  $AffectedVolume[] = Associators(
     $VolumeRelocationJob->,
     "CIM_AffectedJobElement",
     "CIM_StorageVolume",
     null, null, false, false, null)

  #NumExtentsMigrating_Vol = $AffectedVolume[0].NumExtentsMigrating
  // The OperationalStatus of the volume could be 19 (Relocating)
  #OperationalStatus_Vol = $AffectedVolume[0].OperationalStatus[0]

  // Get migrating status of one of the target extents
  $AffectedTargetExtents[] = Associators(
     $VolumeRelocationJob->,
     "CIM_AffectedJobElement",
     "CIM_StorageExtent",
     null, null, false, false, null)
  // The extent status of the target extent could be 18 (Relocating)
  #TargetExtentStatus = $AffectedTargetExtents[0].ExtentStatus

     //Wait until the completion of the job
     // using $VolumeRelocationJob-> as a filter

  // Wait for indication from either filters defined in step 2
     // If the indication states the Job is 'Complete' and 'Error'
     // then exit with error: ERROR! Job did not complete successfully
}
else {
  // The relocated volume is output in TheElement
  $RelocatedVolume-> = %OutArguments["TheElement"]
}
```

29.10.2.4 Relocate StoragePool to StorageExtents

```
// DESCRIPTION
// The goal of this recipe is to relocate a storage pool onto a new group of
                           storage extents
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
// 1.A reference to a CIM_ComputerSystem storage array is previously
//      defined in the $BlockServer-> variable
// 2.$TargetPool-> is set to reference of the target storage pool from which
                           target extents are allocated for pool relocation
// 3.$PoolToRelocate-> is set to reference of the storage pool to be relocated.
// Step 1. Get the storage relocation services and determine the service
//   capabilities. Note that the device may not support storage relocation
//   so it is possible that the service is not present and
//   the desired operation cannot be performed.

try {
    $Services->[] = AssociatorNames($BlockServer->,
         "CIM_HostedService",
         "CIM_StorageRelocationService",
         null,
         null)

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

// There should be only one storage relocation service
// associated with the system
$StorageRelocationService-> = $Services->[0]

// See if the service supports storage pool to storage extent relocation
// There should be zero or only one StorageConfigurationCapabilities instance
$ServiceCapabilities[] = Associators(
     $StorageRelocationService->,
     "CIM_ElementCapabilities",
     "CIM_StorageConfigurationCapabilities",
     null, null, false, false, null)

if ($ServiceCapabilities[] == null ||
    $ServiceCapabilities[].length == 0) {
   <ERROR! Storage relocation not supported>
}

if (contains(17, // StoragePool To StorageExtent Relocation
        $ServiceCapabilities[0].SupportedStorageElementFeatures[]))
{
  #SupportsPoolToExtentRelocation = true
}
else {
  #SupportsPoolToExtentRelocation = false
  <ERROR! Storage pool to storage extent relocation not supported>
}

// Only one of SupportedSynchronousActions or SupportedAsynchronousActions
// should contain StoragePool To StorageExtent Relocation
if (contains(20, // StoragePool To StorageExtent Relocation
        $ServiceCapabilities[0].SupportedSynchronousActions[])
 == contains(20, // StoragePool To StorageExtent Relocation
     $ServiceCapabilities[0].SupportedAsynchronousActions[]))
{
  <ERROR! Storage pool to storage extent relocation should be supported
  in either SupportedSynchronousActions or SupportedAsynchronousActions>
}

if (contains(20, // StoragePool To StorageExtent Relocation
     $ServiceCapabilities[0].SupportedAsynchronousActions[]))
{
  #PoolToExtentProducesJob = true
}

// Step 2. Get available target extents on target pool

%InArguments["TheElement"]  = $PoolToRelocate->
%InArguments["InPool"]  = $TargetPool->
%InArguments["TargetSettingGoal"]= null

#ReturnValue = InvokeMethod(
        $StorageRelocationService->,
        "GetAvailableTargetRelocationExtents",
        %InArguments, %OutArguments)

if( #ReturnValue != 0 )
{   // GetAvailableTargetRelocationExtents failed
        <ERROR! Failed >
}

if( %OutArguments["AvailableExtents"] == null )
{   // There is no available relocation extent
        <ERROR! Failed >
}
$TargetExtents->[] = %OutArguments["AvailableExtents"]

// Step 3. Register for indications on relocation jobs
```

```
if(#PoolToExtentProducesJob == true)
{
     // '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 4. Relocate the pool to the extents

%InArguments["InElements"]  = $TargetExtents->[]
%InArguments["Goal"]        = null
%InArguments["TheElement"]  = $PoolToRelocate->

#ReturnValue = InvokeMethod(
        $StorageRelocationService->,
        "RelocateStoragePoolToStorageExtents",
        %InArguments, %OutArguments)

if(#ReturnValue != 0 && #ReturnValue != 4096)
{   // The pool relocation operation failed
        <ERROR! Failed >
}

if(#PoolToExtentProducesJob == true)
{
  $PoolRelocationJob-> = %OutArguments["Job"]

  // Get migrating status of the source pool
  $AffectedPool[] = Associators(
     $PoolRelocationJob->,
     "CIM_AffectedJobElement",
     "CIM_StoragePool",
     null, null, false, false, null)

  #MigratingOut_SourcePool = $AffectedPool[0].CapacityInMigratingSource
  // The OperationalStatus of the pool could be 19 (Relocating)
  #OperationalStatus_SourcePool = $AffectedPool[0].OperationalStatus[0]

  // Get migrating status of one of the target extents
  $AffectedTargetExtents[] = Associators(
     $PoolRelocationJob->,
     "CIM_AffectedJobElement",
     "CIM_StorageExtent",
     null, null, false, false, null)
  // The extent status of the target extent could be 18 (Relocating)
  #TargetExtentStatus = $AffectedTargetExtents[0].ExtentStatus

     //Wait until the completion of the job
     // using $PoolRelocationJob-> as a filter

  // Wait for indication from either filters defined in step 2
     // If the indication states the Job is 'Complete' and 'Error'
     // then exit with error: ERROR! Job did not complete successfully
}
else {
  // The relocated pool is output in TheElement
  $RelocatedPool-> = %OutArguments["TheElement"]
}
```

29.10.2.5 Relocate LogicalDisk to StorageExtents

```
// DESCRIPTION
// The goal of this recipe is to relocate a logical disk onto a new group of
                          storage extents
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
// 1.A reference to a CIM_ComputerSystem storage array is previously
//      defined in the $BlockServer-> variable
// 2.$TargetPool-> is set to reference of the target storage pool from which
                          target extents are allocated for logical disk
                          relocation
// 3.$LogicalDiskToRelocate-> is set to reference of the logical disk to be
                          relocated.


// Step 1. Get the storage relocation services and determine the service
//   capabilities. Note that the device may not support storage relocation
//   so it is possible that the service is not present and
//   the desired operation cannot be performed.

try {
    $Services->[] = AssociatorNames($BlockServer->,
         “CIM_HostedService”,
         “CIM_StorageRelocationService”,
         null,
          null)

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

// There should be only one storage relocation service
// associated with the system
$StorageRelocationService-> = $Services->[0]

// See if the service supports logical disk to storage extent relocation
// There should be zero or only one StorageConfigurationCapabilities instance
$ServiceCapabilities[] = Associators(
     $StorageRelocationService->,
     “CIM_ElementCapabilities”,
     “CIM_StorageConfigurationCapabilities”,
     null, null, false, false, null)

if ($ServiceCapabilities[] == null ||
    $ServiceCapabilities[].length == 0) {
   <ERROR! Storage relocation not supported>
}

if (contains(18, // LogicalDisk To StorageExtent Relocation
        $ServiceCapabilities[0].SupportedStorageElementFeatures[]))
{
  #SupportsLogicalDiskToExtentRelocation = true
}
else {
  #SupportsLogicalDiskToExtentRelocation = false
  <ERROR! Storage pool to storage extent relocation not supported>
}

// Only one of SupportedSynchronousActions or SupportedAsynchronousActions
// should contain LogicalDisk To StorageExtent Relocation
if (contains(21, // LogicalDisk To StorageExtent Relocation
        $ServiceCapabilities[0].SupportedSynchronousActions[])
 == contains(21, // LogicalDisk To StorageExtent Relocation
     $ServiceCapabilities[0].SupportedAsynchronousActions[]))
{
  <ERROR! Logical disk to storage extent relocation should be supported
  in either SupportedSynchronousActions or SupportedAsynchronousActions>
}

if (contains(21, // LogicalDisk To StorageExtent Relocation
     $ServiceCapabilities[0].SupportedAsynchronousActions[]))
{
  #LogicalDiskToExtentProducesJob = true
}

// Step 2. Get available target extents on target pool

%InArguments[“TheElement”]  = $LogicalDiskToRelocate->
%InArguments[“InPool”]  = $TargetPool->
%InArguments[“TargetSettingGoal”]= null

#ReturnValue = InvokeMethod(
        $StorageRelocationService->,
        “GetAvailableTargetRelocationExtents”,
        %InArguments, %OutArguments)

if( #ReturnValue != 0 )
{   // GetAvailableTargetRelocationExtents failed
        <ERROR! Failed >
}

if( %OutArguments[“AvailableExtents”] == null )
{   // There is no available relocation extent
        <ERROR! Failed >
}
$TargetExtents->[] = %OutArguments[“AvailableExtents”]

// Step 3. Register for indications on relocation jobs

if(#LogicalDiskToExtentProducesJob == true)
{
     // ‘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 4. Relocate the logical disk to the extents

%InArguments[“InElements”]  = $TargetExtents->[]
%InArguments[“Goal”]        = null
%InArguments[“TheElement”]  = $LogicalDiskToRelocate->

#ReturnValue = InvokeMethod(
        $StorageRelocationService->,
        “RelocateLogicalDiskToStorageExtents”,
        %InArguments, %OutArguments)

if(#ReturnValue != 0 && #ReturnValue != 4096)
{   // The logical disk relocation operation failed
        <ERROR! Failed >
}

if(#LogicalDiskToExtentProducesJob == true)
{
  $LogicalDiskRelocationJob-> = %OutArguments[“Job”]

  // Get migrating status of the source logical disk
  $AffectedLogicalDisk[] = Associators(
     $LogicalDiskRelocationJob->,
     “CIM_AffectedJobElement”,
     “CIM_LogicalDisk”,
     null, null, false, false, null)

  #MigratingOut_SourceLogicalDisk = $AffectedLogicalDisk[0].NumExtentsMigrating
  // The OperationalStatus of the logical disk could be 19 (Relocating)
  #OperationalStatus_SourceLogicalDisk =
                         $AffectedLogicalDisk[0].OperationalStatus[0]

  // Get migrating status of one of the target extents
  $AffectedTargetExtents[] = Associators(
     $LogicalDiskRelocationJob->,
     “CIM_AffectedJobElement”,
     “CIM_StorageExtent”,
     null, null, false, false, null)
  // The extent status of the target extent could be 18 (Relocating)
  #TargetExtentStatus = $AffectedTargetExtents[0].ExtentStatus
```

```
                //Wait until the completion of the job
                // using $LogicalDiskRelocationJob-> as a filter

            // Wait for indication from either filters defined in step 2
                // If the indication states the Job is ‘Complete’ and ‘Error’
                // then exit with error: ERROR! Job did not complete successfully
          }
          else {
            // The relocated pool is output in TheElement
            $RelocatedLogicalDisk-> = %OutArguments[“TheElement”]
          }
```

29.11 Registered Name and Version

Storage Relocation version 1.6.0 (Component Profile)

CIM Schema Version: 2.29

29.12 CIM Elements

Table 626 describes the CIM elements for Storage Relocation.

Table 626 - CIM Elements for Storage Relocation

Element Name	Requirement	Description
29.12.1 CIM_AffectedJobElement (LogicalDisk to ConcreteJob)	Conditional	Conditional requirement: This is required if CIM_StorageConfigurationCapabilities.SupportedAsynchronousActions contains '21' (async disk operations). This AffectedJobElement represents an association between a Job and the LogicalDisks(s) that may be affected by its execution.
29.12.2 CIM_AffectedJobElement (StorageExtent to ConcreteJob)	Conditional	Conditional requirement: This is required if CIM_StorageConfigurationCapabilities.SupportedAsynchronousActions contains '19|20|21' (async extent operations). AffectedJobElement represents an association between a Job and the storage extent(s) that may be affected by its execution.

Table 626 - CIM Elements for Storage Relocation

Element Name	Requirement	Description
29.12.3 CIM_AffectedJobElement (StoragePool to ConcreteJob)	Conditional	Conditional requirement: This is required if CIM_StorageConfigurationCapabilities.SupportedAsynchronousActions contains '17|18|20' (async pool operations). AffectedJobElement represents an association between a Job and the storage pool(s) that may be affected by its execution.
29.12.4 CIM_AffectedJobElement (StorageVolume to ConcreteJob)	Conditional	Conditional requirement: This is required if CIM_StorageConfigurationCapabilities.SupportedAsynchronousActions contains '17|19' (async volume operations). AffectedJobElement represents an association between a Job and the storage volumes(s) that may be affected by its execution.
29.12.5 CIM_ElementCapabilities (StorageConfigurationCapabilities to StoragePool)	Mandatory	Associates StorageConfigurationCapabilities with StoragePool. This ElementCapabilities shall represent the capabilities of the StoragePool to which it is associated.
29.12.6 CIM_ElementCapabilities (StorageConfigurationCapabilities to StorageRelocationService)	Mandatory	Associates the global StorageConfigurationCapabilities with StorageRelocationservice. This ElementCapabilities shall represent the capabilities that StorageRelocationService can provide.
29.12.7 CIM_FilterCollection (Storage Relocation 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.
29.12.8 CIM_FilterCollection (Storage Relocation ProfileSpecificAlertIndicationFilterCollection)	Optional	Experimental. This is a collection of Lifecycle IndicationFilters defined by the Storage Relocation Profile.
29.12.9 CIM_HostedCollection (System to Storage Relocation ProfileSpecificAlertIndicationFilterCollection)	Optional	Experimental.
29.12.10 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).

Table 626 - CIM Elements for Storage Relocation

Element Name	Requirement	Description
29.12.11 CIM_HostedService (StorageRelocationService to ComputerSystem)	Mandatory	HostedService represents an association between the scoping System and the StorageRelocationService.
29.12.12 CIM_IndicationFilter (LogicalDiskRelocationEnd)	Optional	Experimental. This is the 'pre-defined' CIM_IndicationFilter instance for the end of a storage relocation of a logical disk instance.
29.12.13 CIM_IndicationFilter (LogicalDiskRelocationStart)	Optional	Experimental. This is the 'pre-defined' CIM_IndicationFilter instance for the start of a storage relocation of a logical disk instance.
29.12.14 CIM_IndicationFilter (StoragePoolRelocationEnd)	Optional	Experimental. This is the 'pre-defined' CIM_IndicationFilter instance for the end of a storage relocation of a storage pool instance.
29.12.15 CIM_IndicationFilter (StoragePoolRelocationStart)	Optional	Experimental. This is the 'pre-defined' CIM_IndicationFilter instance for the start of a storage relocation of a storage pool instance.
29.12.16 CIM_IndicationFilter (VolumeRelocationEnd)	Optional	Experimental. This is the 'pre-defined' CIM_IndicationFilter instance for the end of a storage relocation of a storage volume instance.
29.12.17 CIM_IndicationFilter (VolumeRelocationStart)	Optional	Experimental. This is the 'pre-defined' CIM_IndicationFilter instance for the start of a storage relocation of a storage volume instance.
29.12.18 CIM_LogicalDisk	Conditional	Conditional requirement: Referenced from Volume Management - LogicalDisk is mandatory. The CIM_LogicalDisk is an augmented version of the CIM_LogicalDisk defined in the Block Services package. See CIM_LogicalDisk in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.32 CIM_LogicalDisk.
29.12.19 CIM_MemberOfCollection (Predefined Filter Collection to Storage Relocation 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 Relocation predefined FilterCollection to the predefined Filters supported by the implementation.

Table 626 - CIM Elements for Storage Relocation

Element Name	Requirement	Description
29.12.20 CIM_MemberOfCollection (Storage Relocation 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 Storage Relocation predefined FilterCollection to the FilterCollection for the autonomous profile (e.g., the Array FilterCollection).
29.12.21 CIM_MemberOfCollection (Storage Relocation ProfileSpecificAlertIndicationFilterCollection to Storage Relocation Filters)	Optional	Experimental. This associates the Storage Relocation ProfileSpecificAlertIndicationFilterCollection to the Filters defined by the Storage Relocation Profile.
29.12.22 CIM_OwningJobElement (StorageRelocationService to ConcreteJob)	Conditional	Conditional requirement: This is required if CIM_StorageConfigurationCapabilities.SupportedAsynchronousActions contains '17|18|19|20|21' (async relocation operations). OwningJobElement represents an association between a Job and the StorageRelocationService that initiated its execution.
29.12.23 CIM_StorageConfigurationCapabilities (Concrete)	Mandatory	The Concrete CIM_StorageConfigurationCapabilities an augmented version of the Concrete CIM_StorageConfigurationCapabilities defined in the Block Services package. See CIM_StorageConfigurationCapabilities (Concrete) in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.38 CIM_StorageConfigurationCapabilities (Concrete).
29.12.24 CIM_StorageConfigurationCapabilities (Global)	Mandatory	The global CIM_StorageConfigurationCapabilities an augmented version of the global CIM_StorageConfigurationCapabilities defined in the Block Services package. See CIM_StorageConfigurationCapabilities (Global) in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.39 CIM_StorageConfigurationCapabilities (Global).

Table 626 - CIM Elements for Storage Relocation

Element Name	Requirement	Description
29.12.25 CIM_StorageConfigurationCapabilities (Primordial)	Mandatory	The primordial CIM_StorageConfigurationCapabilities an augmented version of the primordial CIM_StorageConfigurationCapabilities defined in the Block Services package. See CIM_StorageConfigurationCapabilities (Primordial) in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.40 CIM_StorageConfigurationCapabilities (Primordial).
29.12.26 CIM_StorageExtent (Relocatable)	Optional	A StorageExtent that can be a source or target of a relocation operation.
29.12.27 CIM_StoragePool (Concrete)	Mandatory	The concrete CIM_StoragePool an augmented version of the concrete CIM_StoragePool defined in the Block Services package. See CIM_StoragePool (Concrete) in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.42 CIM_StoragePool (Concrete).
29.12.28 CIM_StoragePool (Primordial)	Mandatory	The primordial CIM_StoragePool an augmented version of the primordial CIM_StoragePool defined in the Block Services package. See CIM_StoragePool (Primordial) in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.44 CIM_StoragePool (Primordial).
29.12.29 CIM_StorageRelocationService	Mandatory	Experimental. CIM_StorageRelocationService that provides the storage relocation methods.
29.12.30 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. The CIM_StorageVolume is an augmented version of the CIM_StorageVolume defined in the Block Services package. See CIM_StorageVolume in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.49 CIM_StorageVolume.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity='SNIA' and MessageID='DRM32'	Optional	Indication that StorageVolume relocation starts.

Table 626 - CIM Elements for Storage Relocation

Element Name	Requirement	Description
SELECT * FROM CIM_AlertIndication WHERE OwningEntity='SNIA' and MessageID='DRM33'	Optional	Indication that StorageVolume relocation ends.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity='SNIA' and MessageID='DRM34'	Optional	Indication that StoragePool relocation starts.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity='SNIA' and MessageID='DRM35'	Optional	Indication that StoragePool relocation ends.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity='SNIA' and MessageID='DRM36'	Optional	Indication that LogicalDisk relocation starts.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity='SNIA' and MessageID='DRM37'	Optional	Indication that LogicalDisk relocation ends.

29.12.1 CIM_AffectedJobElement (LogicalDisk to ConcreteJob)

Requirement: This is required if CIM_StorageConfigurationCapabilities.SupportedAsynchronousActions contains '21' (async disk operations).

Table 627 describes class CIM_AffectedJobElement (LogicalDisk to ConcreteJob).

Table 627 - SMI Referenced Properties/Methods for CIM_AffectedJobElement (LogicalDisk to ConcreteJob)

Properties	Flags	Requirement	Description & Notes
AffectedElement		Mandatory	A reference to a CIM_LogicalDisk instance that is affected by the execution of the job.
AffectingElement		Mandatory	The job that is affecting the logical disk.

29.12.2 CIM_AffectedJobElement (StorageExtent to ConcreteJob)

Requirement: This is required if CIM_StorageConfigurationCapabilities.SupportedAsynchronousActions contains '19|20|21' (async extent operations).

Table 628 describes class CIM_AffectedJobElement (StorageExtent to ConcreteJob).

Table 628 - SMI Referenced Properties/Methods for CIM_AffectedJobElement (StorageExtent to ConcreteJob)

Properties	Flags	Requirement	Description & Notes
AffectedElement		Mandatory	A reference to a CIM_StorageExtent instance that is affected by the execution of the job.
AffectingElement		Mandatory	The job that is affecting the storage extent.

29.12.3 CIM_AffectedJobElement (StoragePool to ConcreteJob)

Requirement: This is required if CIM_StorageConfigurationCapabilities.SupportedAsynchronousActions contains '17|18|20' (async pool operations).

Table 629 describes class CIM_AffectedJobElement (StoragePool to ConcreteJob).

Table 629 - SMI Referenced Properties/Methods for CIM_AffectedJobElement (StoragePool to ConcreteJob)

Properties	Flags	Requirement	Description & Notes
AffectedElement		Mandatory	A reference to a CIM_StoragePool instance that is affected by the execution of the job.
AffectingElement		Mandatory	The job that is affecting the storage pool.

29.12.4 CIM_AffectedJobElement (StorageVolume to ConcreteJob)

Requirement: This is required if CIM_StorageConfigurationCapabilities.SupportedAsynchronousActions contains '17|19' (async volume operations).

Table 630 describes class CIM_AffectedJobElement (StorageVolume to ConcreteJob).

Table 630 - SMI Referenced Properties/Methods for CIM_AffectedJobElement (StorageVolume to ConcreteJob)

Properties	Flags	Requirement	Description & Notes
AffectedElement		Mandatory	A reference to a CIM_StorageVolume instance that is affected by the execution of the job.
AffectingElement		Mandatory	The job that is affecting the storage volume.

29.12.5 CIM_ElementCapabilities (StorageConfigurationCapabilities to StoragePool)

Requirement: Mandatory

Table 631 describes class CIM_ElementCapabilities (StorageConfigurationCapabilities to StoragePool).

Table 631 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (StorageConfigurationCapabilities to StoragePool)

Properties	Flags	Requirement	Description & Notes
Capabilities		Mandatory	The Pool StorageConfigurationCapabilities instance associated with the StoragePool.
ManagedElement		Mandatory	The StoragePool reference.

29.12.6 CIM_ElementCapabilities (StorageConfigurationCapabilities to StorageRelocationService)

Requirement: Mandatory

Table 632 describes class CIM_ElementCapabilities (StorageConfigurationCapabilities to StorageRelocationService).

Table 632 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (StorageConfigurationCapabilities to StorageRelocationService)

Properties	Flags	Requirement	Description & Notes
Capabilities		Mandatory	The global StorageConfigurationCapabilities associated with the element.
ManagedElement		Mandatory	The StorageRelocationService.

29.12.7 CIM_FilterCollection (Storage Relocation Predefined FilterCollection)

Experimental. This is a collection of predefined IndicationFilters to which a client may subscribe. A Storage Relocation 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 633 describes class CIM_FilterCollection (Storage Relocation Predefined FilterCollection).

Table 633 - SMI Referenced Properties/Methods for CIM_FilterCollection (Storage Relocation Predefined FilterCollection)

Properties	Flags	Requirement	Description & 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 Relocation:Predefined'.

29.12.8 CIM_FilterCollection (Storage Relocation ProfileSpecificAlertIndicationFilterCollection)

Experimental. This is a collection of Lifecycle IndicationFilters defined by the Storage Relocation Profile.

Requirement: Optional

Table 634 describes class CIM_FilterCollection (Storage Relocation ProfileSpecificAlertIndicationFilterCollection).

Table 634 - SMI Referenced Properties/Methods for CIM_FilterCollection (Storage Relocation ProfileSpecificAlertIndicationFilterCollection)

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	See the InstanceID definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.13 CIM_FilterCollection (StaticFilterCollection).
CollectionName		Mandatory	The value of CollectionName shall be 'SNIA:Storage Relocation:ProfileSpecificAlertIndicationFilterCollection'.

29.12.9 CIM_HostedCollection (System to Storage Relocation ProfileSpecificAlertIndicationFilterCollection)

Experimental.

Requirement: Optional

Table 635 describes class CIM_HostedCollection (System to Storage Relocation ProfileSpecificAlertIndicationFilterCollection).

Table 635 - SMI Referenced Properties/Methods for CIM_HostedCollection (System to Storage Relocation ProfileSpecificAlertIndicationFilterCollection)

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	Reference to the ProfileSpecificAlertIndicationFilterCollection for the Storage Relocation Profile.
Antecedent		Mandatory	Reference to the 'Top level' System.

29.12.10CIM_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 636 describes class CIM_HostedCollection (System to predefined IndicationFilters).

Table 636 - SMI Referenced Properties/Methods for CIM_HostedCollection (System to predefined IndicationFilters)

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	Reference to the predefined FilterCollection for the Storage Relocation implementation.
Antecedent		Mandatory	Reference to the System of the referencing profile.

29.12.11CIM_HostedService (StorageRelocationService to ComputerSystem)

Requirement: Mandatory

Table 637 describes class CIM_HostedService (StorageRelocationService to ComputerSystem).

Table 637 - SMI Referenced Properties/Methods for CIM_HostedService (StorageRelocationService to ComputerSystem)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	The hosting System.
Dependent		Mandatory	The StorageRelocationService hosted on the System.

29.12.12CIM_IndicationFilter (LogicalDiskRelocationEnd)

Experimental. This is the 'pre-defined' CIM_IndicationFilter instance for the end of a storage relocation of a CIM_LogicalDisk instance. This would typically occur sometime after getting the alert for the start of the relocation.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 638 describes class CIM_IndicationFilter (LogicalDiskRelocationEnd).

Table 638 - SMI Referenced Properties/Methods for CIM_IndicationFilter (LogicalDiskRelocationEnd)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Name		Mandatory	This shall be 'SNIA:Storage Relocation:LogicalDiskRelocationEnd'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).

Table 638 - SMI Referenced Properties/Methods for CIM_IndicationFilter (LogicalDiskRelocationEnd)

Properties	Flags	Requirement	Description & Notes
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Query		Mandatory	SELECT * FROM CIM_AlertIndication WHERE OwningEntity='SNIA' and MessageID='DRM37'.
QueryLanguage		Mandatory	This should be 'DMTF:CQL'.
ElementName	N	Optional	See the ElementName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).

29.12.13CIM_IndicationFilter (LogicalDiskRelocationStart)

Experimental. This is the 'pre-defined' CIM_IndicationFilter instance for the start of a storage relocation of a CIM_LogicalDisk instance. This would typically occur as a result of an invocation of Storage Relocation Service method.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 639 describes class CIM_IndicationFilter (LogicalDiskRelocationStart).

Table 639 - SMI Referenced Properties/Methods for CIM_IndicationFilter (LogicalDiskRelocationStart)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Name		Mandatory	This shall be 'SNIA:Storage Relocation:LogicalDiskRelocationStart'.

Table 639 - SMI Referenced Properties/Methods for CIM_IndicationFilter (LogicalDiskRelocation-Start)

Properties	Flags	Requirement	Description & Notes
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Query		Mandatory	SELECT * FROM CIM_AlertIndication WHERE OwningEntity='SNIA' and MessageID='DRM36'.
QueryLanguage		Mandatory	This should be 'DMTF:CQL'.
ElementName	N	Optional	See the ElementName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).

29.12.14CIM_IndicationFilter (StoragePoolRelocationEnd)

Experimental. This is the 'pre-defined' CIM_IndicationFilter instance for the end of a storage relocation of a CIM_StoragePool instance. This would typically occur sometime after getting the alert for the start of the relocation.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 640 describes class CIM_IndicationFilter (StoragePoolRelocationEnd).

Table 640 - SMI Referenced Properties/Methods for CIM_IndicationFilter (StoragePoolRelocationEnd)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).

Table 640 - SMI Referenced Properties/Methods for CIM_IndicationFilter (StoragePoolRelocationEnd)

Properties	Flags	Requirement	Description & Notes
Name		Mandatory	This shall be 'SNIA:Storage Relocation:StoragePoolRelocationEnd'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Query		Mandatory	SELECT * FROM CIM_AlertIndication WHERE OwningEntity='SNIA' and MessageID='DRM35'.
QueryLanguage		Mandatory	This should be 'DMTF:CQL'.
ElementName	N	Optional	See the ElementName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).

29.12.15CIM_IndicationFilter (StoragePoolRelocationStart)

Experimental. This is the 'pre-defined' CIM_IndicationFilter instance for the start of a storage relocation of a CIM_StoragePool instance. This would typically occur as a result of an invocation of Storage Relocation Service method.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 641 describes class CIM_IndicationFilter (StoragePoolRelocationStart).

Table 641 - SMI Referenced Properties/Methods for CIM_IndicationFilter (StoragePoolRelocationStart)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).

Table 641 - SMI Referenced Properties/Methods for CIM_IndicationFilter (StoragePoolRelocation-Start)

Properties	Flags	Requirement	Description & Notes
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Name		Mandatory	This shall be 'SNIA:Storage Relocation:StoragePoolRelocationStart'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Query		Mandatory	SELECT * FROM CIM_AlertIndication WHERE OwningEntity='SNIA' and MessageID='DRM34'.
QueryLanguage		Mandatory	This should be 'DMTF:CQL'.
ElementName	N	Optional	See the ElementName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).

29.12.16CIM_IndicationFilter (VolumeRelocationEnd)

Experimental. This is the 'pre-defined' CIM_IndicationFilter instance for the end of a storage relocation of a CIM_StorageVolume instance. This would typically occur sometime after getting the alert for the start of the relocation.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 642 describes class CIM_IndicationFilter (VolumeRelocationEnd).

Table 642 - SMI Referenced Properties/Methods for CIM_IndicationFilter (VolumeRelocationEnd)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Name		Mandatory	This shall be 'SNIA:Storage Relocation:VolumeRelocationEnd'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Query		Mandatory	SELECT * FROM CIM_AlertIndication WHERE OwningEntity='SNIA' and MessageID='DRM33'.
QueryLanguage		Mandatory	This should be 'DMTF:CQL'.
ElementName	N	Optional	See the ElementName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).

29.12.17CIM_IndicationFilter (VolumeRelocationStart)

Experimental. This is the 'pre-defined' CIM_IndicationFilter instance for the start of a storage relocation of a CIM_StorageVolume instance. This would typically occur as a result of an invocation of Storage Relocation Service method.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Optional

Table 643 describes class CIM_IndicationFilter (VolumeRelocationStart).

Table 643 - SMI Referenced Properties/Methods for CIM_IndicationFilter (VolumeRelocationStart)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Name		Mandatory	This shall be 'SNIA:Storage Relocation:VolumeRelocationStart'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Query		Mandatory	SELECT * FROM CIM_AlertIndication WHERE OwningEntity='SNIA' and MessageID='DRM32'.
QueryLanguage		Mandatory	This should be 'DMTF:CQL'.
ElementName	N	Optional	See the ElementName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).

29.12.18CIM_LogicalDisk

The CIM_LogicalDisk is an augmented version of the CIM_LogicalDisk defined in the Block Services package. When Storage Relocation is implemented this class adds a property and a property value for storage relocation.

Created By: Static

Modified By: Static

Deleted By: Extrinsic: StorageConfigurationService.ReturnToStoragePool

Requirement: Referenced from Volume Management - LogicalDisk is mandatory.

Table 644 describes class CIM_LogicalDisk.

Table 644 - SMI Referenced Properties/Methods for CIM_LogicalDisk

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.32 CIM_LogicalDisk.
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.32 CIM_LogicalDisk.
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.32 CIM_LogicalDisk.
DeviceID		Mandatory	See the DeviceID definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.32 CIM_LogicalDisk.
ElementName		Optional	See the ElementName definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.32 CIM_LogicalDisk.
Name		Mandatory	See the Name definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.32 CIM_LogicalDisk.
NameFormat		Mandatory	See the NameFormat definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.32 CIM_LogicalDisk.
ExtentStatus		Mandatory	See the ExtentStatus definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.32 CIM_LogicalDisk.
OperationalStatus		Mandatory	Value shall be 2|3|6|8|15|19 (OK or Degraded or Error or Starting or Dormant or Relocating). The Storage Relocation Profile adds the 19 enumeration.
BlockSize		Mandatory	See the BlockSize definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.32 CIM_LogicalDisk.
NumberOfBlocks		Mandatory	See the NumberOfBlocks definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.32 CIM_LogicalDisk.
ConsumableBlocks		Mandatory	See the ConsumableBlocks definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.32 CIM_LogicalDisk.
IsBasedOnUnderlyingRedundancy		Mandatory	See the IsBasedOnUnderlyingRedundancy definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.32 CIM_LogicalDisk.

Table 644 - SMI Referenced Properties/Methods for CIM_LogicalDisk

Properties	Flags	Requirement	Description & Notes
NoSinglePointOfFailure		Mandatory	See the NoSinglePointOfFailure definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.32 CIM_LogicalDisk.
DataRedundancy		Mandatory	See the DataRedundancy definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.32 CIM_LogicalDisk.
PackageRedundancy		Mandatory	See the PackageRedundancy definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.32 CIM_LogicalDisk.
DeltaReservation		Mandatory	See the DeltaReservation definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.32 CIM_LogicalDisk.
Usage		Optional	See the Usage definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.32 CIM_LogicalDisk.
OtherUsageDescription		Optional	See the OtherUsageDescription definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.32 CIM_LogicalDisk.
ClientSettableUsage		Optional	See the ClientSettableUsage definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.32 CIM_LogicalDisk.
Primordial		Mandatory	See the Primordial definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.32 CIM_LogicalDisk.
ExtentDiscriminator		Mandatory	Experimental. See the ExtentDiscriminator definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.32 CIM_LogicalDisk.
NumExtentsMigrating		Optional	The number of Extents in the process of migrating for this logical disk when the logical disk relocation is on going. The Storage Relocation Profile adds this property.

29.12.19CIM_MemberOfCollection (Predefined Filter Collection to Storage Relocation Filters)

Experimental. This associates the Storage Relocation 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 645 describes class CIM_MemberOfCollection (Predefined Filter Collection to Storage Relocation Filters).

Table 645 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Predefined Filter Collection to Storage Relocation Filters)

Properties	Flags	Requirement	Description & Notes
Collection		Mandatory	Reference to the Storage Relocation predefined FilterCollection.
Member		Mandatory	Reference to the predefined IndicationFilters of the Storage Relocation implementation.

29.12.20CIM_MemberOfCollection (Storage Relocation Filter Collection to FilterCollection)

Experimental. This associates the Storage Relocation 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 646 describes class CIM_MemberOfCollection (Storage Relocation Filter Collection to FilterCollection).

Table 646 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Storage Relocation Filter Collection to FilterCollection)

Properties	Flags	Requirement	Description & Notes
Collection		Mandatory	Reference to the predefined FilterCollection of the referencing profile (e.g., Array FilterCollection).
Member		Mandatory	Reference to the Storage Relocation predefined FilterCollection.

29.12.21CIM_MemberOfCollection (Storage Relocation ProfileSpecificAlertIndicationFilterCollection to Storage Relocation Filters)

Experimental. This associates the Storage Relocation ProfileSpecificAlertIndicationFilterCollection to the Filters defined by the Storage Relocation Profile.

Requirement: Optional

Table 647 describes class CIM_MemberOfCollection (Storage Relocation ProfileSpecificAlertIndicationFilterCollection to Storage Relocation Filters).

Table 647 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Storage Relocation ProfileSpecificAlertIndicationFilterCollection to Storage Relocation Filters)

Properties	Flags	Requirement	Description & Notes
Collection		Mandatory	Reference to the Storage Relocation ProfileSpecificAlertIndicationFilterCollection.
Member		Mandatory	Reference to the IndicationFilters defined by the Storage Relocation Profile.

29.12.22CIM_OwningJobElement (StorageRelocationService to ConcreteJob)

Requirement: This is required if CIM_StorageConfigurationCapabilities.SupportedAsynchronousActions contains '17|18|19|20|21' (async relocation operations).

Table 648 describes class CIM_OwningJobElement (StorageRelocationService to ConcreteJob).

Table 648 - SMI Referenced Properties/Methods for CIM_OwningJobElement (StorageRelocation-Service to ConcreteJob)

Properties	Flags	Requirement	Description & Notes
OwningElement		Mandatory	A reference to a CIM_StorageRelocationService instance responsible for the creation of the job.
OwnedElement		Mandatory	The job created by the CIM_StorageRelocationService.

29.12.23CIM_StorageConfigurationCapabilities (Concrete)

The global CIM_StorageConfigurationCapabilities an augmented version of the global CIM_StorageConfigurationCapabilities defined in the Block Services package.When Storage Relocation is implemented this class is Mandatory and adds property values for storage relocation.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 649 describes class CIM_StorageConfigurationCapabilities (Concrete).

Table 649 - SMI Referenced Properties/Methods for CIM_StorageConfigurationCapabilities (Con-crete)

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	See the InstanceID definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.38 CIM_StorageConfigurationCapabilities (Concrete).
ElementName		Mandatory	See the ElementName definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.38 CIM_StorageConfigurationCapabilities (Concrete).
SupportedStoragePoolFeatures		Optional	See the SupportedStoragePoolFeatures definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.38 CIM_StorageConfigurationCapabilities (Concrete).

Table 649 - SMI Referenced Properties/Methods for CIM_StorageConfigurationCapabilities (Concrete)

Properties	Flags	Requirement	Description & Notes
SupportedAsynchronousActions		Mandatory	Lists what actions, invoked through StorageConfigurationService methods, shall not produce Concrete jobs. This profile augments the list in Block Services by making this property mandatory and adding actions, invoked through StorageRelocationService methods, that may produce Concrete jobs. 17 = 'StorageVolume To StoragePool Relocation' 18 = 'StoragePool To StoragePool Relocation' 19 = 'StorageVolume To StorageExtent Relocation' 20 = 'StoragePool To StorageExtent Relocation' 21 = 'LogicalDisk To StorageExtent Relocation'.
SupportedStorageElementTypes		Mandatory	See the SupportedStorageElementTypes definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.38 CIM_StorageConfigurationCapabilities (Concrete).
SupportedSynchronousActions		Mandatory	Lists what actions, invoked through StorageConfigurationService methods, may produce Concrete jobs. This profile augments the list in Block Services by making this property mandatory and adding actions, invoked through StorageRelocationService methods, that shall not produce Concrete jobs. 17 = 'StorageVolume To StoragePool Relocation' 18 = 'StoragePool To StoragePool Relocation' 19 = 'StorageVolume To StorageExtent Relocation' 20 = 'StoragePool To StorageExtent Relocation' 21 = 'LogicalDisk To StorageExtent Relocation'.

Table 649 - SMI Referenced Properties/Methods for CIM_StorageConfigurationCapabilities (Concrete)

Properties	Flags	Requirement	Description & Notes
SupportedStorageElementFeatures		Mandatory	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). This profile augments the list in Block Services by making this property mandatory and adding actions supported through the invocation of StorageRelocationService.RelocateStorageElementsToStoragePool() and StorageRelocationService.RelocateStorageElementToElements(). 14 = 'StorageVolume To StoragePool Relocation' 15 = 'StoragePool To StoragePool Relocation' 16 = 'StorageVolume To StorageExtent Relocation' 17 = 'StoragePool To StorageExtent Relocation' 18 = 'LogicalDisk To StorageExtent Relocation'.
SupportedStorageElementUsage		Optional	See the SupportedStorageElementUsage definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.38 CIM_StorageConfigurationCapabilities (Concrete).
ClientSettableElementUsage		Optional	See the ClientSettableElementUsage definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.38 CIM_StorageConfigurationCapabilities (Concrete).

Table 649 - SMI Referenced Properties/Methods for CIM_StorageConfigurationCapabilities (Concrete)

Properties	Flags	Requirement	Description & Notes
SupportedStoragePoolUsage		Mandatory	Indicates the intended usage or any restrictions that may have been imposed on storage pools. This profile augments the list in Block Services by making this property mandatory and adding the following values to the enumerations: 9='Used as source for Relocation Service' 10='Used as target for Relocation Service'.
ClientSettablePoolUsage		Mandatory	Indicates the intended usage or any restrictions that may have been imposed on the usage of a client-settable storage pool. This profile augments the list in Block Services by making this property mandatory and adding the following values to the enumerations: 9='Used as source for Relocation Service' 10='Used as target for Relocation Service'.

29.12.24CIM_StorageConfigurationCapabilities (Global)

The global CIM_StorageConfigurationCapabilities an augmented version of the global CIM_StorageConfigurationCapabilities defined in the Block Services package.When Storage Relocation is implemented this class is Mandatory and adds property values for storage relocation.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 650 describes class CIM_StorageConfigurationCapabilities (Global).

Table 650 - SMI Referenced Properties/Methods for CIM_StorageConfigurationCapabilities (Global)

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	See the InstanceID definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.39 CIM_StorageConfigurationCapabilities (Global).
ElementName		Mandatory	See the ElementName definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.39 CIM_StorageConfigurationCapabilities (Global).

Table 650 - SMI Referenced Properties/Methods for CIM_StorageConfigurationCapabilities (Global)

Properties	Flags	Requirement	Description & Notes
SupportedStoragePoolFeatures		Optional	See the SupportedStoragePoolFeatures definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.39 CIM_StorageConfigurationCapabilities (Global).
SupportedAsynchronousActions		Mandatory	Lists what actions, invoked through StorageConfigurationService methods, shall not produce Concrete jobs. This profile augments the list in Block Services by making this property mandatory and adding actions, invoked through StorageRelocationService methods, that may produce Concrete jobs. 17 = 'StorageVolume To StoragePool Relocation' 18 = 'StoragePool To StoragePool Relocation' 19 = 'StorageVolume To StorageExtent Relocation' 20 = 'StoragePool To StorageExtent Relocation' 21 = 'LogicalDisk To StorageExtent Relocation'.
SupportedStorageElementTypes		Mandatory	See the SupportedStorageElementTypes definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.39 CIM_StorageConfigurationCapabilities (Global).
SupportedSynchronousActions		Mandatory	Lists what actions, invoked through StorageConfigurationService methods, may produce Concrete jobs. This profile augments the list in Block Services by making this property mandatory and adding actions, invoked through StorageRelocationService methods, that shall not produce Concrete jobs. 17 = 'StorageVolume To StoragePool Relocation' 18 = 'StoragePool To StoragePool Relocation' 19 = 'StorageVolume To StorageExtent Relocation' 20 = 'StoragePool To StorageExtent Relocation' 21 = 'LogicalDisk To StorageExtent Relocation'.

Table 650 - SMI Referenced Properties/Methods for CIM_StorageConfigurationCapabilities (Global)

Properties	Flags	Requirement	Description & Notes
SupportedStorageElementFeatures		Mandatory	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). This profile augments the list in Block Services by making this property mandatory and adding actions supported through the invocation of StorageRelocationService.RelocateStorageElementsToStoragePool() and StorageRelocationService.RelocateStorageElementToElements(). 14 = 'StorageVolume To StoragePool Relocation' 15 = 'StoragePool To StoragePool Relocation' 16 = 'StorageVolume To StorageExtent Relocation' 17 = 'StoragePool To StorageExtent Relocation' 18 = 'LogicalDisk To StorageExtent Relocation'.
SupportedStorageElementUsage		Optional	See the SupportedStorageElementUsage definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.39 CIM_StorageConfigurationCapabilities (Global).
ClientSettableElementUsage		Optional	See the ClientSettableElementUsage definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.39 CIM_StorageConfigurationCapabilities (Global).

Table 650 - SMI Referenced Properties/Methods for CIM_StorageConfigurationCapabilities (Global)

Properties	Flags	Requirement	Description & Notes
SupportedStoragePoolUsage		Mandatory	Indicates the intended usage or any restrictions that may have been imposed on storage pools. This profile augments the list in Block Services by making this property mandatory and adding the following values to the enumerations: 9='Used as source for Relocation Service' 10='Used as target for Relocation Service'.
ClientSettablePoolUsage		Mandatory	Indicates the intended usage or any restrictions that may have been imposed on the usage of a client-settable storage pool. This profile augments the list in Block Services by making this property mandatory and adding the following values to the enumerations: 9='Used as source for Relocation Service' 10='Used as target for Relocation Service'.

29.12.25CIM_StorageConfigurationCapabilities (Primordial)

The global CIM_StorageConfigurationCapabilities an augmented version of the global CIM_StorageConfigurationCapabilities defined in the Block Services package.When Storage Relocation is implemented this class is Mandatory and adds property values for storage relocation.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 651 describes class CIM_StorageConfigurationCapabilities (Primordial).

Table 651 - SMI Referenced Properties/Methods for CIM_StorageConfigurationCapabilities (Primordial)

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	See the InstanceID definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.40 CIM_StorageConfigurationCapabilities (Primordial).
ElementName		Mandatory	See the ElementName definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.40 CIM_StorageConfigurationCapabilities (Primordial).

Table 651 - SMI Referenced Properties/Methods for CIM_StorageConfigurationCapabilities (Primordial)

Properties	Flags	Requirement	Description & Notes
SupportedStoragePoolFeatures		Optional	See the SupportedStoragePoolFeatures definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.40 CIM_StorageConfigurationCapabilities (Primordial).
SupportedAsynchronousActions		Mandatory	Lists what actions, invoked through StorageConfigurationService methods, shall not produce Concrete jobs. This profile augments the list in Block Services by making this property mandatory and adding actions, invoked through StorageRelocationService methods, that may produce Concrete jobs. 17 = 'StorageVolume To StoragePool Relocation' 18 = 'StoragePool To StoragePool Relocation' 19 = 'StorageVolume To StorageExtent Relocation' 20 = 'StoragePool To StorageExtent Relocation' 21 = 'LogicalDisk To StorageExtent Relocation'.
SupportedStorageElementTypes		Mandatory	See the SupportedStorageElementTypes definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.40 CIM_StorageConfigurationCapabilities (Primordial).
SupportedSynchronousActions		Mandatory	Lists what actions, invoked through StorageConfigurationService methods, may produce Concrete jobs. This profile augments the list in Block Services by making this property mandatory and adding actions, invoked through StorageRelocationService methods, that shall not produce Concrete jobs. 17 = 'StorageVolume To StoragePool Relocation' 18 = 'StoragePool To StoragePool Relocation' 19 = 'StorageVolume To StorageExtent Relocation' 20 = 'StoragePool To StorageExtent Relocation' 21 = 'LogicalDisk To StorageExtent Relocation'.

Table 651 - SMI Referenced Properties/Methods for CIM_StorageConfigurationCapabilities (Primordial)

Properties	Flags	Requirement	Description & Notes
SupportedStorageElementFeatures		Mandatory	Lists actions supported through the invocation of StorageServiceService.CreateOrModifyElementFromStoragePool(). This profile augments the list in Block Services by making this property mandatory and adding actions supported through the invocation of StorageRelocationService.RelocateStorageElementsToStoragePool() and StorageRelocationService.RelocateStorageElementToElements(). 14 = 'StorageVolume To StoragePool Relocation' 15 = 'StoragePool To StoragePool Relocation' 16 = 'StorageVolume To StorageExtent Relocation' 17 = 'StoragePool To StorageExtent Relocation' 18 = 'LogicalDisk To StorageExtent Relocation'.
SupportedStorageElementUsage		Optional	See the SupportedStorageElementUsage definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.40 CIM_StorageConfigurationCapabilities (Primordial).
ClientSettableElementUsage		Optional	See the ClientSettableElementUsage definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.40 CIM_StorageConfigurationCapabilities (Primordial).
SupportedStoragePoolUsage		Mandatory	Indicates the intended usage or any restrictions that may have been imposed on storage pools. This profile augments the list in Block Services by making this property mandatory and adding the following values to the enumerations: 9='Used as source for Relocation Service' 10='Used as target for Relocation Service'.
ClientSettablePoolUsage		Mandatory	Indicates the intended usage or any restrictions that may have been imposed on the usage of a client-settable storage pool. This profile augments the list in Block Services by making this property mandatory and adding the following values to the enumerations: 9='Used as source for Relocation Service' 10='Used as target for Relocation Service'.

29.12.26CIM_StorageExtent (Relocatable)

A StorageExtent that can be a source or target of a relocation operation.

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

Table 652 describes class CIM_StorageExtent (Relocatable).

Table 652 - SMI Referenced Properties/Methods for CIM_StorageExtent (Relocatable)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	
ExtentStatus		Mandatory	A value 18 means the extent is actively involved in a relocating operation.
NumberOfBlocks		Mandatory	
ConsumableBlocks		Mandatory	The number of usable blocks.
BlockSize		Mandatory	
Primordial		Mandatory	This shall be 'false' for extents that are components of Concrete StoragePools and 'true' for extents that are components of Primordial StoragePools.
ExtentDiscriminator		Mandatory	Experimental. This is array of values that shall contain 'SNIA:Pool Component'.

29.12.27CIM_StoragePool (Concrete)

The concrete CIM_StoragePool an augmented version of the concrete CIM_StoragePool defined in the Block Services package. When Storage Relocation is implemented this class adds properties for storage relocation.

Created By: Extrinsic: StorageConfigurationService.CreateOrModifyStoragePool
Modified By: Extrinsic: StorageConfigurationService.CreateOrModifyStoragePool
Deleted By: Extrinsic: StorageConfigurationService.DeleteStoragePool
Requirement: Mandatory

Table 653 describes class CIM_StoragePool (Concrete).

Table 653 - SMI Referenced Properties/Methods for CIM_StoragePool (Concrete)

Properties	Flags	Requirement	Description & Notes
Primordial		Mandatory	See the Primordial definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.42 CIM_StoragePool (Concrete)Shall be false.
InstanceID		Mandatory	See the InstanceID definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.42 CIM_StoragePool (Concrete).
ElementName		Optional	See the ElementName definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.42 CIM_StoragePool (Concrete).
PoolID		Mandatory	See the PoolID definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.42 CIM_StoragePool (Concrete).
TotalManagedSpace		Mandatory	See the TotalManagedSpace definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.42 CIM_StoragePool (Concrete).
RemainingManaged Space		Mandatory	See the RemainingManagedSpace definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.42 CIM_StoragePool (Concrete).
Usage		Optional	See the Usage definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.42 CIM_StoragePool (Concrete).
OtherUsageDescripti on		Optional	See the OtherUsageDescription definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.42 CIM_StoragePool (Concrete).
ClientSettableUsage		Optional	See the ClientSettableUsage definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.42 CIM_StoragePool (Concrete).
OperationalStatus		Mandatory	The Storage Relocation Profile adds the requirement for this property. The values shall be either '2' (OK) or '19' (Relocating).
CapacityInMigratingS ource		Optional	The Storage Relocation Profile adds this property.The total capacity of extents migrating out from this storage pool.
CapacityInMigratingT arget		Optional	The Storage Relocation Profile adds this property.The total capacity of extents migrating into this storage pool.
GetSupportedSizes()		Conditional	Conditional requirement: Support for StorageConfigurationService. See the GetSupportedSizes definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.42 CIM_StoragePool (Concrete).

Table 653 - SMI Referenced Properties/Methods for CIM_StoragePool (Concrete)

Properties	Flags	Requirement	Description & Notes
GetSupportedSizeRange()		Conditional	Conditional requirement: Support for StorageConfigurationService. See the GetSupportedSizeRange definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.42 CIM_StoragePool (Concrete).
GetAvailableExtents()		Optional	See the GetAvailableExtents definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.42 CIM_StoragePool (Concrete).

29.12.28CIM_StoragePool (Primordial)

The primordial CIM_StoragePool an augmented version of the primordial CIM_StoragePool defined in the Block Services package. When Storage Relocation is implemented this class adds properties for storage relocation.

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 654 describes class CIM_StoragePool (Primordial).

Table 654 - SMI Referenced Properties/Methods for CIM_StoragePool (Primordial)

Properties	Flags	Requirement	Description & Notes
Primordial		Mandatory	See the Primordial definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.44 CIM_StoragePool (Primordial)Shall be true.
InstanceID		Mandatory	See the InstanceID definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.44 CIM_StoragePool (Primordial).
ElementName		Optional	See the ElementName definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.44 CIM_StoragePool (Primordial).
PoolID		Mandatory	See the PoolID definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.44 CIM_StoragePool (Primordial).
TotalManagedSpace		Mandatory	See the TotalManagedSpace definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.44 CIM_StoragePool (Primordial).
RemainingManagedSpace		Mandatory	See the RemainingManagedSpace definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.44 CIM_StoragePool (Primordial).

Table 654 - SMI Referenced Properties/Methods for CIM_StoragePool (Primordial)

Properties	Flags	Requirement	Description & Notes
Usage		Optional	See the Usage definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.44 CIM_StoragePool (Primordial).
OtherUsageDescription		Optional	See the OtherUsageDescription definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.44 CIM_StoragePool (Primordial).
ClientSettableUsage		Optional	See the ClientSettableUsage definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.44 CIM_StoragePool (Primordial).
OperationalStatus		Mandatory	The Storage Relocation Profile adds the requirement for this property. The values shall be either '2' (OK) or '19' (Relocating).
CapacityInMigratingSource		Optional	The Storage Relocation Profile adds this property.The total capacity of extents migrating out from this storage pool.
CapacityInMigratingTarget		Optional	The Storage Relocation Profile adds this property.The total capacity of extents migrating into this storage pool.
GetSupportedSizes()		Conditional	Conditional requirement: Support for StorageConfigurationService. See the GetSupportedSizes definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.44 CIM_StoragePool (Primordial).
GetSupportedSizeRange()		Conditional	Conditional requirement: Support for StorageConfigurationService. See the GetSupportedSizeRange definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.44 CIM_StoragePool (Primordial).
GetAvailableExtents()		Optional	See the GetAvailableExtents definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.44 CIM_StoragePool (Primordial).

29.12.29CIM_StorageRelocationService

Experimental.

Requirement: Mandatory

Table 655 describes class CIM_StorageRelocationService.

Table 655 - SMI Referenced Properties/Methods for CIM_StorageRelocationService

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
CreationClassName		Mandatory	

Table 655 - SMI Referenced Properties/Methods for CIM_StorageRelocationService

Properties	Flags	Requirement	Description & Notes
SystemName		Mandatory	
Name		Mandatory	
RelocateStorageVolumesToStoragePool()		Optional	Relocate storage volumes to specified target storage pool.
RelocateStoragePoolsToStoragePool()		Optional	Relocate storage pools to specified target storage pool.
RelocateStorageVolumeToStorageExtents()		Optional	Relocate a storage volume to specified storage extents.
RelocateStoragePoolToStorageExtents()		Optional	Relocate a storage pool to specified storage extents.
RelocateLogicalDiskToStorageExtents()		Optional	Relocate a logical disk to specified storage extents.
GetAvailableTargetRelocationExtents()		Optional	Get available target storage extents as new group for relocation.

29.12.30CIM_StorageVolume

The CIM_StorageVolume is an augmented version of the CIM_StorageVolume defined in the Block Services package. When Storage Relocation is implemented this class adds a property and a property value for storage relocation.

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 656 describes class CIM_StorageVolume.

Table 656 - SMI Referenced Properties/Methods for CIM_StorageVolume

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.49 CIM_StorageVolume.
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.49 CIM_StorageVolume.
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.49 CIM_StorageVolume.

Table 656 - SMI Referenced Properties/Methods for CIM_StorageVolume

Properties	Flags	Requirement	Description & Notes
DeviceID		Mandatory	See the DeviceID definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.49 CIM_StorageVolume.
ElementName		Optional	See the ElementName definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.49 CIM_StorageVolume.
Name	CD	Mandatory	See the Name definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.49 CIM_StorageVolume.
OtherIdentifyingInfo	CD	Optional	See the OtherIdentifyingInfo definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.49 CIM_StorageVolume.
IdentifyingDescriptions		Optional	See the IdentifyingDescriptions definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.49 CIM_StorageVolume.
NameFormat		Mandatory	See the NameFormat definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.49 CIM_StorageVolume.
NameNamespace		Mandatory	See the NameNamespace definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.49 CIM_StorageVolume.
ExtentStatus		Mandatory	See the ExtentStatus definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.49 CIM_StorageVolume.
OperationalStatus		Mandatory	Value shall be 2|3|6|8|15|19 (OK or Degraded or Error or Starting or Dormant or Relocating).The Storage Relocation Profile adds the enumeration for relocating.
NumExtentsMigrating		Optional	The Storage Relocation Profile adds this property. The number of Extents in the process of migrating for this storage volume when the volume relocation is on going.
BlockSize		Mandatory	See the BlockSize definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.49 CIM_StorageVolume.
NumberOfBlocks		Mandatory	See the NumberOfBlocks definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.49 CIM_StorageVolume.
ConsumableBlocks		Mandatory	See the ConsumableBlocks definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.49 CIM_StorageVolume.
IsBasedOnUnderlyingRedundancy		Mandatory	See the IsBasedOnUnderlyingRedundancy definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.49 CIM_StorageVolume.

Table 656 - SMI Referenced Properties/Methods for CIM_StorageVolume

Properties	Flags	Requirement	Description & Notes
NoSinglePointOfFailure		Mandatory	See the NoSinglePointOfFailure definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.49 CIM_StorageVolume.
DataRedundancy		Mandatory	See the DataRedundancy definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.49 CIM_StorageVolume.
PackageRedundancy		Mandatory	See the PackageRedundancy definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.49 CIM_StorageVolume.
DeltaReservation		Mandatory	See the DeltaReservation definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.49 CIM_StorageVolume.
Usage		Optional	See the Usage definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.49 CIM_StorageVolume.
OtherUsageDescription		Optional	See the OtherUsageDescription definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.49 CIM_StorageVolume.
ClientSettableUsage		Optional	See the ClientSettableUsage definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.49 CIM_StorageVolume.
Primordial		Mandatory	See the Primordial definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.49 CIM_StorageVolume.
ExtentDiscriminator		Mandatory	Experimental. See the ExtentDiscriminator definition in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.49 CIM_StorageVolume.

EXPERIMENTAL

EXPERIMENTAL

Clause 30: Thin Provisioning Profile

30.1 Description

30.1.1 Overview

The Block Services with Thin Provisioning Profile is a specialization of the Block Services Package (see Clause 5: Block Services Package), adding support for thin provisioning. All the provisions of the Block Services Package apply, in addition to those defined in this profile.

This profile is nearly compatible with the Block Services Package. A client supporting the Block Services Package interacting with a Block Server with Thin Provisioning Profile agent should be able to actively manage fully-provisioned volumes and pools, but discovery will be slightly impacted because it will see instances of both fully and thinly provisioned pools and volumes. The model is the same, but the client needs to consider the values of thin-provisioning-specific properties to fully understand capacity utilization.

30.1.2 Background

Thin provisioning is a capability of some block server implementations to defer provisioning of backing store for regions of a volume until the regions have been accessed (written) by the consumer (e.g., host file system). The alternatives (fully provisioned volumes) allocate all of the requested capacity from the backing store at the time the volume is created. For thin provisioned volumes, the block server implementation tracks information about which regions have been accessed, and once a region is accessed, the backing storage is allocated.

There are various approaches to implementing thin provisioning; some vendors pattern thin provisioning logic after OS virtual memory or journaled file systems, and there are numerous variations. This profile does not address techniques or algorithms for thin provisioning; these details are left to innovation of the vendors delivering thin provisioning solutions. This profile provides a common abstraction for the management features of thin provisioning. In particular, this profile allows SMI-S clients to determine whether a storage system (and children such as pools, and volumes) supports thin provisioning, determine the difference between the exposed “virtual capacity” and actual, committed physical storage, and create thinly provisioned volumes and pools.

30.1.3 Model

No new classes are defined by this profile; it extends the classes of Block Services.

Throughout this profile, ***volume*** refers to either StorageVolume or LogicalDisk, which are the two types of elements exported from the Block Services Profile. ***Pool children*** refers to the three types of elements (StorageVolume, LogicalDisk, and StoragePool) that may be carved from a pool.

30.1.3.1 Capacity Concepts for Volumes

Each storage volume has a ***nominal capacity*** value, the capacity seen by users and applications (such as file systems). This capacity is also reported through in-band interfaces such as SCSI READ CAPACITY. Applications cannot write more than this capacity at a given time. When fully provisioned volumes are created, the nominal capacity is allocated by the block server. When thin provisioned volumes are created, a smaller value (referred to here as the ***initial reserve capacity***) is allocated (this value may be zero).

Capacity consumed is the capacity the application is actually using at a give time (the block server may have rounded this up to a multiple of some internal granule size). For thin provisioned volumes, the capacity consumed on the backend storage may be smaller than the nominal capacity. The capacity consumed grows from the initial reserve capacity as the application (such as a file system) writes new areas of the volume. In theory, the capacity consumed could grow to equal (or exceed when metadata is considered) the nominal capacity.

The nominal capacity is represented in the model by the ConsumableBlocks property of volumes. Capacity consumed is modeled by the SpaceConsumed property of the AllocatedFromStoragePool association referencing the volume. Initial reserve capacity is modeled as a percentage of the nominal capacity using the ThinProvisionedInitialReserve property of StorageSettings. In some block servers, the smaller capacity is a characteristic of a StoragePool and is represented by the SpaceConsumed on the AllocatedFromStoragePool association between the StorageVolume or LogicalDisk and StoragePool.

Note that these concepts and properties also apply to delta replicas as defined in Clause 9: Copy Services Subprofile and Clause 26: Replication Services Profile.

30.1.3.2 Capacity Concepts for Pools

Block Servers supporting thin provisioned volumes have different approaches to modeling capacity in pools. This profile supports three approaches:

- The first approach is used when a pool supports thin provisioned children, but the “advertised” capacity of the pool matches the actual capacity of its underlying storage. In this case, the block server follows the provisions in Clause 5: Block Services Package.
- The second approach is used when a pool supports thin provisioned children and has a defined capacity to which its children can grow, but this capacity is greater than the capacity of underlying storage.
- The third approach is when the block server does not assign a maximum capacity to the pool.

The model supporting these three approaches is documented in 30.1.3.3.1 Pool Capacity.

Note that primordial StoragePools cannot be thinly provisioned, but can support allocation of thinly provisioned concrete pools.

30.1.3.3 Overview

Figure 157 presents the key classes and related to this profile.

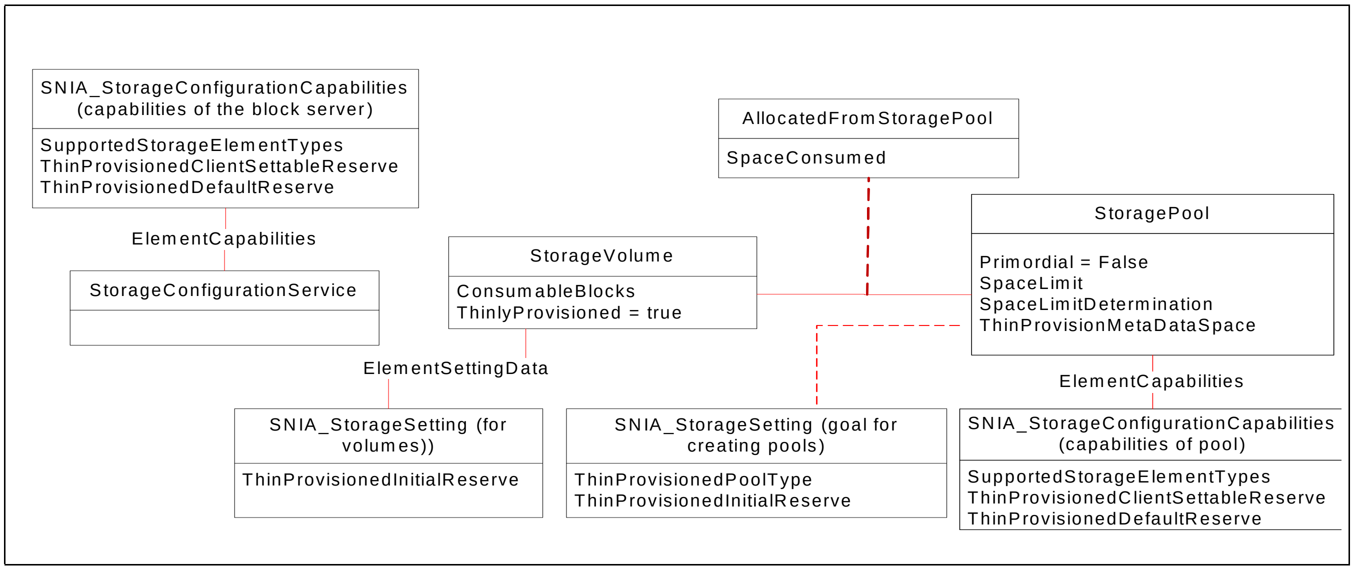
SNIA_StorageConfigurationCapabilities (capabilities of the block server)
SupportedStorageElementTypes
ThinProvisionedClientSettableReserve
ThinProvisionedDefaultReserve
ElementCapabilities
StorageConfigurationService
AllocatedFromStoragePool
SpaceConsumed
StorageVolume
ConsumableBlocks
ThinlyProvisioned = true
StoragePool
Primordial = False
SpaceLimit
SpaceLimitDetermination
ThinProvisionMetaDataSpace
ElementSettingData
ElementCapabilities
SNIA_StorageSetting (for volumes))
ThinProvisionedInitialReserve
SNIA_StorageSetting (goal for creating pools)
ThinProvisionedPoolType
ThinProvisionedInitialReserve
SNIA_StorageConfigurationCapabilities (capabilities of pool)
SupportedStorageElementTypes
ThinProvisionedClientSettableReserve
ThinProvisionedDefaultReserve

Figure 157 - Thin Provisioning Model

StorageConfigurationCapabilities.SupportedStorageElementTypes shall include a subset of ThinlyProvisionedStorageVolume, ThinlyProvisionedLogicalDisk, ThinlyProvisionedAllocatedStoragePool, ThinlyProvisionedQuotaStoragePool, or ThinlyProvisionedLimitlessStoragePool to indicate support for allocation of thinly provisioned StorageVolumes, LogicalDisks, or StoragePools. The three SupportedStorageElementTypes values related to pools allow the block server to advertise which types of pool capacity approaches are available for child pools. The meaning of Allocated, Quota and Limitless pools is expanded in 30.1.3.2 Capacity Concepts for Pools. Similar values are used in ElementType parameters of methods to specify which approach the client prefers when creating new children. Note that as defined in the Block Services Package, StorageConfigurationCapabilities associated to StorageConfigurationService defines global block server capabilities; other instances of StorageConfigurationCapabilities may optionally be associated to StoragePool to provide pool-specific overrides.

The SpaceLimitDetermination property of StoragePool defines the approach associated with the pool for determining capacity information for the pool. See 30.1.3.3.1 Pool Capacity. The SpaceLimitDetermination property is undefined if the Block Services with Thin Provisioning Profile is not supported. SpaceLimitDetermination shall be present on any StoragePool instance that supports thin provisioning and SpaceLimitDetermination is not Allocated.

The SpaceLimit property of StoragePool is the capacity of the storage allocated to the pool when Spacelimitdetermination has the value 3 (Quota) or 4 (Limitless) or is set to the value of TotalManagedSpace if SpaceLimitDetermination has the value 2 (Allocated). The value of SpaceLimit may be modified by a client using CreateOrModifyStoragePool. The upper bounds returned from GetAvailableSizes and GetAvailableSizeRanges should be approximately the same as SpaceLimit. See 30.1.3.3.1 Pool Capacity. The SpaceLimit property is not defined if the Block Services with Thin Provisioning Profile is not supported.

The ThinProvisionMetaDataSpace property of StoragePool is the size of the pool's metadata (in bytes). Unlike fully-provisioned pools, this value cannot be determined by subtracting the sum of SpaceConsumed of child elements from TotalManagedSpace. The ThinProvisionMetaDataSpace property is undefined if the Block Services with Thin Provisioning Profile is not supported.

If the ThinlyProvisioned property of StorageVolume or LogicalDisk is "true", then the block server shall support thin provisioning for the StorageVolume or LogicalDisk. If ThinlyProvisioned is undefined or the value is null, the StorageVolume or LogicalDisk shall not be thin provisioned. The ThinlyProvisioned property is undefined if the Block Services with Thin Provisioning Profile is not supported.

30.1.3.3.1 Pool Capacity

StoragePool.SpaceLimitDetermination indicates which of three approaches apply to determining the capacity related properties of the associated StoragePool.

In all cases, StoragePool.TotalManagedSpace represents the sum of the usable capacity from underlying StorageExtents. The StorageExtents may or may not be modeled and the usable capacity may have been reduced due to redundancy or metadata. In all cases, RemainingManagedSpace shall be set to SpaceLimit minus the sum of SpaceConsumed on AllocatedFromStoragePool associations to all pool children.

This profile supports three techniques for determining the space available for creating or expanding child elements.

- If StoragePool.SpaceLimitDetermination is set to 2 (Allocated), TotalManagedSpace is also the capacity that may be used to create or expand pool children (StorageVolumes, LogicalDisks, or other StoragePools). And StoragePool.RemamingManagedSpace represents the capacity left to create a new storage element or expand an existing storage element. This approach is common to fully provisioned pools. The SpaceLimit property should be set to the same value as TotalManagedSpace.
- If StoragePool.SpaceLimitDetermination is set to 3 (Quota), StoragePool.SpaceLimit serves as an administratively defined limit on the capacity that may be used to create or expand child elements (StorageVolumes, LogicalDisks, or other StoragePools).
- If StoragePool.SpaceLimitDetermination is set to 4 (Limitless), then the block server does not have a defined limit on the capacity for creating or expanding children. Clients that support thin provisioning should not use SpaceLimit when SpaceLimitDetermination is set to 4 (Limitless). But for compatibility with clients that do not support this profile, the instrumentation should use a heuristic to set SpaceLimit (and to values returned from GetAvailableSizes and GetAvailableSizeRanges) to a reasonable value. One possible heuristic is to set SpaceLimit to the value of the largest volume supported by the implementation (e.g., 2 terabytes if the implementation does not support SCSI sixteen byte CDBs).

- If StoragePool.SpaceLimitDetermination is set to 5 (Fully Provisioned), then the block server allocates child elements' nominal capacity when they are created.

If SpaceLimitDetermination is null or undefined, clients should treat the pools as if SpaceLimitDetermination was 5 (Fully Provisioned).

30.1.3.4 Indications

30.1.3.4.1 Capacity Warning

This is an alert message indicating that the actual capacity of a volume or pool is nearing a limit (e.g., actual usage of containing pool is nearing SpaceLimit). The related standard message is

```
Thin provisioned <Volume or Pool> with identifier <Volume or Pool ID> capacity in
                        use nearing available limit.
```

30.1.3.4.2 Capacity Critical

This is an alert message indicating that the actual capacity of a volume or pool has reached a limit (e.g., actual usage of containing pool is equal to SpaceLimit). Write commands from hosts to the volume or pool are failing. The related standard message is

```
Thin provisioned <Volume or Pool> with identifier <Volume or Pool ID> capacity in
                        use exceeded available limit.
```

30.1.3.4.3 Capacity Okay

This is an alert message indicating that the actual capacity of a volume or pool is no longer in a capacity warning or critical state. The related standard message is

```
Thin provisioned <Volume or Pool> with identifier <Volume or Pool ID> capacity
                        condition cleared.
```

30.2 Health and Fault Management Consideration

Not defined in this standard.

30.3 Cascading Considerations

Not defined in this standard.

30.4 Supported Profiles, Subprofiles, and Packages

Table 657 describes the supported profiles for Thin Provisioning.

Table 657 - Supported Profiles for Thin Provisioning

Profile Name	Organization	Version	Requirement	Description
Job Control	SNIA	1.5.0	Optional	
Extent Composition	SNIA	1.6.0	Optional	

30.5 Methods of the Profile

This profile uses the same methods and approach to creating/modifying volumes and pools as Block Services, with additional properties used for active management of thin provisioned elements. The next few sections provides details of how these properties are used.

30.5.1 StoragePool GetSupportedSizes() and GetSupportedSizeRanges()

When a client invokes GetSupportedSizes() or GetSupportedSizeRanges() with ElementType set to 5 (Thin Provisioned Volume) or 6 (Thin Provisioning Logical Disk), the instrumentation shall return size information relative to the value of SpaceLimitDetermination for the related pools.

- For pools with SpaceLimitDetermination of 2 (Allocated), the instrumentation shall return sizes using the same approach for fully provisioned volumes as described in Clause 5: Block Services Package.
- For pools with SpaceLimitDetermination set to 3 (Quota) or 4 (Limitless), the sizes returned should not exceed the value of SpaceLimit for pools supporting thin provisioning.
- For pools with SpaceLimitDetermination of 5 (Fully Provisioned) or with SpaceLimitDetermination unspecified or null, the provider shall return 3 (Invalid Element Type).

When a client invokes GetSupportedSizes() or GetSupportedSizeRanges() with ElementType set to 3 (Storage Volume) or 4 (Logical Disk),

- For pools with SpaceLimitDetermination of 2 (Allocated), 3 (Quota) or 4 (Limitless), the provider shall return 3 (Invalid Element Type)

30.5.2 StorageSetting CreateSetting

When creating thinly provisioned StoragePools: ThinProvisionedPoolType shall be set to a value from SupportedStorageElementTypes in the parent pool's applicable StorageConfigurationCapabilities instance.

30.5.3 StorageConfigurationService CreateOrModifyStoragePool()

CreateOrModifyStoragePool is used to create a thinly provisioned pool. The ElementType parameter (e.g., ThinlyProvisionedAllocatedStoragePool) shall be included in SupportedStorageElementTypes in the StorageConfigurationCapabilities instances applicable to the pool specified in the InPools parameter.

Note: If the client sets SpaceLimitDetermination to Quota, the Size parameter becomes the value of SpaceLimit in the created pool.

Note: CreateOrModifyStoragePool also supports pool creation using InExtents, instead of InPools.

30.5.4 StorageConfigurationService CreateOrModifyElementFromElements()

CreateOrModifyElementFromElements may be used to create a pool from extents

To create a thinly-provisioned pool, the ElementType shall be ThinlyProvisionedAllocatedStoragePool, ThinlyProvisionedQuotaStoragePool, or ThinlyProvisionedLimitlessStoragePool.

The size parameter is ignored if ElementType is ThinlyProvisionedAllocatedStoragePool. In this case, the size is set by the block server based on the capacity of the extents allocated to the pool

30.5.5 StorageConfigurationService CreateOrModifyElementFromStoragePool()

CreateOrModifyElementFromStoragePool is used to create a StorageVolume or LogicalDisk. The size parameter holds the desired nominal size. The ElementType parameter shall be ThinlyProvisionedStorageVolume or ThinlyProvisionedLogicalDisk.

See 30.6 Client Considerations and Recipes for examples using these methods.

30.6 Client Considerations and Recipes

30.6.1 Create a Pool from a Parent Pool

Creating a thin provisioned pool follows the same approach as creating fully provisioned pool with the changes in step 1 below. Assume the client wishes to create a pool using the Allocated approach to space determination

1) find a parent pool associated to a StorageConfigurationCapabilities instance where SupportedStorageElementTypes incudes ThinlyProvisionedAllocatedStoragePool

2) create a (or locate an existing usable) StorageSetting instance

3) Call CreateOrModifyStoragePool

- the StorageSetting as the Goal Parameter
- the appropriate parent pool as the PoolToDrawFrom,
- the size parameter is set to the client’s requested size
- ElementType is ThinlyProvisionedAllocatedStoragePool

Note: If the client sets SpaceLimitDetermination to Quota, the Size parameter becomes the value of SpaceLimit in the created pool.

```
// DESCRIPTION
// The goal of this recipe is to create a thin provisioned pool
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
// 1.  A reference to a CIM_ComputerSystem storage array is previously
//    defined in the $BlockServer-> variable
// 2.  #PoolSize is set to the size for the new Storage Pool in bytes
// 3.  #StorageElementClass is set to the class name of the element being created
//    like CIM_StorageVolume or CIM_LogicalDisk.
// 4. #ElementType is set to the element to created:
//    ThinlyProvisionedAllocatedStoragePool


// 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]

    // See if the service supports thin provisioned pool creation
    // There should be zero or only one StorageConfigurationCapabilities instance
    $ServiceCapabilities[] = Associators(
      $StorageConfigurationService->,
      “CIM_ElementCapabilities”,
      “SNIA_StorageConfigurationCapabilities”,
      null, null, false, false, null)

    if ($ServiceCapabilities[] == null ||
        $ServiceCapabilities[].length == 0) {
       <ERROR! Creation of thin provisioned pools not supported>
    }

    if (contains(2, // Storage Pool Creation
         $ServiceCapabilities[0].SupportedSynchronousActions[])
      || contains(2, // Storage Pool Creation
         $ServiceCapabilities[0].SupportedAsynchronousActions[]))
    {
      #SupportsPoolCreation = true
    }

    if (contains(
      2, // Storage Pool Creation
      $ServiceCapabilities[0].SupportedAsyncronousActions[]))
    {
      #PoolCreationProducesJob = true
    }

    if (contains(7, // 7 = ThinlyProvisionedAllocatedStoragePool
      $ServiceCapabilities[0].SupportedStorageElementTypes[]))
    {
      #SupportsThinPoolCreation = true
    }
```

```
// Return if thin provisioned pools cannot be created
if ((#SupportesPoolCreation == false) &&
    (#SupportsThinPoolCreation == false)) {
   <ERROR! Creation of thin provisioned pools not supported>
}
$StorageCapabilitiesOffered = $ServiceCapabilities[0]

// Step 2. Enumerate over the CIM_HostedStoragePool associations to find
//  a StoragePool from which thin provisioned storage pools might be created.

$PoolToDrawFrom-> = null

// Find the associated StoragePools
$StoragePools[] = Associators(
  $BlockServer->,
  “CIM_HostedStoragePool”,
  “CIM_StoragePool”,
  null, null,  false, false, null)

for #i in $StoragePools[]
{
  // Step 3. For each StoragePool, follow the CIM_ElementCapabilities
  //  asociation to the StorageCapabilities of that pool. Compare the
  //  StorageCapabilities to the desired StorageSetting and find the
  //  best match.

  // See if this pool has its own StorageConfigurationCapabilities.
  $PoolServiceCapabilities[] = Associators(
    $StoragePools[#i].getObjectPath(),
    “CIM_ElementCapabilities”,
    “SNIA_StorageConfigurationCapabilities”,
    null, null, false, false, null)

  if( $PoolServiceCapabilities[] != null ) {
    // see if there is a capability for this pool to create the proper pool
    for #c in $PoolServiceCapabilities[]
    {
      if ( contains(2, // Storage Pool Creation
                    $PoolServiceCapabilities[#c].SupportedSynchronousActions[])
               || contains(2, // Storage Pool Creation
                    $PoolServiceCapabilities[#c].SupportedAsynchronousActions[]))
      {
        #Pool_SupportsPoolCreation = true
      }
      if (contains(2, // Storage Pool Creation
                      $PoolServiceCapabilities[#c].SupportedAsyncronousActions[]))
      {
```

```
      #Pool_PoolCreationProducesJob = true
    }
    #Pool_SupportsThinPoolCreation = contains(
      7, // 7 = ThinlyProvisionedAllocatedStoragePool
      $PoolServiceCapabilities[0].SupportedStorageElementTypes[])

    if (#Pool_SupportsPoolCreation == true &&
        #Pool_SupportsThinPoolCreation == true) {
         #SupportsPoolCreation = true
         #SupportsThinPoolCreation = true
         $StorageCapabilitiesOffered = $PoolServiceCapabilities[#c]
      break;
    }
  }
} // end of  if( $PoolServiceCapabilities[] != null )

if($StorageCapabilitiesOffered != null)  {

  $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) || (%OutArguments[“NewSetting”] == null))
  {
    <ERROR! Unable to create storage setting >
  }

  $GeneratedStorageSetting-> = %OutArguments[“NewSetting”]

  // Determine the possible size, closest to the requested size
  #PossibleSize = &Block_Services_PoolSizeAvailable(
    $PoolToDrawFrom->,
    $GeneratedStorageSetting->,
    #RequestedSize,
    #ElementType)
    if(0 != #PossibleSize) // we found a size close to #RequestedSize
    {
      break;
    }
    else
    {
      // Causes an error to be returned if there are no more candidate Pools
      $PoolToDrawFrom-> = NULL;
    }
  }
} // for

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 == true)
{
  // ‘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
%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 >
}

if(#PoolCreationProducesJob == true && $PoolCreationJob-> != null)
{
  $PoolCreationJob-> = %OutArguments[“Job”]

  //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
}
else {
  $PoolToDrawFrom-> = %OutArguments[“Pool”]
}

// Use the new pool
```

30.6.2 Create a Pool from Extents

This recipe is similar to the above except it uses CreateOrModifyElementFromElement.

ElementType is ThinlyProvisionedAllocatedStoragePool, ThinlyProvisionedQuotaStoragePool, and ThinlyProvisionedLimitlessStoragePool.

The size parameter is ignored if ElementType is ThinlyProvisionedAllocatedStoragePool. In this case, the size is set by the block server based on the capacity of the extents allocated to the pool.

30.6.3 Creating a Thinly Provisioned Volume

Creating a thin provisioned volume follows the same approach as creating fully provisioned volume with the following extra steps:

1) verify that the parent pool supports thin provisioned child volumes by verifying that StorageConfigurationCapabilities.SupportedStorageElementTypes incudes ThinlyProvisionedStorageVolume

2) use StorageConfigurationCapabilities.ThinProvisionedClientSettableReserve to determine whether the client can specify an desired initial reserve

3) create a (or locate and existing usable) StorageSetting instance, set ThinProvisionedInitialReserve as needed

4) call CreateOrModifyElementFromStoragePool using

- the StorageSetting as the Goal,

- the appropriate parent pool as the PoolToDrawFrom,
- the size parameter holds the nominal size,
- ElementType is ThinlyProvisionedStorageVolume

```
// DESCRIPTION
// The goal of this recipe is to create a thin provisioned StorageVolume
//
// PRE-EXISTING CONDITIONS AND ASSUMPTION
// 1. A reference to a CIM_ComputerSystem storage array is previously
//    defined in the $BlockServer-> variable
// 2. #PossibleSize is set to the size for the new StorageVolume in bytes
// 3. #StorageElementClass is set to the class name of the element being created
//    like CIM_StorageVolume or CIM_LogicalDisk.
// 4. #ElementType is set to the element to created:
//    ThinlyProvisionedStorageVolume


// 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]
```

```
// See if the service supports thin provisioned pool creation
// There should be only one StorageConfigurationCapabilities instance
$ServiceCapabilities[] = Associators(
  $StorageConfigurationService->,
  “CIM_ElementCapabilities”,
  “SNIA_StorageConfigurationCapabilities”,
  null, null, false, false, null)

#SupportsElementCreationSync = contains(5, // Storage Element Creation
    $ServiceCapabilities[0].SupportedSynchronousActions[])
#SupportsElementCreationFeature = contains(3, // StorageElementCreation
    $ServiceCapabilities[0].SupportedStorageElementFeatures[])
#ElementCreationProducesJob = contains(5, // Storage Element Creation
    $ServiceCapabilities[0].SupportedAsynchronousActions[])
#SupportsThinPoolCreation = contains(5, // 5 = ThinlyProvisionedStorageVolume
    $ServiceCapabilities[0].SupportedStorageElementTypes[])

// If a storage element can not be created and that storage element is
// neither created synchronously or asynchronously, then fail the test
if (#SupportedElementCreationFeature == false ||
    (#SupportedElementCreationSync == false &&
     #ElementCreationProducesJob == false) ||
    #SupportsThinPoolCreation == false)
{
   <ERROR! Thin provisioned StorageElement creation is not supported.>
}
$StorageCapabilitiesOffered = $ServiceCapabilities[0]

// Step 2. Enumerate over the CIM_HostedStoragePool associations to find
//  a StoragePool from which thin provisioned storage pools might be created.

$PoolToDrawFrom-> = null

// Find the associated StoragePools
$StoragePools[] = Associators(
  $BlockServer->,
  “CIM_HostedStoragePool”,
  “CIM_StoragePool”,
  null, null,  false, false, null)

for #i in $StoragePools[]
{

  // Skip primordial pools
  if ($StoragePool[#i].Primordial == true)
  {
    <continue>
```

```
}

// Step 3. For each StoragePool, follow the CIM_ElementCapabilities
//  asociation to the StorageCapabilities of that pool. Compare the
//  StorageCapabilities to the desired StorageSetting and find the
//  best match.

// See if this pool has its own StorageConfigurationCapabilities.
$PoolServiceCapabilities[] = Associators(
  $StoragePools[#i].getObjectPath(),
  “CIM_ElementCapabilities”,
  “SNIA_StorageConfigurationCapabilities”,
  null, null, false, false, null)

if( $PoolServiceCapabilities[] != null ) {
  for #c in $PoolServiceCapabilities[]
  {
    #SupportsElementCreationSync = contains(5, // Storage Element Creation
        $PoolServiceCapabilities[#c].SupportedSynchronousActions[])
    #SupportsElementCreationFeature = contains(3, // StorageElementCreation
        $PoolServiceCapabilities[#c].SupportedStorageElementFeatures[])
    #ElementCreationProducesJob = contains(5, // Storage Element Creation
        $PoolServiceCapabilities[#c].SupportedAsynchronousActions[])
    #SupportsThinPoolCreation = contains(5, // 5 =
                    ThinlyProvisionedStorageVolume
        $PoolServiceCapabilities[#c].SupportedStorageElementTypes[])

    // If a storage element can not be created and that storage element is
    // neither created synchronously or asynchronously, then skip this capability
    if (#SupportedElementCreationFeature == false ||
        (#SupportedElementCreationSync == false &&
         #ElementCreationProducesJob == false) ||
        #SupportsThinPoolCreation == false)
    {
       <continue>
    }
    else {
      $StorageCapabilitiesOffered = $ServiceCapabilities[0]
    }
  }
} // end of  if( $PoolServiceCapabilities[]-> != null )

$PoolToDrawFrom-> = $StoragePool[#i].getObjectPath()

// Step 4. Register for indications on configuration jobs
if(#SupportedElementProducesJob == true)
{
  // ‘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 5. 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 4
            // 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, 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 storage element will be created,
            $CreatedElement-> = $CreatedElement[0].getObjectPath()
          }
        }
```

30.6.4 Capacity Properties for Fully-provisioned RAID1 Volume

Figure 158 demonstrates two approaches for setting capacity properties. In one approach, the capacity due to redundancy on RAID is included in the concrete pool; in the other approach, the capacity in the concrete pool reflects the factoring out of the RAID overhead. In this array configuration, there is a primordial pool showing the capacity from two 502 block disks. (The disks are not modeled, a valid option in SMI-S.) Each disk has two blocks of metadata - yielding 2 * 500 usable blocks. The block server has assembled these two disks into a RAID1 set (represented by the Concrete pool)—a process which consumes four blocks for metadata. A single StorageVolume is allocated. This volume consumes 110 blocks. The SpaceConsumed value of 224 in the upper right reflects two times 110 (the nominal volume capacity times 2 for RAID1) plus four blocks metadata.

Note that Block Services allows an arbitrary number of concrete pools between the primordial pool and the pool from which the volume is allocated, so other sets of instances could also represent the same RAID1 configuration.

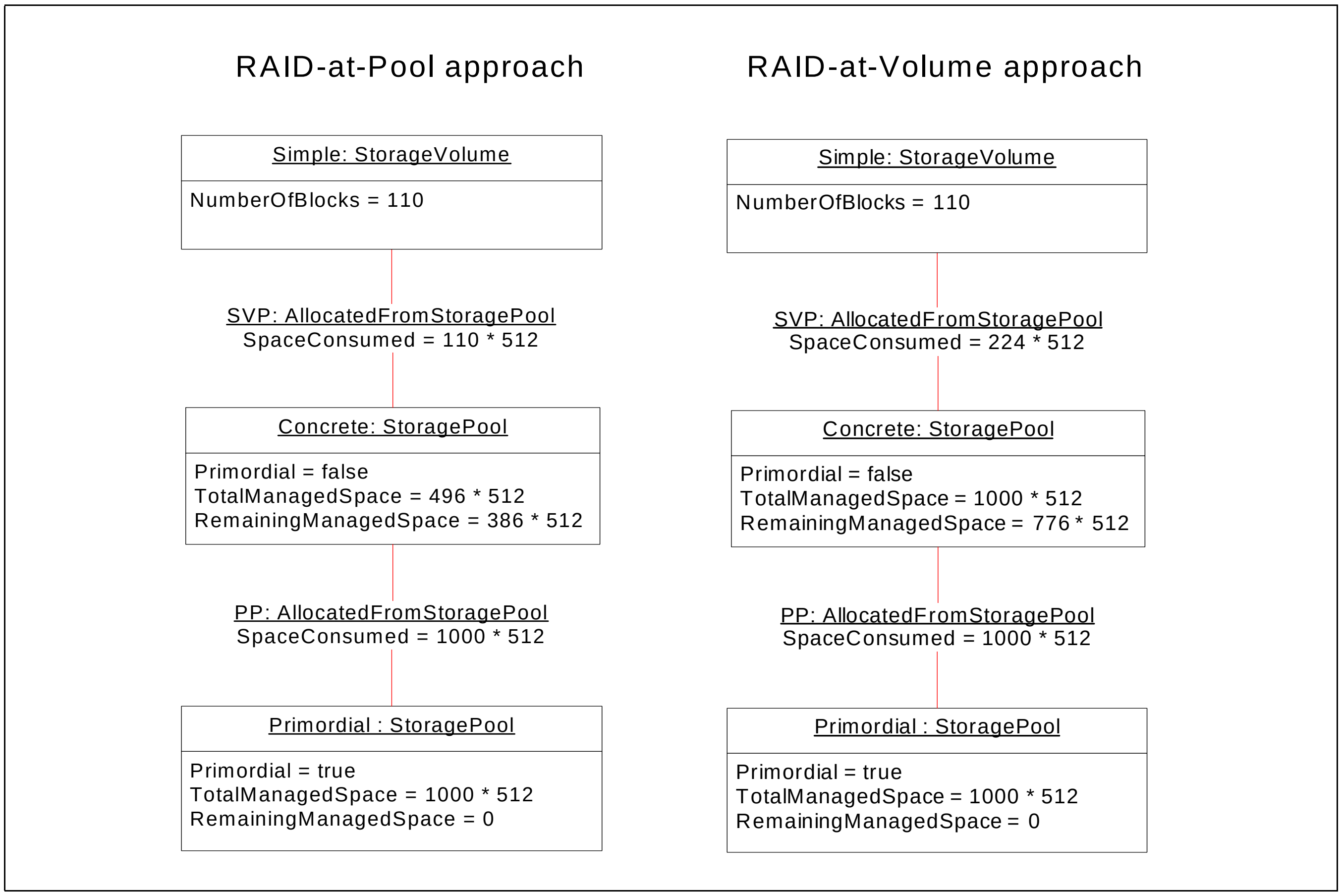
RAID-at-Pool approach
Simple: StorageVolume
NumberOfBlocks = 110
SVP: AllocatedFromStoragePool
SpaceConsumed = 110 * 512
Concrete: StoragePool
Primordial = false
TotalManagedSpace = 496 * 512
RemainingManagedSpace = 386 * 512
PP: AllocatedFromStoragePool
SpaceConsumed = 1000 * 512
Primordial : StoragePool
Primordial = true
TotalManagedSpace = 1000 * 512
RemainingManagedSpace = 0
RAID-at-Volume approach
Simple: StorageVolume
NumberOfBlocks = 110
SVP: AllocatedFromStoragePool
SpaceConsumed = 224 * 512
Concrete: StoragePool
Primordial = false
TotalManagedSpace = 1000 * 512
RemainingManagedSpace = 776 * 512
PP: AllocatedFromStoragePool
SpaceConsumed = 1000 * 512
Primordial : StoragePool
Primordial = true
TotalManagedSpace = 1000 * 512
RemainingManagedSpace = 0

Figure 158 - RAID1 Capacity after Volume Creation

30.6.5 Capacity Properties for Thin Provisioning

Figure 159 builds on Figure 158, showing a newly created thinly provisioned volume with of 50 blocks consumed.

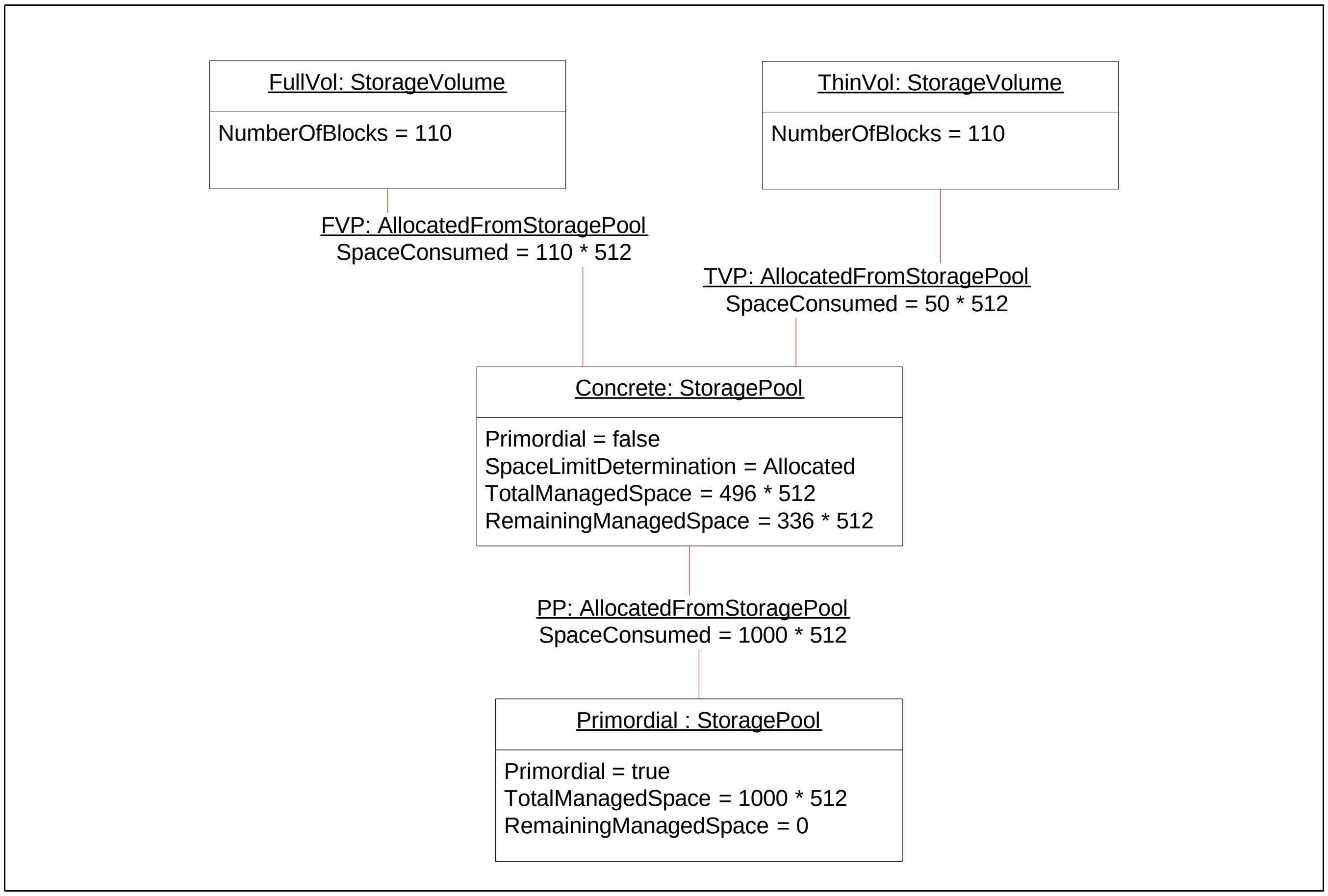
FullVol: StorageVolume
NumberOfBlocks = 110
ThinVol: StorageVolume
NumberOfBlocks = 110
FVP: AllocatedFromStoragePool
SpaceConsumed = 110 * 512
TVP: AllocatedFromStoragePool
SpaceConsumed = 50 * 512
Concrete: StoragePool
Primordial = false
SpaceLimitDetermination = Allocated
TotalManagedSpace = 496 * 512
RemainingManagedSpace = 336 * 512
PP: AllocatedFromStoragePool
SpaceConsumed = 1000 * 512
Primordial : StoragePool
Primordial = true
TotalManagedSpace = 1000 * 512
RemainingManagedSpace = 0

Figure 159 - RAID1 Capacity with Thin Volume and RAID-at-Pool Approach

Figure 160 adds the same thin volume, but uses the RAID-on-Volume approach.

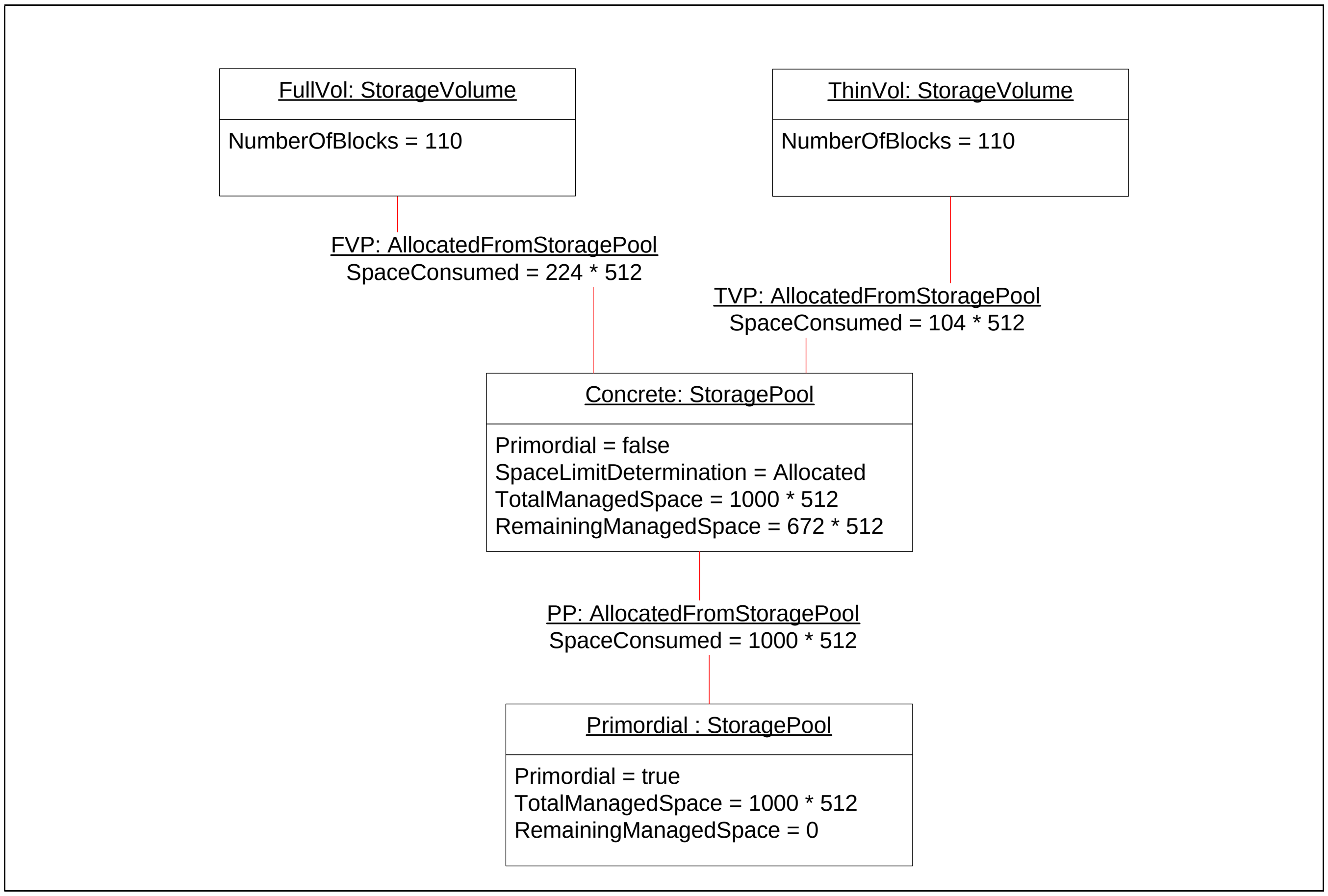
FullVol: StorageVolume
NumberOfBlocks = 110
ThinVol: StorageVolume
NumberOfBlocks = 110
FVP: AllocatedFromStoragePool
SpaceConsumed = 224 * 512
TVP: AllocatedFromStoragePool
SpaceConsumed = 104 * 512
Concrete: StoragePool
Primordial = false
SpaceLimitDetermination = Allocated
TotalManagedSpace = 1000 * 512
RemainingManagedSpace = 672 * 512
PP: AllocatedFromStoragePool
SpaceConsumed = 1000 * 512
Primordial : StoragePool
Primordial = true
TotalManagedSpace = 1000 * 512
RemainingManagedSpace = 0

Figure 160 - RAID1 Capacity with Thin Volume and RAID-at-Volume Approach

30.7 Registered Name and Version

Thin Provisioning version 1.6.0 (Component Profile)

CIM Schema Version: 2.28

Specializes SNIA Block Services version 1.6.0

30.8 CIM Elements

Table 658 describes the CIM elements for Thin Provisioning.

Table 658 - CIM Elements for Thin Provisioning

Element Name	Requirement	Description
30.8.1 CIM_AllocatedFromStoragePool (Pool from Pool)	Mandatory	AllocatedFromStoragePool.
30.8.2 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.
30.8.3 CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StorageVolume or LogicalDisk)	Optional	Expressed the ability for the element to be named or have its state changed.
30.8.4 CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StoragePool)	Optional	Expressed the ability for the element to be named or have its state changed.
30.8.5 CIM_ElementCapabilities (StorageCapabilities to StorageConfigurationService)	Optional	Associates StorageCapabilities with StorageConfigurationService. This StorageCapabilities shall represent the capabilities of the entire implementation.
30.8.6 CIM_ElementCapabilities (StorageCapabilities to StoragePool)	Mandatory	Associates StorageCapabilities with StoragePool. This StorageCapabilities shall represent the capabilities of the StoragePool to which it is associated.
30.8.7 CIM_ElementCapabilities (StorageConfigurationCapabilities to StorageConfigurationService)	Mandatory	Associates StorageConfigurationCapabilities with StorageConfigurationService.
30.8.8 CIM_ElementCapabilities (StorageConfigurationCapabilities to concrete StoragePool)	Optional	Associates StorageConfigurationCapabilities with StoragePool.
30.8.9 CIM_ElementCapabilities (StorageConfigurationCapabilities to primordial StoragePool)	Optional	Associates StorageConfigurationCapabilities with StoragePool.
30.8.10 CIM_ElementCapabilities (Used to declare the naming capabilities of the StoragePool)	Optional	Associates EnabledLogicalElementCapabilities with StorageConfigurationService.
30.8.11 CIM_ElementCapabilities (Used to declare the naming capabilities of the StorageVolume or LogicalDisk)	Optional	Associates EnabledLogicalElementCapabilities with StorageConfigurationService.
30.8.12 CIM_ElementSettingData	Mandatory	

Table 658 - CIM Elements for Thin Provisioning

Element Name	Requirement	Description
30.8.13 CIM_EnabledLogicalElementCapabilities (For StorageConfigurationService)	Optional	This class is used to express the naming and possible requested state change possibilities for storage elements.
30.8.14 CIM_EnabledLogicalElementCapabilities (For StoragePool)	Optional	This class is used to express the naming and possible requested state change possibilities for storage pools.
30.8.15 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.
30.8.16 CIM_FilterCollection (Block Services ProfileSpecificLifecycleIndicationFilterCollection)	Optional	Experimental. This is a collection of Lifecycle IndicationFilters defined by the Block Services Profile.
30.8.17 CIM_FilterCollection (Thin Provisioning 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.
30.8.18 CIM_FilterCollection (Thin Provisioning ProfileSpecificAlertIndicationFilterCollection)	Optional	Experimental. This is a collection of Lifecycle IndicationFilters defined by the Thin Provisioning Profile.
30.8.19 CIM_HostedCollection (Block Services to ProfileSpecificLifecycleIndicationFilterCollection)	Optional	Experimental.
30.8.20 CIM_HostedCollection (System to Thin Provisioning ProfileSpecificAlertIndicationFilterCollection)	Optional	Experimental.
30.8.21 CIM_HostedCollection (System to Thin Provisioning predefined IndicationFilters)	Conditional	Experimental. Conditional requirement: Required if the Experimental Indication Profile is supported and the SNIA_IndicationConfigurationCapabilities.SupportedFeatures='5' (Predefined Filter Collections).

Table 658 - CIM Elements for Thin Provisioning

Element Name	Requirement	Description
30.8.22 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).
30.8.23 CIM_HostedService	Conditional	Conditional requirement: Support for StorageConfigurationService.
30.8.24 CIM_HostedStoragePool	Mandatory	
30.8.25 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.
30.8.26 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.
30.8.27 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.
30.8.28 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 658 - CIM Elements for Thin Provisioning

Element Name	Requirement	Description
30.8.29 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.
30.8.30 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.
30.8.31 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.
30.8.32 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.
30.8.33 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.
30.8.34 CIM_IndicationFilter (Thin Provisioning Critical)	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 an alert when the Thin Provisioning critical threshold is reached.

Table 658 - CIM Elements for Thin Provisioning

Element Name	Requirement	Description
30.8.35 CIM_IndicationFilter (Thin Provisioning Threshold Cleared)	Optional	Experimental. This is the 'pre-defined' CIM_IndicationFilter instance for an alert when a Thin Provisioning threshold has been cleared.
30.8.36 CIM_IndicationFilter (Thin Provisioning Warning)	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 an alert when the Thin Provisioning warning threshold is reached.
30.8.37 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.
30.8.38 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.
30.8.39 CIM_LogicalDisk	Conditional	Conditional requirement: Referenced from Volume Management - LogicalDisk is mandatory. A LogicalDisk is allocated from a concrete StoragePool.
30.8.40 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).

Table 658 - CIM Elements for Thin Provisioning

Element Name	Requirement	Description
30.8.41 CIM_MemberOfCollection (Block Services ProfileSpecificLifecycleIndicationFilterCollection to Block Services Filters)	Optional	Experimental. This associates the Block Services ProfileSpecificLifecycleIndicationFilterCollection to the Filters defined by the Block Services Profile.
30.8.42 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.
30.8.43 CIM_MemberOfCollection (Predefined Filter Collection to Thin Provisioning 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 Thin Provisioning predefined FilterCollection to the predefined Filters supported by the implementation.
30.8.44 CIM_MemberOfCollection (Thin Provisioning 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 Thin Provisioning predefined FilterCollection to the FilterCollection for the autonomous profile (e.g., the Array FilterCollection).
30.8.45 CIM_MemberOfCollection (Thin Provisioning ProfileSpecificAlertIndicationFilterCollection to Thin Provisioning Filters)	Optional	Experimental. This associates the Thin Provisioning ProfileSpecificAlertIndicationFilterCollection to the Filters defined by the Thin Provisioning Profile.
30.8.46 CIM_OwningJobElement	Conditional	Conditional requirement: Support for Job Control profile.
30.8.47 CIM_StorageCapabilities	Mandatory	
30.8.48 CIM_StorageConfigurationCapabilities (Concrete)	Mandatory	StorageConfigurationCapabilities as defined in Block Services, with the addition of SupportedStorageElementTypes for thin pools and volumes.

Table 658 - CIM Elements for Thin Provisioning

Element Name	Requirement	Description
30.8.49 CIM_StorageConfigurationCapabilities (Global)	Mandatory	StorageConfigurationCapabilities as defined in Block Services, with the addition of SupportedStorageElementTypes for thin pools and volumes.
30.8.50 CIM_StorageConfigurationCapabilities (Primordial)	Mandatory	StorageConfigurationCapabilities as defined in Block Services, with the addition of SupportedStorageElementTypes for thin pools and volumes.
30.8.51 CIM_StorageConfigurationService	Mandatory	StorageConfigurationService as defined in Block Services, adding thin provisioning values to the ElementType parameter.
30.8.52 CIM_StoragePool (Concrete)	Mandatory	Concrete StoragePool as defined in Block Services with the addition of SpaceLimit, SpaceLimitDetermination, and ThinProvisionMetaDataSpace.
30.8.53 CIM_StoragePool (Empty)	Mandatory	Empty StoragePool as defined in Block Services with the addition of SpaceLimit, SpaceLimitDetermination, and ThinProvisionMetaDataSpace.
30.8.54 CIM_StoragePool (Primordial)	Mandatory	Primordial StoragePool as defined in Block Services with the addition of SpaceLimit, SpaceLimitDetermination, and ThinProvisionMetaDataSpace.
30.8.55 CIM_StorageSetting	Mandatory	StorageSetting as defined in Block Services with the addition of Thin Provisioning properties.
30.8.56 CIM_StorageSettingWithHints	Optional	
30.8.57 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").

Table 658 - CIM Elements for Thin Provisioning

Element Name	Requirement	Description
30.8.58 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.
30.8.59 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 the Storage Management Technical Specification, Part 1 Common Architecture for details on how to set Name, NameFormat, and NameNamespace properties.
30.8.60 CIM_SystemDevice (System to StorageVolume or LogicalDisk)	Mandatory	Associates top level system from Array, Virtualizer, ... to StorageVolume or LogicalDisk.
SELECT * FROM CIM_InstCreation WHERE SourceInstance ISA CIM_StoragePool	Mandatory	Creation/Deletion of StoragePool. See section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.24 CIM_IndicationFilter (Storage Pool Creation).
SELECT * FROM CIM_InstDeletion WHERE SourceInstance ISA CIM_StoragePool	Mandatory	Deletion of StoragePool. See section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.25 CIM_IndicationFilter (Storage Pool Deletion).

Table 658 - CIM Elements for Thin Provisioning

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 section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.27 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 section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.28 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 section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.31 CIM_IndicationFilter (WQL Storage Volume OperationalStatus).
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 section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.29 CIM_IndicationFilter (Storage Volume OperationalStatus).

Table 658 - CIM Elements for Thin Provisioning

Element Name	Requirement	Description
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 section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.21 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 section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.22 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 section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.30 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 section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.23 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 section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.26 CIM_IndicationFilter (Storage Pool TotalManagedSpace).
SELECT * FROM CIM_AlertIndication WHERE OwningEntity='SNIA' and MessageID='DRM28'	Mandatory	Indication that capacity is running low. See 30.1.3.4.1 Capacity Warning.

Table 658 - CIM Elements for Thin Provisioning

Element Name	Requirement	Description
SELECT * FROM CIM_AlertIndication WHERE OwningEntity='SNIA' and MessageID='DRM29'	Mandatory	Indication that capacity is has run out. See 30.1.3.4.2 Capacity Critical.
SELECT * FROM CIM_AlertIndication WHERE OwningEntity='SNIA' and MessageID='DRM30'	Optional	Indication that capacity condition has been cleared. See 30.1.3.4.3 Capacity Okay.

30.8.1 CIM_AllocatedFromStoragePool (Pool from Pool)

Created By: Static

Modified By: Static

Deleted By: Static

Requirement: Mandatory

Table 659 describes class CIM_AllocatedFromStoragePool (Pool from Pool).

Table 659 - SMI Referenced Properties/Methods for CIM_AllocatedFromStoragePool (Pool from Pool)

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

30.8.2 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 660 describes class CIM_AllocatedFromStoragePool (Volume or LogicalDisk from Pool).

Table 660 - SMI Referenced Properties/Methods for CIM_AllocatedFromStoragePool (Volume or LogicalDisk from Pool)

Properties	Flags	Requirement	Description & Notes
SpaceConsumed		Mandatory	

Table 660 - SMI Referenced Properties/Methods for CIM_AllocatedFromStoragePool (Volume or LogicalDisk from Pool)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	
Dependent		Mandatory	

30.8.3 CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StorageVolume or LogicalDisk)

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

Table 661 describes class CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StorageVolume or LogicalDisk).

Table 661 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StorageVolume or LogicalDisk)

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

30.8.4 CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StoragePool)

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

Table 662 describes class CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StoragePool).

Table 662 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (EnabledLogicalElementCapabilities to StoragePool)

Properties	Flags	Requirement	Description & 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.

30.8.5 CIM_ElementCapabilities (StorageCapabilities to StorageConfigurationService)

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

Table 663 describes class CIM_ElementCapabilities (StorageCapabilities to StorageConfigurationService).

Table 663 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (StorageCapabilities to StorageConfigurationService)

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

30.8.6 CIM_ElementCapabilities (StorageCapabilities to StoragePool)

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

Table 664 describes class CIM_ElementCapabilities (StorageCapabilities to StoragePool).

Table 664 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (StorageCapabilities to StoragePool)

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

30.8.7 CIM_ElementCapabilities (StorageConfigurationCapabilities to StorageConfigurationService)

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

Table 665 describes class CIM_ElementCapabilities (StorageConfigurationCapabilities to StorageConfigurationService).

Table 665 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (StorageConfigurationCapabilities to StorageConfigurationService)

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

30.8.8 CIM_ElementCapabilities (StorageConfigurationCapabilities to concrete StoragePool)

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

Table 666 describes class CIM_ElementCapabilities (StorageConfigurationCapabilities to concrete StoragePool).

Table 666 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (StorageConfigurationCapabilities to concrete StoragePool)

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

30.8.9 CIM_ElementCapabilities (StorageConfigurationCapabilities to primordial StoragePool)

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

Table 667 describes class CIM_ElementCapabilities (StorageConfigurationCapabilities to primordial StoragePool).

Table 667 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (StorageConfigurationCapabilities to primordial StoragePool)

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

30.8.10 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 668 describes class CIM_ElementCapabilities (Used to declare the naming capabilities of the StoragePool).

Table 668 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (Used to declare the naming capabilities of the StoragePool)

Properties	Flags	Requirement	Description & 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.

30.8.11 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 669 describes class CIM_ElementCapabilities (Used to declare the naming capabilities of the StorageVolume or LogicalDisk).

Table 669 - SMI Referenced Properties/Methods for CIM_ElementCapabilities (Used to declare the naming capabilities of the StorageVolume or LogicalDisk)

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

30.8.12 CIM_ElementSettingData

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

Table 670 describes class CIM_ElementSettingData.

Table 670 - SMI Referenced Properties/Methods for CIM_ElementSettingData

Properties	Flags	Requirement	Description & 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.

30.8.13 CIM_EnabledLogicalElementCapabilities (For StorageConfigurationService)

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

Table 671 describes class CIM_EnabledLogicalElementCapabilities (For StorageConfigurationService).

Table 671 - SMI Referenced Properties/Methods for CIM_EnabledLogicalElementCapabilities (For StorageConfigurationService)

Properties	Flags	Requirement	Description & 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 Capacilities.

Table 671 - SMI Referenced Properties/Methods for CIM_EnabledLogicalElementCapabilities (For StorageConfigurationService)

Properties	Flags	Requirement	Description & Notes
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.

30.8.14 CIM_EnabledLogicalElementCapabilities (For StoragePool)

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

Table 672 describes class CIM_EnabledLogicalElementCapabilities (For StoragePool).

Table 672 - SMI Referenced Properties/Methods for CIM_EnabledLogicalElementCapabilities (For StoragePool)

Properties	Flags	Requirement	Description & 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, it may be assumed that the state may not be changed.

30.8.15 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 673 describes class CIM_FilterCollection (Block Services Predefined FilterCollection).

Table 673 - SMI Referenced Properties/Methods for CIM_FilterCollection (Block Services Predefined FilterCollection)

Properties	Flags	Requirement	Description & 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:Predefined'.

30.8.16 CIM_FilterCollection (Block Services ProfileSpecificLifecycleIndicationFilterCollection)

Experimental. This is a collection of Lifecycle IndicationFilters defined by the Block Services Profile.

Requirement: Optional

Table 674 describes class CIM_FilterCollection (Block Services ProfileSpecificLifecycleIndicationFilterCollection).

Table 674 - SMI Referenced Properties/Methods for CIM_FilterCollection (Block Services ProfileSpecificLifecycleIndicationFilterCollection)

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	See the InstanceID definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.13 CIM_FilterCollection (StaticFilterCollection).
CollectionName		Mandatory	The value of CollectionName shall be 'SNIA:Block Services:ProfileSpecificLifecycleIndicationFilterCollection'.

30.8.17 CIM_FilterCollection (Thin Provisioning Predefined FilterCollection)

Experimental. This is a collection of predefined IndicationFilters to which a client may subscribe. A Thin Provisioning 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 675 describes class CIM_FilterCollection (Thin Provisioning Predefined FilterCollection).

Table 675 - SMI Referenced Properties/Methods for CIM_FilterCollection (Thin Provisioning Predefined FilterCollection)

Properties	Flags	Requirement	Description & 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:Thin Provisioning:Predefined'.

30.8.18 CIM_FilterCollection (Thin Provisioning ProfileSpecificAlertIndicationFilterCollection)

Experimental. This is a collection of Lifecycle IndicationFilters defined by the Thin Provisioning Profile.

Requirement: Optional

Table 676 describes class CIM_FilterCollection (Thin Provisioning ProfileSpecificAlertIndicationFilterCollection).

Table 676 - SMI Referenced Properties/Methods for CIM_FilterCollection (Thin Provisioning ProfileSpecificAlertIndicationFilterCollection)

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	See the InstanceID definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.13 CIM_FilterCollection (StaticFilterCollection).
CollectionName		Mandatory	The value of CollectionName shall be 'SNIA:Thin Provisioning:ProfileSpecificAlertIndicationFilterCollection'.

30.8.19 CIM_HostedCollection (Block Services to ProfileSpecificLifecycleIndicationFilterCollection)

Experimental.

Requirement: Optional

Table 677 describes class CIM_HostedCollection (Block Services to ProfileSpecificLifecycleIndicationFilterCollection).

Table 677 - SMI Referenced Properties/Methods for CIM_HostedCollection (Block Services to ProfileSpecificLifecycleIndicationFilterCollection)

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	Reference to the ProfileSpecificLifecycleIndicationFilterCollection for the Block Services Profile.
Antecedent		Mandatory	Reference to the 'Top level' System.

30.8.20 CIM_HostedCollection (System to Thin Provisioning ProfileSpecificAlertIndicationFilterCollection)

Experimental.

Requirement: Optional

Table 678 describes class CIM_HostedCollection (System to Thin Provisioning ProfileSpecificAlertIndicationFilterCollection).

Table 678 - SMI Referenced Properties/Methods for CIM_HostedCollection (System to Thin Provisioning ProfileSpecificAlertIndicationFilterCollection)

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	Reference to the ProfileSpecificAlertIndicationFilterCollection for the Thin Provisioning Profile.
Antecedent		Mandatory	Reference to the 'Top level' System.

30.8.21 CIM_HostedCollection (System to Thin Provisioning predefined IndicationFilters)

Experimental.

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

Table 679 describes class CIM_HostedCollection (System to Thin Provisioning predefined IndicationFilters).

Table 679 - SMI Referenced Properties/Methods for CIM_HostedCollection (System to Thin Provisioning predefined IndicationFilters)

Properties	Flags	Requirement	Description & Notes
Dependent		Mandatory	Reference to the predefined FilterCollection for Thin Provisioning.
Antecedent		Mandatory	Reference to the System of the referencing profile.

30.8.22 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 680 describes class CIM_HostedCollection (System to predefined IndicationFilters).

Table 680 - SMI Referenced Properties/Methods for CIM_HostedCollection (System to predefined IndicationFilters)

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

30.8.23 CIM_HostedService

Created By: Static
Modified By: Static
Deleted By: Static
Requirement: Support for StorageConfigurationService.

Table 681 describes class CIM_HostedService.

Table 681 - SMI Referenced Properties/Methods for CIM_HostedService

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

30.8.24 CIM_HostedStoragePool

Requirement: Mandatory

Table 682 describes class CIM_HostedStoragePool.

Table 682 - SMI Referenced Properties/Methods for CIM_HostedStoragePool

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

30.8.25 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 683 describes class CIM_IndicationFilter (Logical Disk Creation).

Table 683 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Logical Disk Creation)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Name		Mandatory	This shall be 'SNIA:Block Services:LogicalDiskCreation'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
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 section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).

30.8.26 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 684 describes class CIM_IndicationFilter (Logical Disk Deletion).

Table 684 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Logical Disk Deletion)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Name		Mandatory	This shall be 'SNIA:Block Services:LogicalDiskDeletion'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
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 section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).

30.8.27 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 685 describes class CIM_IndicationFilter (Logical Disk OperationalStatus).

Table 685 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Logical Disk OperationalStatus)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Name		Mandatory	This shall be 'SNIA:Block Services:LogicalDiskOperationalStatus'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
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 section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).

30.8.28 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 686 describes class CIM_IndicationFilter (Storage Pool Creation).

Table 686 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Storage Pool Creation)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Name		Mandatory	This shall be 'SNIA:Block Services:StoragePoolCreation'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
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 section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).

30.8.29 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 687 describes class CIM_IndicationFilter (Storage Pool Deletion).

Table 687 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Storage Pool Deletion)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Name		Mandatory	This shall be 'SNIA:Block Services:StoragePoolDeletion'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
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 section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).

30.8.30 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 688 describes class CIM_IndicationFilter (Storage Pool TotalManagedSpace).

Table 688 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Storage Pool TotalManagedSpace)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Name		Mandatory	This shall be 'SNIA:Block Services:StoragePoolTotalManagedSpace'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
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 section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).

30.8.31 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 689 describes class CIM_IndicationFilter (Storage Volume Creation).

Table 689 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Storage Volume Creation)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Name		Mandatory	This shall be 'SNIA:Block Services:StorageVolumeCreation'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
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 section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).

30.8.32 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 690 describes class CIM_IndicationFilter (Storage Volume Deletion).

Table 690 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Storage Volume Deletion)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Name		Mandatory	This shall be 'SNIA:Block Services:StorageVolumeDeletion'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
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 section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).

30.8.33 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 691 describes class CIM_IndicationFilter (Storage Volume OperationalStatus).

Table 691 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Storage Volume OperationalStatus)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Name		Mandatory	This shall be 'SNIA:Block Services:StorageVolumeOperationalStatus'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
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 section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).

30.8.34 CIM_IndicationFilter (Thin Provisioning Critical)

Experimental. This is the 'pre-defined' CIM_IndicationFilter instance for an alert when the Thin Provisioning critical threshold is reached. This would typically occur as a result write activity to the volume or storage pool involved.

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 692 describes class CIM_IndicationFilter (Thin Provisioning Critical).

Table 692 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Thin Provisioning Critical)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Name		Mandatory	This shall be 'SNIA:Thin Provisioning:ThinProvisioningCritical'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Query		Mandatory	SELECT * FROM CIM_AlertIndication WHERE OwningEntity='SNIA' and MessageID='DRM29'.
QueryLanguage		Mandatory	This shall be 'DMTF:CQL'.
ElementName	N	Optional	See the ElementName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).

30.8.35 CIM_IndicationFilter (Thin Provisioning Threshold Cleared)

Experimental. This is the 'pre-defined' CIM_IndicationFilter instance for an alert when a Thin Provisioning threshold has been cleared. This would typically occur as a result freeing or adding space to the volume or storage pool involved.

Created By: Static

Modified By: Static
Deleted By: Static
Requirement: Optional

Table 693 describes class CIM_IndicationFilter (Thin Provisioning Threshold Cleared).

Table 693 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Thin Provisioning Threshold Cleared)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Name		Mandatory	This shall be 'SNIA:Thin Provisioning:ThinProvisioningCleared'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Query		Mandatory	SELECT * FROM CIM_AlertIndication WHERE OwningEntity='SNIA' and MessageID='DRM30'.
QueryLanguage		Mandatory	This shall be 'DMTF:CQL'.
ElementName	N	Optional	See the ElementName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).

30.8.36 CIM_IndicationFilter (Thin Provisioning Warning)

Experimental. This is the 'pre-defined' CIM_IndicationFilter instance for an alert when the Thin Provisioning warning threshold is reached. This would typically occur as a result write activity to the volume or storage pool involved.

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 694 describes class CIM_IndicationFilter (Thin Provisioning Warning).

Table 694 - SMI Referenced Properties/Methods for CIM_IndicationFilter (Thin Provisioning Warning)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Name		Mandatory	This shall be 'SNIA:Thin Provisioning:ThinProvisioningWarning'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Query		Mandatory	SELECT * FROM CIM_AlertIndication WHERE OwningEntity='SNIA' and MessageID='DRM28'.
QueryLanguage		Mandatory	This shall be 'DMTF:CQL'.
ElementName	N	Optional	See the ElementName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).

30.8.37 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 695 describes class CIM_IndicationFilter (WQL Logical Disk OperationalStatus).

Table 695 - SMI Referenced Properties/Methods for CIM_IndicationFilter (WQL Logical Disk OperationalStatus)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Name		Mandatory	This shall be 'SNIA:Block Services:LogicalDiskOperationalStatusWQL'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
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 section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).

30.8.38 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 696 describes class CIM_IndicationFilter (WQL Storage Volume OperationalStatus).

Table 696 - SMI Referenced Properties/Methods for CIM_IndicationFilter (WQL Storage Volume OperationalStatus)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	See the SystemCreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
CreationClassName		Mandatory	See the CreationClassName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
SystemName		Mandatory	See the SystemName definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
Name		Mandatory	This shall be 'SNIA:Block Services:StorageVolumeOperationalStatusWQL'.
SourceNamespace	N	Optional	Deprecated. See the SourceNamespace definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).
SourceNamespaces	N	Mandatory	Experimental. See the SourceNamespaces definition in section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.25 CIM_IndicationFilter (StaticIndicationFilter).
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 section *Storage Management Technical Specification, Part 2 Common Profiles, 1.6.0 Rev 4* 50.6.28 CIM_IndicationFilter (pre-defined).

30.8.39 CIM_LogicalDisk

Created By: Static

Modified By: Static

Deleted By: Extrinsic: StorageConfigurationService.ReturnToStoragePool
Requirement: Referenced from Volume Management - LogicalDisk is mandatory.

Table 697 describes class CIM_LogicalDisk.

Table 697 - SMI Referenced Properties/Methods for CIM_LogicalDisk

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	Opaque identifier.
ElementName		Optional	User-friendly name.
Name		Mandatory	OS Device Name.
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). In addition, the secondary OperationalStatus may be 19 (Relocating) with 2|3|15 (OK or Degraded 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.

Table 697 - SMI Referenced Properties/Methods for CIM_LogicalDisk

Properties	Flags	Requirement	Description & Notes
ExtentDiscriminator		Mandatory	Experimental. This is an array of values that shall contain 'SNIA:Allocated'.
NumExtentsMigrating		Optional	Experimental. The number of Extents in the process of migrating for this logical disk when the logical disk relocation is on going.

30.8.40 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 698 describes class CIM_MemberOfCollection (Block Services Filter Collection to FilterCollection).

Table 698 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Block Services Filter Collection to FilterCollection)

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

30.8.41 CIM_MemberOfCollection (Block Services ProfileSpecificLifecycleIndicationFilterCollection to Block Services Filters)

Experimental. This associates the Block Services ProfileSpecificLifecycleIndicationFilterCollection to the Filters defined by the Block Services Profile.

Requirement: Optional

Table 699 describes class CIM_MemberOfCollection (Block Services ProfileSpecificLifecycleIndicationFilterCollection to Block Services Filters).

Table 699 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Block Services ProfileSpecificLifecycleIndicationFilterCollection to Block Services Filters)

Properties	Flags	Requirement	Description & Notes
Collection		Mandatory	Reference to the Block Services ProfileSpecificLifecycleIndicationFilterCollection.
Member		Mandatory	Reference to the IndicationFilters defined by the Block Services Profile.

30.8.42 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 700 describes class CIM_MemberOfCollection (Predefined Filter Collection to Block Services Filters).

Table 700 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Predefined Filter Collection to Block Services Filters)

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

30.8.43 CIM_MemberOfCollection (Predefined Filter Collection to Thin Provisioning Filters)

Experimental. This associates the Thin Provisioning 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 701 describes class CIM_MemberOfCollection (Predefined Filter Collection to Thin Provisioning Filters).

Table 701 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Predefined Filter Collection to Thin Provisioning Filters)

Properties	Flags	Requirement	Description & Notes
Collection		Mandatory	Reference to the Thin Provisioning predefined FilterCollection.
Member		Mandatory	Reference to the predefined IndicationFilters of the Thin Provisioning implementation.

30.8.44 CIM_MemberOfCollection (Thin Provisioning Filter Collection to FilterCollection)

Experimental. This associates the Thin Provisioning 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 702 describes class CIM_MemberOfCollection (Thin Provisioning Filter Collection to FilterCollection).

Table 702 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Thin Provisioning Filter Collection to FilterCollection)

Properties	Flags	Requirement	Description & Notes
Collection		Mandatory	Reference to the predefined FilterCollection of the referencing profile.
Member		Mandatory	Reference to the Thin Provisioning predefined FilterCollection.

30.8.45 CIM_MemberOfCollection (Thin Provisioning ProfileSpecificAlertIndicationFilterCollection to Thin Provisioning Filters)

Experimental. This associates the Thin Provisioning ProfileSpecificAlertIndicationFilterCollection to the Filters defined by the Thin Provisioning Profile.

Requirement: Optional

Table 703 describes class CIM_MemberOfCollection (Thin Provisioning ProfileSpecificAlertIndicationFilterCollection to Thin Provisioning Filters).

Table 703 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Thin Provisioning ProfileSpecificAlertIndicationFilterCollection to Thin Provisioning Filters)

Properties	Flags	Requirement	Description & Notes
Collection		Mandatory	Reference to the Thin Provisioning ProfileSpecificAlertIndicationFilterCollection.
Member		Mandatory	Reference to the IndicationFilters defined by the Thin Provisioning Profile.

30.8.46 CIM_OwningJobElement

Conditional on support for Job Control profile.

Requirement: Support for Job Control profile.

Table 704 describes class CIM_OwningJobElement.

Table 704 - SMI Referenced Properties/Methods for CIM_OwningJobElement

Properties	Flags	Requirement	Description & Notes
OwnedElement		Mandatory	
OwningElement		Mandatory	

30.8.47 CIM_StorageCapabilities

Created By: Static
Modified By: Static

Deleted By: Static
Requirement: Mandatory

Table 705 describes class CIM_StorageCapabilities.

Table 705 - SMI Referenced Properties/Methods for CIM_StorageCapabilities

Properties	Flags	Requirement	Description & 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).
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.

Table 705 - SMI Referenced Properties/Methods for CIM_StorageCapabilities

Properties	Flags	Requirement	Description & Notes
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.
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.

30.8.48 CIM_StorageConfigurationCapabilities (Concrete)

StorageConfigurationCapabilities as defined in Block Services, with the addition of SupportedStorageElementTypes for thin pools and volumes. The class definition specializes the CIM_StorageConfigurationCapabilities 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 706 describes class CIM_StorageConfigurationCapabilities (Concrete).

Table 706 - SMI Referenced Properties/Methods for CIM_StorageConfigurationCapabilities (Concrete)

Properties	Flags	Requirement	Description & 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		Conditional	Conditional requirement: Support for the Storage Relocation profile. Lists what actions, invoked through StorageConfigurationService methods, shall not produce Concrete jobs. This version of the 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) or "17" (StorageVolume To StoragePool Relocation) or "18" (StoragePool To StoragePool Relocation) or "19" (StorageVolume To StorageExtent Relocation) or "20" (StoragePool To StorageExtent Relocation) or "21" (LogicalDisk To StorageExtent Relocation) or "22" (Multiple Storage Element Creation) or "23" (Multiple Storage Element Return).
SupportedStorageElementTypes (overridden)		Mandatory	Extended for Thin Provisioning to include 5 (ThinlyProvisionedStorageVolume), 6 (ThinlyProvisionedLogicalDisk), 7 (ThinlyProvisionedAllocatedStoragePool), 8 (ThinlyProvisionedQuotaStoragePool), or 9 (ThinlyProvisionedLimitlessStoragePool).

Table 706 - SMI Referenced Properties/Methods for CIM_StorageConfigurationCapabilities (Concrete)

Properties	Flags	Requirement	Description & Notes
SupportedAsynchronousActions		Conditional	Conditional requirement: Support for the Storage Relocation profile. Lists what actions, invoked through StorageConfigurationService methods, may produce Concrete jobs. This version of the 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) or "17" (StorageVolume To StoragePool Relocation) or "18" (StoragePool To StoragePool Relocation) or "19" (StorageVolume To StorageExtent Relocation) or "20" (StoragePool To StorageExtent Relocation) or "21" (LogicalDisk To StorageExtent Relocation) or "22" (Multiple Storage Element Creation) or "23" (Multiple Storage Element Return)..
SupportedStorageElementFeatures		Conditional	Conditional requirement: Support for the Storage Relocation profile. Lists actions supported through the invocation of StorageServiceService.CreateOrModifyElementFromStoragePool(). Matches 3|8|14|15|16|17|18 (StorageVolume Creation or LogicalDisk Creation or StorageVolume To StoragePool Relocation or StoragePool To StoragePool Relocation or StorageVolume To StorageExtent Relocation or StoragePool To StorageExtent Relocation LogicalDisk To StorageExtent Relocation).
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		Conditional	Conditional requirement: Support for the Storage Relocation profile. Indicates the intended usage or any restrictions that may have been imposed on storage pools.
ClientSettablePoolUsage		Conditional	Conditional requirement: Support for the Storage Relocation profile. Indicates the intended usage or any restrictions that may have been imposed on the usage of a client-settable storage pool.
MaximumElementCreateCount		Optional	Indicates the maximum number of elements that can be specified to be created in a single method call. If 0 or null, there is no limit.
MaximumElementDeleteCount		Conditional	Conditional requirement: Support for the Storage Relocation profile. Indicates the maximum number of elements that can be deleted in a single method call. If 0 or null, there is no limit.

Table 706 - SMI Referenced Properties/Methods for CIM_StorageConfigurationCapabilities (Concrete)

Properties	Flags	Requirement	Description & Notes
MultipleElementCreateFeatures		Optional	Enumeration indicating features offered by the multiple element create method. "2" (Single instance creation indication).
MultipleElementDeleteFeatures		Optional	Enumeration indicating features offered by the multiple element delete method. "2" (Continue on nonexistent element) or "3" (Return error on nonexistent element).
ThinProvisionedClientSettableReserve (added)		Mandatory	Experimental.
ThinProvisionedDefaultReserve (added)		Mandatory	Experimental.

30.8.49 CIM_StorageConfigurationCapabilities (Global)

StorageConfigurationCapabilities as defined in Block Services, with the addition of SupportedStorageElementTypes for thin pools and volumes. The class definition specializes the CIM_StorageConfigurationCapabilities 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 707 describes class CIM_StorageConfigurationCapabilities (Global).

Table 707 - SMI Referenced Properties/Methods for CIM_StorageConfigurationCapabilities (Global)

Properties	Flags	Requirement	Description & 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		Conditional	Conditional requirement: Support for the Storage Relocation profile. Lists what actions, invoked through StorageConfigurationService methods, shall not produce Concrete jobs.

Table 707 - SMI Referenced Properties/Methods for CIM_StorageConfigurationCapabilities (Global)

Properties	Flags	Requirement	Description & Notes
SupportedStorageElementTypes (overridden)		Mandatory	Extended for Thin Provisioning to include 5 (ThinlyProvisionedStorageVolume), 6 (ThinlyProvisionedLogicalDisk), 7 (ThinlyProvisionedAllocatedStoragePool), 8 (ThinlyProvisionedQuotaStoragePool), or 9 (ThinlyProvisionedLimitlessStoragePool).
SupportedAsynchronousActions		Conditional	Conditional requirement: Support for the Storage Relocation profile. Lists what actions, invoked through StorageConfigurationService methods, may produce Concrete jobs.
SupportedStorageElementFeatures		Conditional	Conditional requirement: Support for the Storage Relocation profile. Lists actions supported through the invocation of StorageServiceService.CreateOrModifyElementFromStoragePool(). Matches 3|5|8|9|11|12|13|14|15|16|17|18 (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 or StorageVolume To StoragePool Relocation or StoragePool To StoragePool Relocation or StorageVolume To StorageExtent Relocation or 'StoragePool To StorageExtent Relocation or LogicalDisk To StorageExtent Relocation).
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		Conditional	Conditional requirement: Support for the Storage Relocation profile. Indicates the intended usage or any restrictions that may have been imposed on storage pools.
ClientSettablePoolUsage		Conditional	Conditional requirement: Support for the Storage Relocation profile. Indicates the intended usage or any restrictions that may have been imposed on the usage of a client-settable storage pool.
MaximumElementCreateCount		Optional	Indicates the maximum number of elements that can be specified to be created in a single method call. If 0 or null, there is no limit.
MaximumElementDeleteCount		Conditional	Conditional requirement: Support for the Storage Relocation profile. Indicates the maximum number of elements that can be deleted in a single method call. If 0 or null, there is no limit.

Table 707 - SMI Referenced Properties/Methods for CIM_StorageConfigurationCapabilities (Global)

Properties	Flags	Requirement	Description & Notes
MultipleElementCreateFeatures		Optional	Enumeration indicating features offered by the multiple element create method. "2" (Single instance creation indication).
MultipleElementDeleteFeatures		Optional	Enumeration indicating features offered by the multiple element delete method. "2" (Continue on nonexistent element) or "3" (Return error on nonexistent element).
ThinProvisionedClientSettableReserve (added)		Mandatory	Experimental.
ThinProvisionedDefaultReserve (added)		Mandatory	Experimental.

30.8.50 CIM_StorageConfigurationCapabilities (Primordial)

StorageConfigurationCapabilities as defined in Block Services, with the addition of SupportedStorageElementTypes for thin pools and volumes. The class definition specializes the CIM_StorageConfigurationCapabilities 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 708 describes class CIM_StorageConfigurationCapabilities (Primordial).

Table 708 - SMI Referenced Properties/Methods for CIM_StorageConfigurationCapabilities (Primordial)

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
ElementName		Mandatory	
SupportedStoragePoolFeatures		Optional	Lists what StorageConfigurationService functionalities are implemented. Matches 2|3 (InExtents or Single InPool).

Table 708 - SMI Referenced Properties/Methods for CIM_StorageConfigurationCapabilities (Primordial)

Properties	Flags	Requirement	Description & Notes
SupportedSynchronousActions		Conditional	Conditional requirement: Support for the Storage Relocation profile. Lists what actions, invoked through StorageConfigurationService methods, shall not produce Concrete jobs. This version of the standard recognizes "2" (Storage Pool Creation), "12" (Storage Element from Element Creation) or "15" (StoragePool Usage Modification) or "17" (StorageVolume To StoragePool Relocation) or "18" (StoragePool To StoragePool Relocation) or "19" (StorageVolume To StorageExtent Relocation) or "20" (StoragePool To StorageExtent Relocation) or "21" (LogicalDisk To StorageExtent Relocation).
SupportedStorageElementTypes (overridden)		Mandatory	Extended for Thin Provisioning to include 5 (ThinlyProvisionedStorageVolume), 6 (ThinlyProvisionedLogicalDisk), 7 (ThinlyProvisionedAllocatedStoragePool), 8 (ThinlyProvisionedQuotaStoragePool), or 9 (ThinlyProvisionedLimitlessStoragePool).
SupportedAsynchronousActions		Conditional	Conditional requirement: Support for the Storage Relocation profile. Lists what actions, invoked through StorageConfigurationService methods, may produce Concrete jobs. This version of the standard recognizes "2" (Storage Pool Creation), "12" (Storage Element from Element Creation) or "15" (StoragePool Usage Modification) or "17" (StorageVolume To StoragePool Relocation) or "18" (StoragePool To StoragePool Relocation) or "19" (StorageVolume To StorageExtent Relocation) or "20" (StoragePool To StorageExtent Relocation) or "21" (LogicalDisk To StorageExtent Relocation) or "22" (Multiple Storage Element Creation) or "23" (Multiple Storage Element Return).
SupportedStorageElementFeatures		Conditional	Conditional requirement: Support for the Storage Relocation profile. Lists actions supported through the invocation of StorageServiceService.CreateOrModifyElementFromStoragePool(). This version of the 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) or 14 (StorageVolume To StoragePool Relocation) or 15 (StoragePool To StoragePool Relocation) or 16 (StorageVolume To StorageExtent Relocation) or 17 (StoragePool To StorageExtent Relocation) or 18 (LogicalDisk To StorageExtent Relocation).
SupportedStorageElementUsage		Optional	For Primordial StorageConfigurationCapabilities, this shall be NULL.

Table 708 - SMI Referenced Properties/Methods for CIM_StorageConfigurationCapabilities (Primordial)

Properties	Flags	Requirement	Description & Notes
ClientSettableElementUsage		Optional	For Primordial StorageConfigurationCapabilities, this shall be NULL.
SupportedStoragePoolUsage		Conditional	Conditional requirement: Support for the Storage Relocation profile. Indicates the intended usage or any restrictions that may have been imposed on storage pools.
ClientSettablePoolUsage		Conditional	Conditional requirement: Support for the Storage Relocation profile. Indicates the intended usage or any restrictions that may have been imposed on the usage of a client-settable storage pool.
ThinProvisionedClientSettableReserve (added)		Mandatory	Experimental.
ThinProvisionedDefaultReserve (added)		Mandatory	Experimental.

30.8.51 CIM_StorageConfigurationService

StorageConfigurationService as defined in Block Services, adding thin provisioning values to the ElementType parameter. The class definition specializes the CIM_StorageConfigurationService 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 709 describes class CIM_StorageConfigurationService.

Table 709 - SMI Referenced Properties/Methods for CIM_StorageConfigurationService

Properties	Flags	Requirement	Description & 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.

Table 709 - SMI Referenced Properties/Methods for CIM_StorageConfigurationService

Properties	Flags	Requirement	Description & Notes
CreateOrModifyElementFromStoragePool() (overridden)		Mandatory	Expanded ElementType parameter.
CreateElementsFromStoragePool()		Optional	Experimental. Create one or more storage elements. A job may be created as well.
CreateOrModifyElementFromElements() (overridden)		Mandatory	Expanded ElementType parameter.
ReturnToStoragePool()		Mandatory	Release the capacity represented by this storage element back to the Pool.
ReturnElementsToStoragePool()		Optional	Experimental. Release the capacity represented by one or more storage elements 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.

30.8.52 CIM_StoragePool (Concrete)

Concrete StoragePool as defined in Block Services with the addition of SpaceLimit, SpaceLimitDetermination, and ThinProvisionMetaDataSpace properties. The class definition specializes the CIM_StoragePool 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: StorageConfigurationService.CreateOrModifyStoragePool

Modified By: Extrinsic: StorageConfigurationService.CreateOrModifyStoragePool

Deleted By: Extrinsic: StorageConfigurationService.DeleteStoragePool

Requirement: Mandatory

Table 710 describes class CIM_StoragePool (Concrete).

Table 710 - SMI Referenced Properties/Methods for CIM_StoragePool (Concrete)

Properties	Flags	Requirement	Description & Notes
Primordial (overridden)		Mandatory	Shall be false.
InstanceID		Mandatory	
ElementName		Optional	
PoolID		Mandatory	A unique name in the context of this system that identifies this Pool.

Table 710 - SMI Referenced Properties/Methods for CIM_StoragePool (Concrete)

Properties	Flags	Requirement	Description & Notes
OperationalStatus		Conditional	Experimental. Conditional requirement: Support for the Storage Relocation profile. Value shall be 2|3|6|15 (OK or Degraded or Error or Dormant). In addition, the secondary OperationalStatus may be 19 (Relocating) with 2|3|15 (OK or Degraded or Dormant).
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.
CapacityInMigratingSource		Optional	Experimental. The total capacity of extents migrating out from this storage pool.
CapacityInMigratingTarget		Optional	Experimental. The total capacity of extents migrating into this storage pool.
SpaceLimit (added)		Mandatory	Experimental. The capacity of the storage allocated to the pool when SpaceLimitDetermination has the value 3 (Quota) or 4 (Limitless) or set to the value of TotalManagedSpace if SpaceLimitDetermination has the value 2 (Allocated).
SpaceLimitDetermination (added)		Mandatory	Experimental. The SpaceLimitDetermination property of StoragePool defines the approach associated with the pool for determining capacity information for the pool.
ThinProvisionMetaDataSpace (added)		Optional	Experimental. The size of metadata consumed by this storage pool.
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.

30.8.53 CIM_StoragePool (Empty)

Empty StoragePool as defined in Block Services with the addition of SpaceLimit, SpaceLimitDetermination, and ThinProvisionMetaDataSpace properties. The class definition specializes the CIM_StoragePool 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: StorageConfigurationService.CreateOrModifyStoragePool

Modified By: Extrinsic: StorageConfigurationService.CreateOrModifyStoragePool

Deleted By: Extrinsic: StorageConfigurationService.DeleteStoragePool

Requirement: Mandatory

Table 711 describes class CIM_StoragePool (Empty).

Table 711 - SMI Referenced Properties/Methods for CIM_StoragePool (Empty)

Properties	Flags	Requirement	Description & 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	
OperationalStatus		Conditional	Experimental. Conditional requirement: Support for the Storage Relocation profile. Value shall be 2|3|6|15 (OK or Degraded or Error or Dormant). In addition, the secondary OperationalStatus may be 19 (Relocating) with 2|3|15 (OK or Degraded or Dormant).
TotalManagedSpace		Mandatory	This shall be 0 for an empty StoragePool.
RemainingManaged Space		Mandatory	
Usage		Optional	
OtherUsageDescripti on		Optional	
ClientSettableUsage		Optional	
CapacityInMigratingS ource		Optional	Experimental. The total capacity of extents migrating out from this storage pool.
CapacityInMigratingT arget		Optional	Experimental. The total capacity of extents migrating into this storage pool.
SpaceLimit (added)		Mandatory	Experimental. The capacity of the storage allocated to the pool when SpaceLimitDetermination has the value 3 (Quota) or 4 (Limitless) or set to the value of TotalManagedSpace if SpaceLimitDetermination has the value 2 (Allocated).

Table 711 - SMI Referenced Properties/Methods for CIM_StoragePool (Empty)

Properties	Flags	Requirement	Description & Notes
SpaceLimitDetermination (added)		Mandatory	Experimental. The SpaceLimitDetermination property of StoragePool defines the approach associated with the pool for determining capacity information for the pool.
ThinProvisionMetaDataSpace (added)		Optional	Experimental. The size of metadata consumed by this storage pool.
GetSupportedSizes()		Conditional	Conditional requirement: Support for StorageConfigurationService.
GetSupportedSizeRange()		Conditional	Conditional requirement: Support for StorageConfigurationService.
GetAvailableExtents()		Optional	

30.8.54 CIM_StoragePool (Primordial)

Primordial StoragePool as defined in Block Services with the addition of SpaceLimit, SpaceLimitDetermination, and ThinProvisionMetaDataSpace properties. The class definition specializes the CIM_StoragePool 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 712 describes class CIM_StoragePool (Primordial).

Table 712 - SMI Referenced Properties/Methods for CIM_StoragePool (Primordial)

Properties	Flags	Requirement	Description & Notes
Primordial (overridden)		Mandatory	Shall be true.
InstanceID		Mandatory	
ElementName		Optional	
PoolID		Mandatory	A unique name in the context of this system that identifies this Pool.
OperationalStatus		Conditional	Experimental. Conditional requirement: Support for the Storage Relocation profile. Value shall be 2|3|6|15 (OK or Degraded or Error or Dormant). In addition, the secondary OperationalStatus may be 19 (Relocating) with 2|3|15 (OK or Degraded or Dormant).
TotalManagedSpace		Mandatory	
RemainingManagedSpace		Mandatory	

Table 712 - SMI Referenced Properties/Methods for CIM_StoragePool (Primordial)

Properties	Flags	Requirement	Description & Notes
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.
CapacityInMigratingSource		Optional	Experimental. The total capacity of extents migrating out from this storage pool.
CapacityInMigratingTarget		Optional	Experimental. The total capacity of extents migrating into this storage pool.
SpaceLimit (added)		Mandatory	Experimental. The capacity of the storage allocated to the pool when SpaceLimitDetermination has the value 3 (Quota) or 4 (Limitless) or set to the value of TotalManagedSpace if SpaceLimitDetermination has the value 2 (Allocated).
SpaceLimitDetermination (added)		Mandatory	Experimental. The SpaceLimitDetermination property of StoragePool defines the approach associated with the pool for determining capacity information for the pool.
ThinProvisionMetaDataSpace (added)		Optional	Experimental. The size of metadata consumed by this storage pool.
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.

30.8.55 CIM_StorageSetting

StorageSetting as defined in Block Services with the addition of and Thin Provisioning properties. 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 713 describes class CIM_StorageSetting.

Table 713 - SMI Referenced Properties/Methods for CIM_StorageSetting

Properties	Flags	Requirement	Description & 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.

Table 713 - SMI Referenced Properties/Methods for CIM_StorageSetting

Properties	Flags	Requirement	Description & Notes
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.
ThinProvisionedPoolType (added)		Mandatory	Experimental. This property is needed when the Setting is used as goal in CreateOrModify... but is not needed when the Setting class is associated to a pool or volume.
ThinProvisionedInitialReserve (added)		Mandatory	

30.8.56 CIM_StorageSettingWithHints

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

Table 714 describes class CIM_StorageSettingWithHints.

Table 714 - SMI Referenced Properties/Methods for CIM_StorageSettingWithHints

Properties	Flags	Requirement	Description & 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	

Table 714 - SMI Referenced Properties/Methods for CIM_StorageSettingWithHints

Properties	Flags	Requirement	Description & Notes
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	
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.

Table 714 - SMI Referenced Properties/Methods for CIM_StorageSettingWithHints

Properties	Flags	Requirement	Description & Notes
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	

30.8.57 CIM_StorageSettingsAssociatedToCapabilities

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

Table 715 describes class CIM_StorageSettingsAssociatedToCapabilities.

Table 715 - SMI Referenced Properties/Methods for CIM_StorageSettingsAssociatedToCapabilities

Properties	Flags	Requirement	Description & 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.

30.8.58 CIM_StorageSettingsGeneratedFromCapabilities

Created By: Extrinsic: CreateSetting
Modified By: Static
Deleted By: Static
Requirement: Support for StorageConfigurationService.

Table 716 describes class CIM_StorageSettingsGeneratedFromCapabilities.

Table 716 - SMI Referenced Properties/Methods for CIM_StorageSettingsGeneratedFromCapabilities

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

30.8.59 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 717 describes class CIM_StorageVolume.

Table 717 - SMI Referenced Properties/Methods for CIM_StorageVolume

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

Table 717 - SMI Referenced Properties/Methods for CIM_StorageVolume

Properties	Flags	Requirement	Description & 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). In addition, the secondary OperationalStatus may be 19 (Relocating) with 2|3|15 (OK or Degraded 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".

Table 717 - SMI Referenced Properties/Methods for CIM_StorageVolume

Properties	Flags	Requirement	Description & Notes
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'.
CanDelete		Optional	Experimental. Indicates if the volume is able to be deleted by a client application.
NumExtentsMigrating		Optional	Experimental. The number of Extents in the process of migrating for this storage volume when the volume relocation is on going.

30.8.60 CIM_SystemDevice (System to StorageVolume or LogicalDisk)

Created By: Static
Modified By: Static
Deleted By: Extrinsic: StorageConfigurationService.ReturnToStoragePool
Requirement: Mandatory

Table 718 describes class CIM_SystemDevice (System to StorageVolume or LogicalDisk).

Table 718 - SMI Referenced Properties/Methods for CIM_SystemDevice (System to StorageVolume or LogicalDisk)

Properties	Flags	Requirement	Description & Notes
PartComponent		Mandatory	
GroupComponent		Mandatory	

EXPERIMENTAL

Clause 31: Automated Storage Tiering Profile

EXPERIMENTAL

31.1 Description

31.1.1 Synopsis

Profile Name: Automated Storage Tiering

Version: 1.6.0

Organization: SNIA

CIM schema version: 2.29.1

Central Class: TierService

Scoping Class: ComputerSystem

31.1.2 Supported Profiles, Subprofiles, and Packages

Profile Name: Automated Storage Tiering (Component Profile)

Version: 1.6.0

Organization: SNIA

CIM Schema Version: 2.29.1

Table 719 describes the related profiles for Automated Storage Tiering.

Table 719 - Related Profiles for Automated Storage Tiering

Profile Name	Organization	Version	Requirement	Description
Block Services	SNIA	1.6.0	Mandatory	
Pools from Volumes	SNIA	1.4.0	Optional	
Job Control	SNIA	1.5.0	Optional	
Extent Composition	SNIA	1.6.0	Optional	

31.1.3 Overview

The storage industry offers a range of storage products with varying performance characteristics. For example, Solid State Drive (SSD), Fibre Channel (FC), Serial ATA (SATA), etc.

Various storage products can be configured into different tiers of storage. To enable storage arrays to provide faster access to data based on how frequently the data is accessed, it is desirable to monitor the frequency of data access enabling the optimized placement of the data in the appropriate storage tier.

The Automated Storage Tiering profile, a component profile, includes classes and methods to expose the storage tiering feature of the storage array. The storage tiers may be created by the storage array based on the performance characteristics of the underlying hardware and the quality of service associated with that hardware.

Alternatively, SMI-S clients may have the ability to create the storage tiers based on various requirements such as disk drive technology, quality of service, etc.

Once the storage tiers have been identified, the storage array will monitor the data usage of various storage elements (e.g., volumes) and “automatically” move the data to the appropriate storage tier in order to optimize the response time of the applications using that data. At the same time, as the data in a high performing storage tiers is used less often, the storage array will move that data to a less performing storage tier.

Storage tiering may be a fully automated feature managed by the storage array. Deciding when and under what performance criteria to move storage volumes between storage tiers may also be based on policies and schedules provided by the storage administrators. In some cases, a storage array may have well established default policies for optimum performance; however, it may also allow the storage administrators to fine tune the policies to better meet the requirements of the applications accessing the data.

Throughout this profile, there are specific references to class properties and methods pertaining to each section. Refer to 31.4 "CIM Elements" for a complete list of all properties and methods, including their description.

31.1.4 Key Components

Table 720 shows a list of key classes used by the Automated Storage Tiering.

Clients should refer to 31.3 "Client Considerations and Recipes" for a list of steps to follow to utilize the storage tiering service.

Table 720 - Key Classes

Class Name	Notes
TierService	The main class for Automated Storage Tiering. It contains methods for manipulating storage tiers.
TierServiceCapabilities	Contains a set of properties and methods that describe the capabilities of automated storage tiering.
StorageTier	Represents a collection of storage objects belong to a tier.
AssociatedElementTier	Associates an element to a storage tier.
TierDomain	Represents a set of storage tiers that have a common scope and management domain.

31.1.5 Automated Storage Tiering Discovery

Figure 161 depicts the Automated Storage Tiering discovery instance diagram.

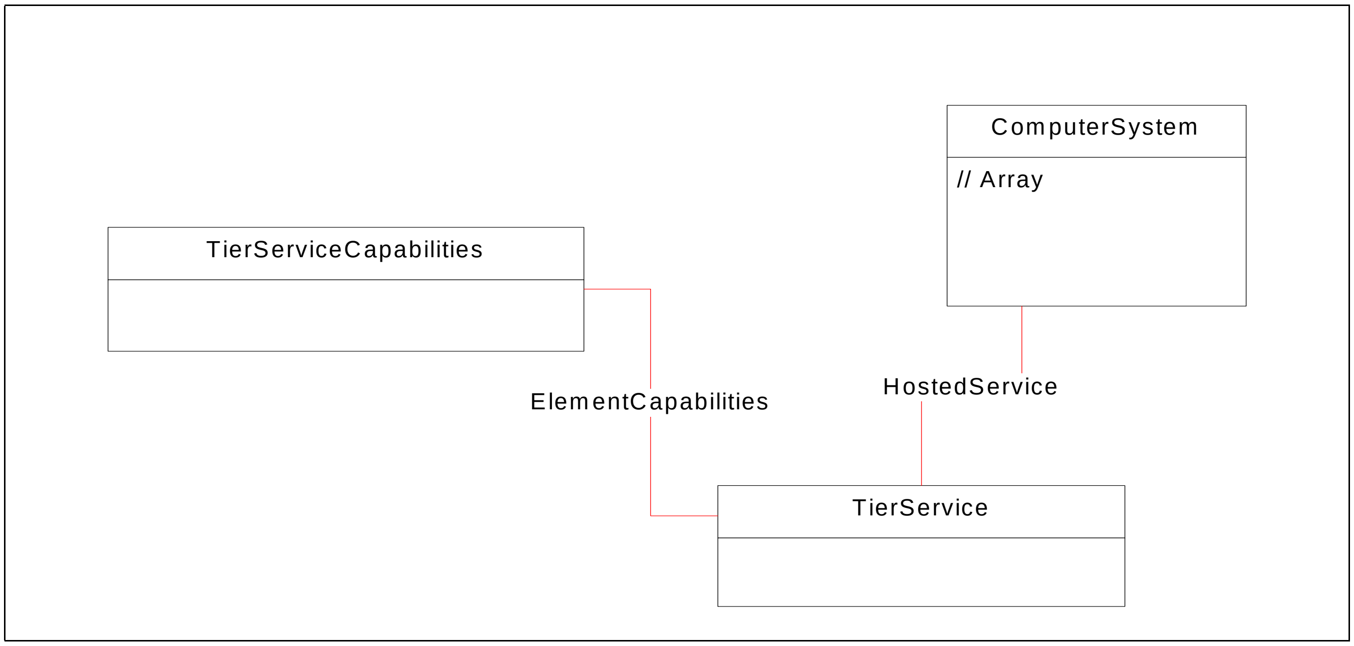

Figure 161 - Automated Storage Tiering Discovery

As shown in Figure 162, the single instance of the class TierService and its methods provide the mechanism for creating and managing storage tiers if they are not automatically created by the storage array.

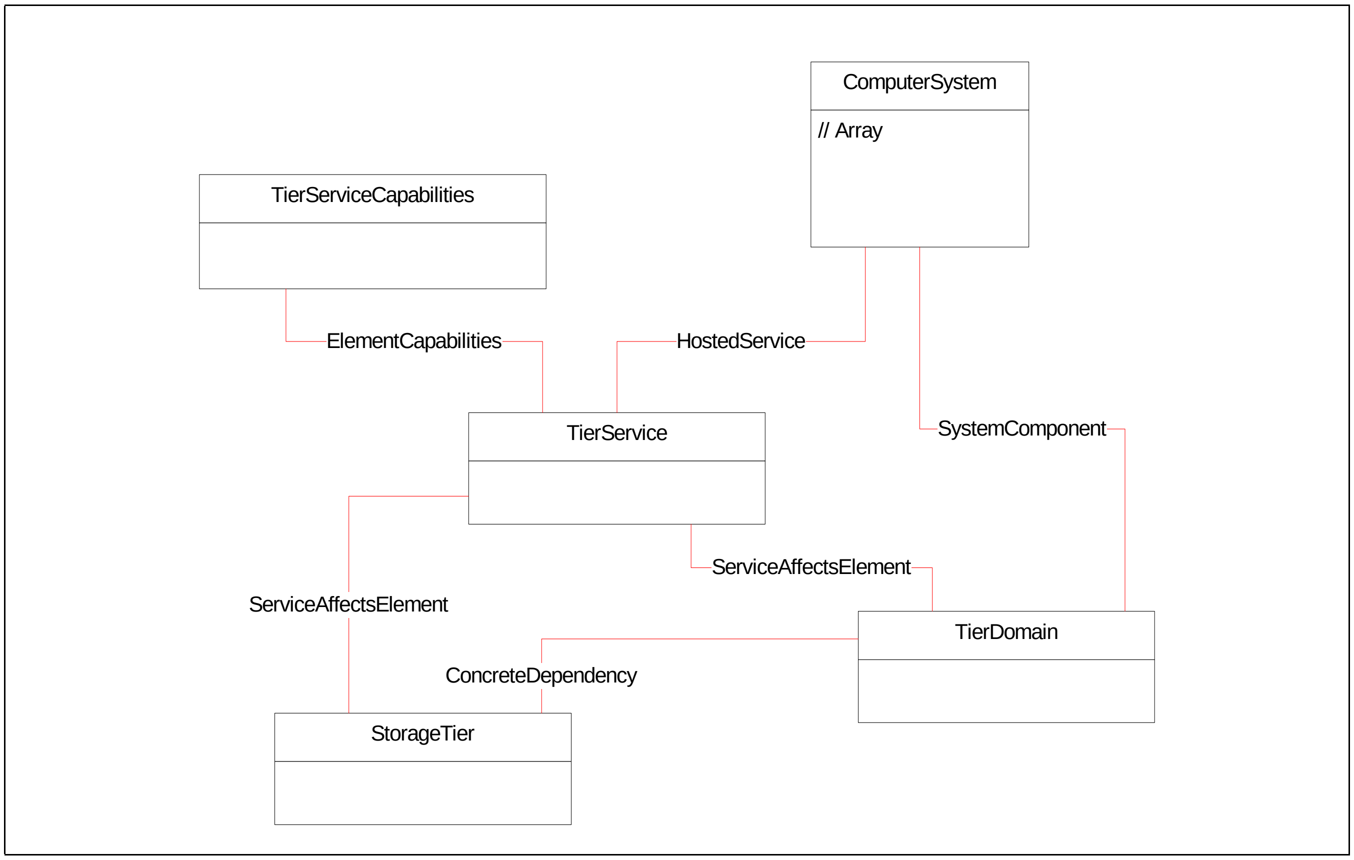

Figure 162 - Additional Automated Storage Tiering Components

31.1.6 Storage Tiers

A storage tier is a collection of storage objects that have the same performance characteristics and/or Quality of Service (QoS).

An implementation offering support for storage tiering, may create the storage tiers automatically or allow clients to invoke the appropriate methods in the TierService class to create the storage tiers.

Figure 163 shows two storage tiers -- one based on SSD (Solid State Drives) disk drives, and one based on SATA (Serial ATA) disk drives.

Storage volumes can be automatically relocated from one storage tier to another storage tier in order to improve access to data stored on the storage volumes.

A storage tier with a lower value RelativePerformanceOrder has a “better” performance characteristics. On systems that create storage tiers automatically, the system assigns an appropriate RelativePerformanceOrder value to the storage tier. On systems that clients request a storage tier to be created (by invoking the method CreateStorageTier, see section 31.2.1), clients can specify the RelativePerformanceOrder for the storage tier, as well as the acceptable minimum and maximum values for the RelativePerformanceOrder for the requested storage tier.

The method GetStorageTierCandidateObjects (see section 31.2.8) allows clients to locate the appropriate storage objects to form a storage tier.

Storage volumes with a null or 0 StorageVolume.StorageTieringSelection property are not subject to storage tiering. The property StorageVolume.StorageTieringSelection has the following possible values:

- 0: The StorageVolume is not subject to storage tiering
- 1: “Use RelativePerformanceOrder” -- Use the property StorageVolume.RelativePerformanceOrder to locate an appropriate storage tier for this storage volume.
- 2: “Use RelativePerformanceOrderSet” -- which indicates the storage volume can only be associated with storage tiers that have a RelativePerformanceOrder value included in this set.

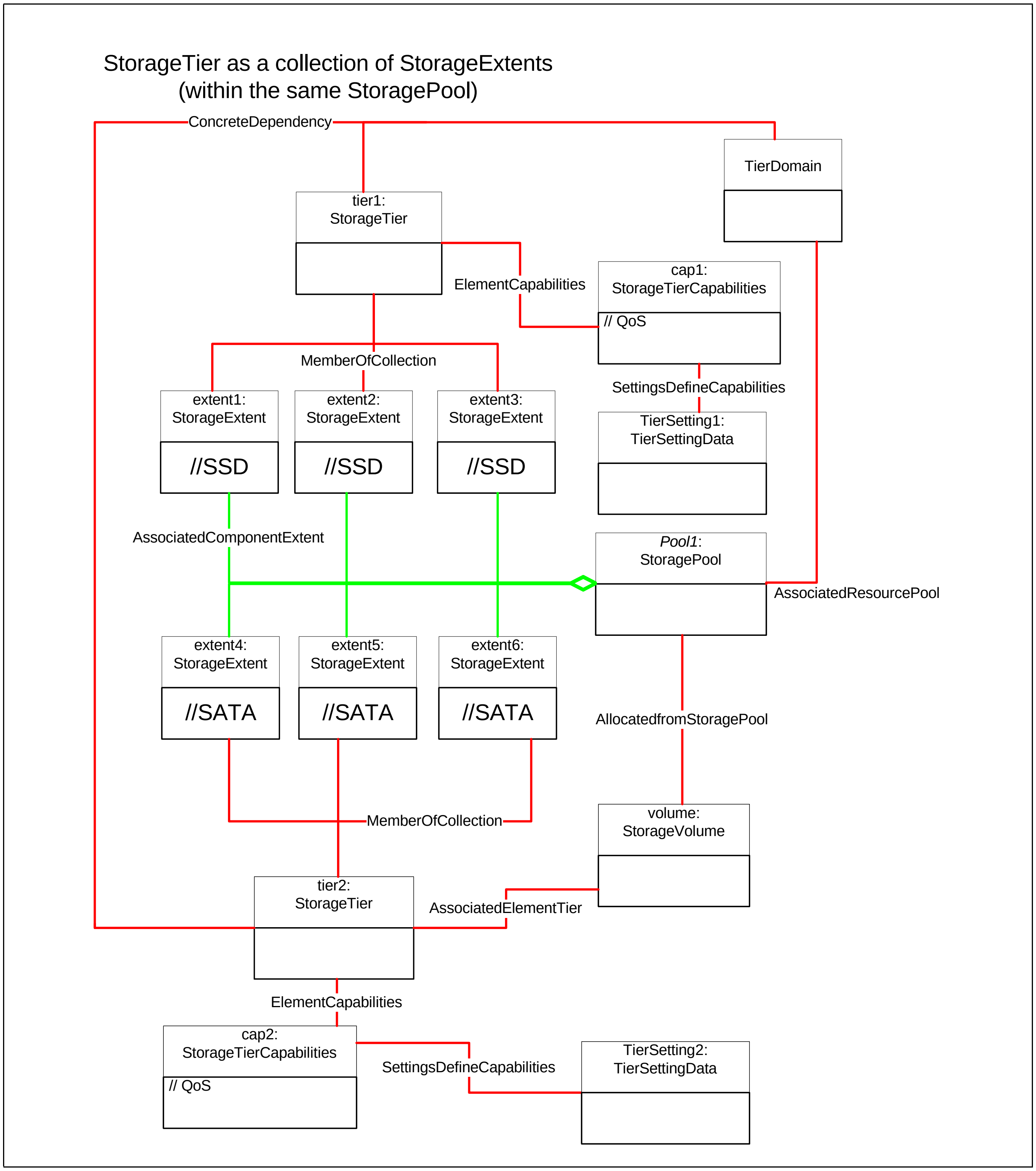

Figure 163 - Storage Tiering Model

A storage tier may be comprised of storage objects with the same performance characteristics (and QoS) from different storage pools, as shown in Figure 164.

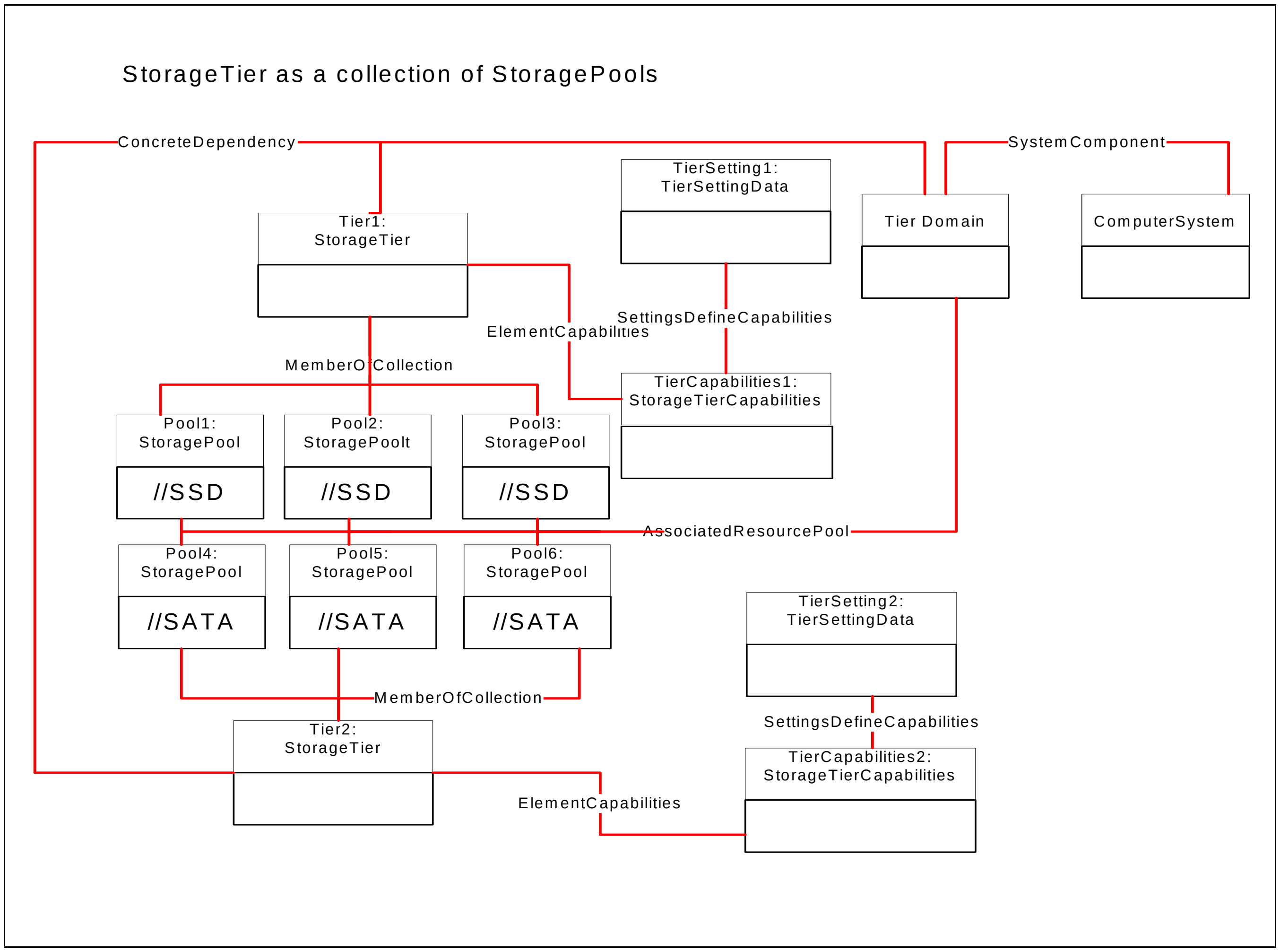

Figure 164 - Storage Tiering Model based on different pools

A storage tier may consist of StorageVolumes allocated from different storage pools with similar performance characteristics (and/or QoS). In this configuration, StorageVolumes form a StoragePool (per Pools from Volumes profile). Figure 165 shows such configuration.

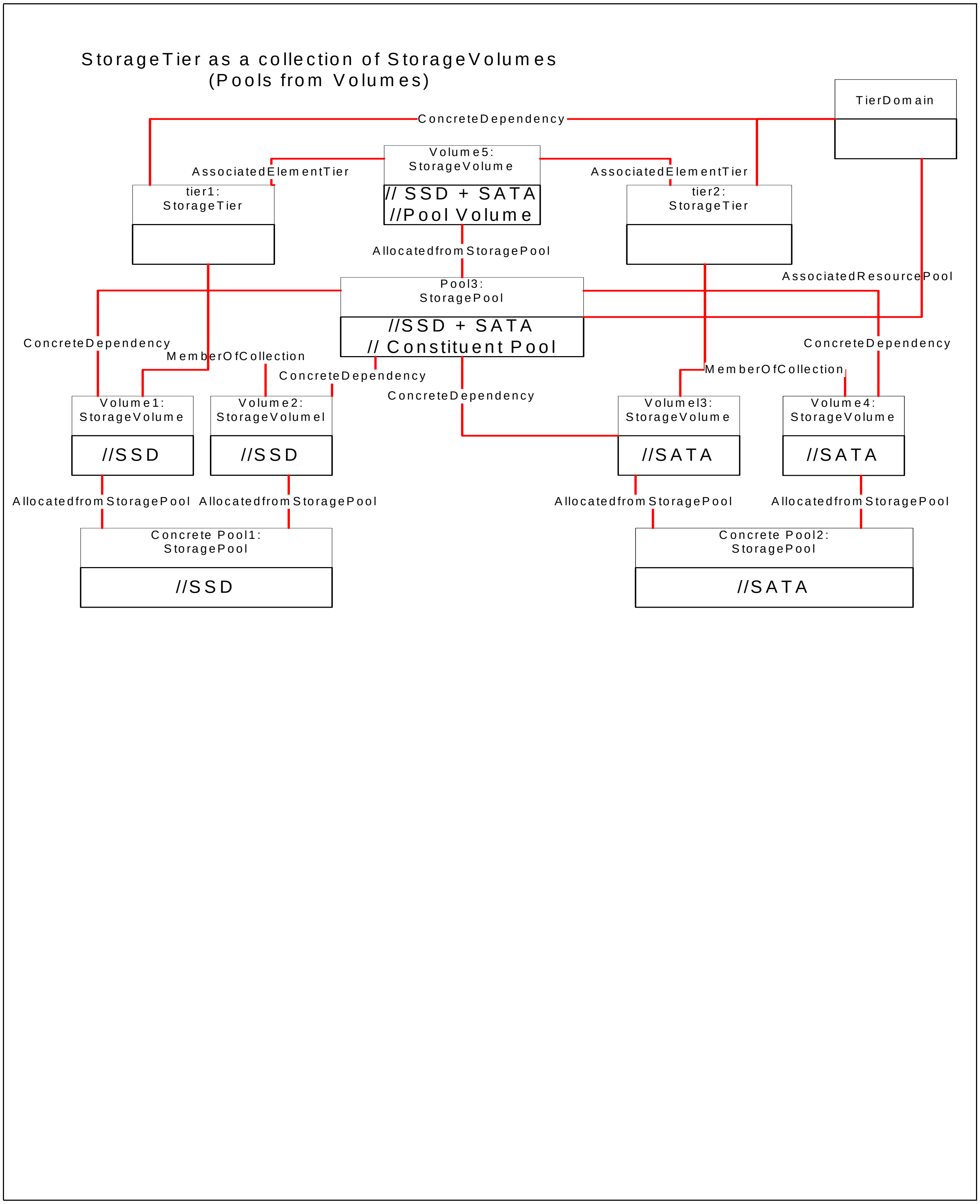

Figure 165 - Storage Tiering based on StorageVolumes forming a StoragePool

Storage tiers can be based on the Quality of Service (QoS) offered by the underlying storage pools. Figure 166 shows two different storage tiers--one that is “highly available,” and another that has “low availability.” The storage capabilities associated to each storage tier indicate the range of redundancies offered by the corresponding storage tier.

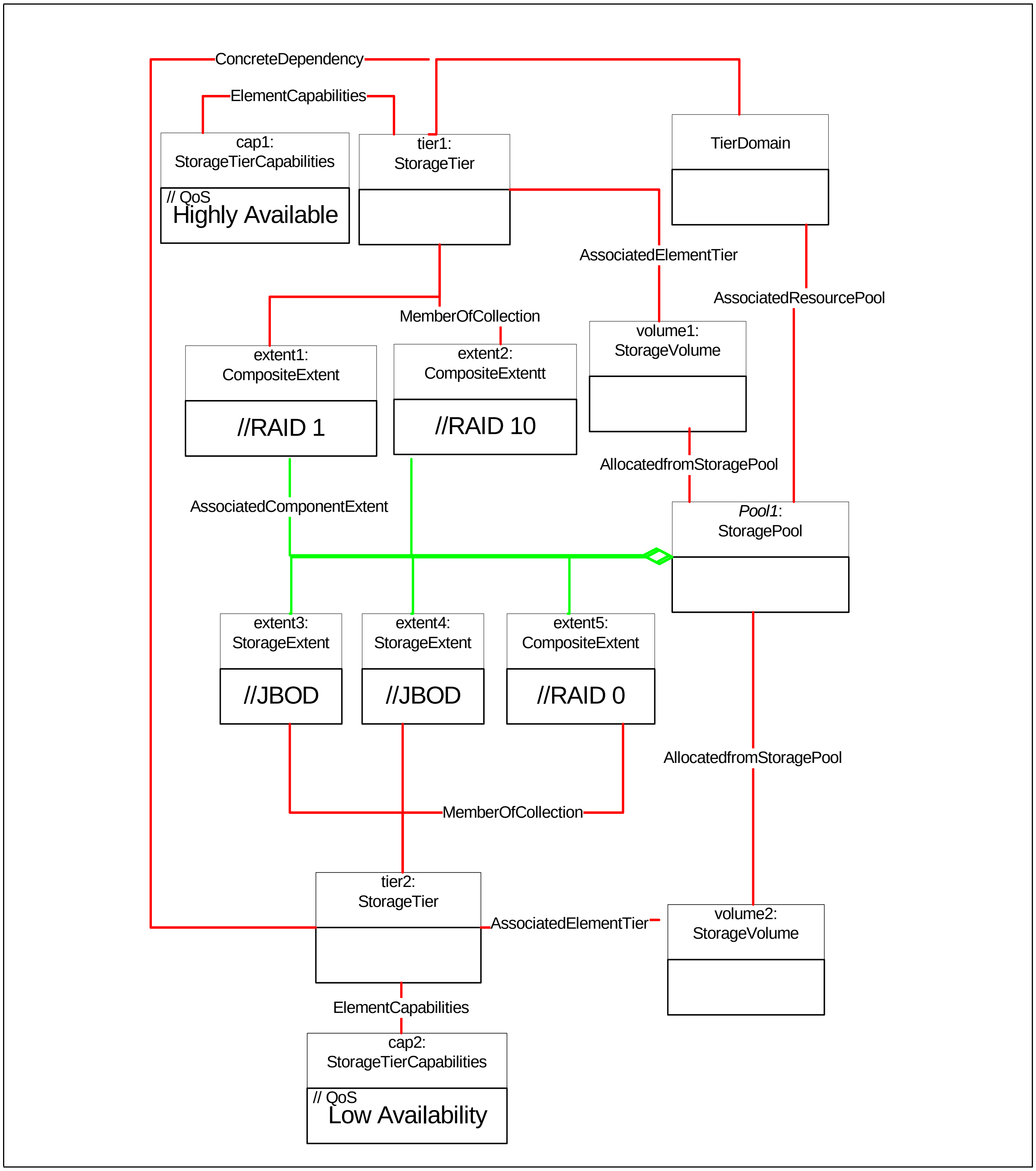

Figure 166 - Storage tiers based on QoS

31.1.7 StorageTier and StoragePool

There is a correlation between storage tiers and storage pools since storage volumes allocated from a storage pool also have an association to one or more storage tiers. However, a StorageTier represents a *collection* of resources, which are identified as a tier; whereas, a StoragePool represents a *pool* of resources, which contain the elements such as a StorageVolume.

For example it is possible to divide up a storage pool based on disk drive technologies (and/or protection) into storage tiers--for example, all SSD drives into tier1, FC drives into tier2, and SATA drives into tier3.

Alternatively, it is possible to create a storage tier from one or more storage pools--for example, one or more storage pools that are based on SSD drives can be grouped into tier1, one or more storage pools that are based on FC drives can be grouped into tier2, etc.

31.1.8 TierDomain

A tier domain is a collection of storage tiers. The storage tiers belonging to a tier domain are associated to the TierDomain via the ConcreteDependency association. TierDomains are associated to the top level ComputerSystem via the SystemComponent association.

The underlying storage pools are associated to one or more tier domains via the AssociatedResourcePool association.

The storage elements subject to storage tiering are restricted to move between storage tiers in the same tier domain. In Figure 167, there are two tier domains. In this configuration, for example, the storage tiering subsystem may move a storage element allocated from storage pool 3 to storage pool 1, which is associated to tier 1.

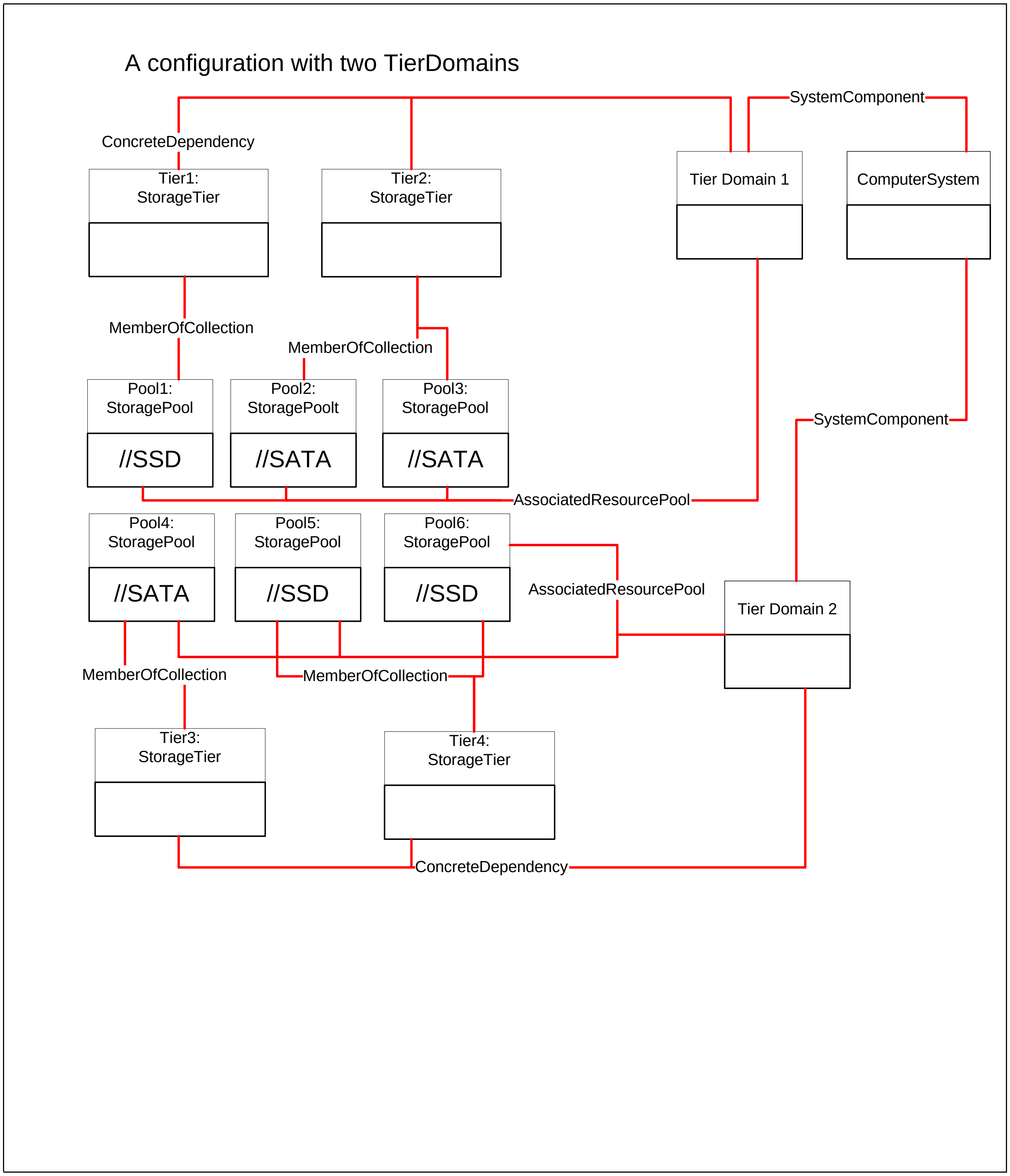

Figure 167 - Two TierDomain Configuration

31.1.9 Support for Sub-LUN tiering

An implementation may support storage tiering for Sub-LUNs. Such support involves automatic movement of only certain, "heavily used", extents of a storage volume to more performant storage tiers -- as opposed to moving all the extents of the storage volume. Figure 168 shows a storage volume that is associated to two different storage tiers. See the TierServiceCapabilities in section 31.1.10.

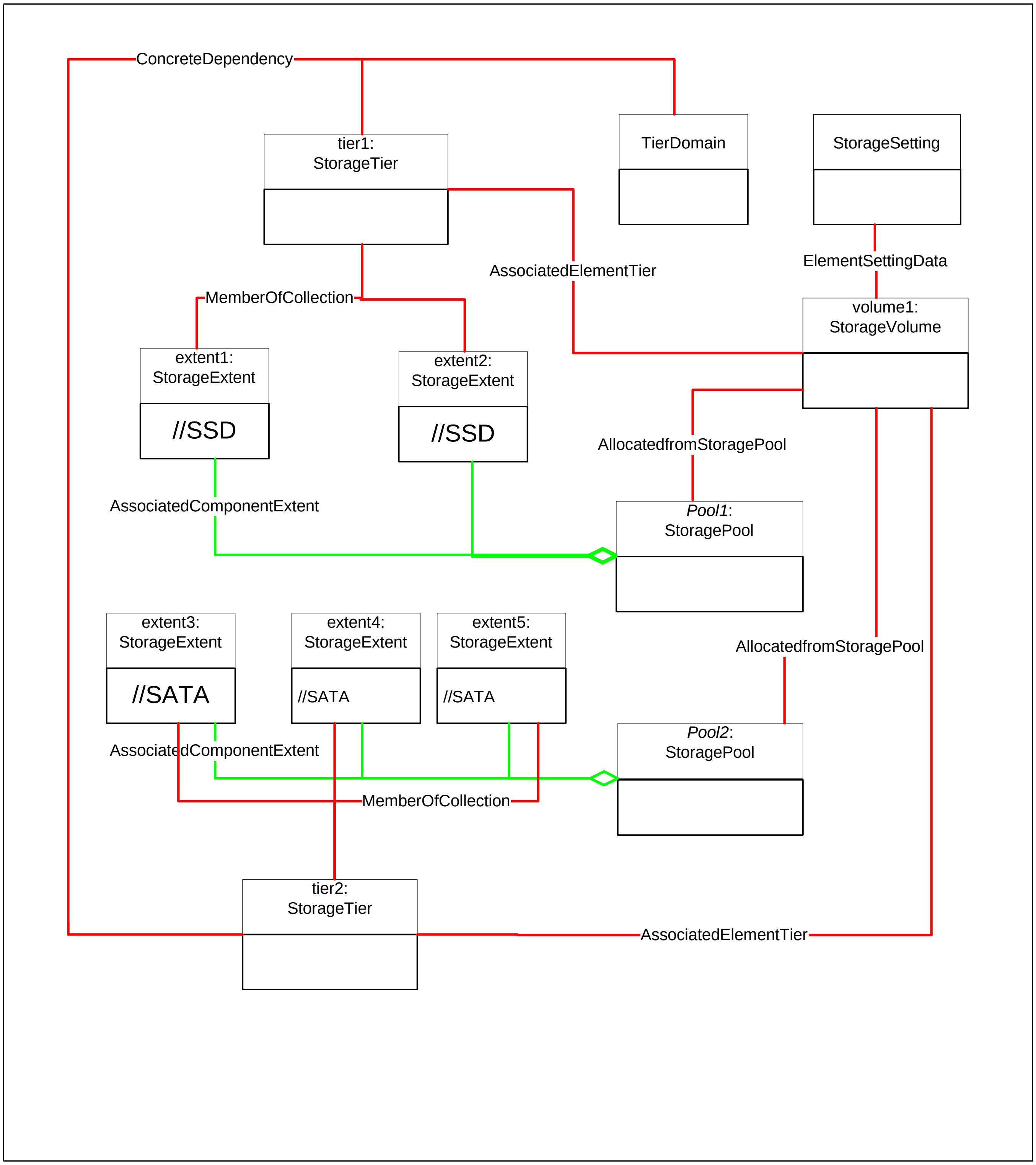

Figure 168 - A volume associated to two storage tiers

31.1.10 Storage Tiering Capabilities

The single instance of the class TierServiceCapabilities describes various capabilities of the storage tiering feature. Clients should examine the TierServiceCapabilities instance to determine the specific capabilities of the storage tiering implementation.

The property TierServiceCapabilities.SupportedFeatures is an array indicating the supported features of the storage tiering service. Table 721 show the possible values for this property.

Table 721 - SupportedFeatures

Feature
System Creates StorageTiers
Client Can Create StorageTiers
System Creates TierDomains
Client Can Create TierDomains
Supports SubLUN

The property TierServiceCapabilities.SupportedTierFeatures is an array indicating the supported tier features.

Table 722 - SupportedTierFeatures

Storage Tier Features
StorageTiers Based On Performance Only
StorageTiers Based On QoS Only
StorageTiers Based On Performance and QoS
StorageTiers Based On Other Characteristics
StorageTiers Can Overlap
StorageTiers Can Be Empty
StorageTiers In Multiple TierDomains

31.2 Methods of the Profile

The Automated Storage Tiering Profile has a number of extrinsic methods for storage tier management.

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.

Furthermore, the Profile relies on a number of intrinsic methods such as ModifyInstance for modifying properties such as ElementName.

- 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 723 summarizes the extrinsic methods (class TierService).

Table 723 - Extrinsic Methods

Method	Described in
CreateStorageTier()	See 31.2.1
DeleteStorageTier()	See 31.2.2
AddToStorageTier()	See 31.2.3
RemoveFromStorageTier()	See 31.2.4
CreateTierDomain()	See 31.2.5
DeleteTierDomain()	See 31.2.6
ModifyStorageTierDomainAssociation()	See 31.2.7
GetStorageTierCandidateObjects()	See 31.2.8

31.2.1 CreateStorageTier

```
uint32 TierService.CreateStorageTier(
         [IN, Description (
             "A end user relevant name. If null, then a "
             "system supplied default name may be used.")]
      string ElementName,
         [IN, Description (
             "List of elements to use to create a storage "
             "tier. " )]
      CIM_LogicalElement REF Members[],
         [IN, Description (
             "If provided, it overrides the default tiering "
             "setting data that is used. " ),
          EmbeddedInstance ( "CIM_TierSettingData" )]
      string TierSettingData,
         [IN, Description (
             "The Quality of Service per the properties "
             "of supplied StorageSetting." ) ]
      CIM_ManagedElement REF Goal,
         [IN, Description (
             "Array of references to CIM_StoragePool "
             "instances. " )]
      CIM_AdminDomain REF TierDomain[],
         [IN ( false ), OUT, Description (
             "Reference to the job (may be Null if job is "
             "completed).")]
      CIM_ConcreteJob REF Job,
         [IN ( false ), OUT, Description (
             "Reference to the created StorageExtentTier." )]
      CIM_StorageTier REF StorageTier );
```

This method allows a client to create a storage tier based on the supplied information. For example, it is possible to create a storage tier based on the storage extents of a storage pool that are utilizing the SSD technology. In this case, the parameter Members[0] is set to reference the desired storage pool, TierSettingData.Technology property set to SSD, and TierSettingData.StorageObjectType to ComponentsOfStoragePool.

Basically, the more parameters specified, the narrower the criteria for selecting the storage objects comprising a storage tier.

The parameters are as follows:

- ElementName: An end-user relevant name for the element being created. If null, then a system supplied name may be used. The value will be stored in the 'ElementName' property for the created element.
- Members: List of elements (e.g., StorageExtents) to use to create a storage tier. If null, locate storage object from other parameters -- for example, from TierSettingData.
- TierSettingData: Its properties are used to populate the applicable properties of the created storage tier -- for example, RelativePerformanceOrder. Additionally, its other properties are used to narrow the selection criteria for the locating appropriate storage objects for the tier -- for example, Technology, with values such as SSD, Mixed, etc.
- Goal: The definition for the StorageSetting to locate storage objects with the desired Quality of Service.
- TierDomain: As an input, the created storage tier will be associated to this array of TierDomains. If null, the implementation may create a TierDomain, or associate the created storage tier to existing TierDomains. As an output, it will contain references to the TierDomains that the system decided to use or create.
- 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).
- StorageTier: Refers to the created storage tier. If a job is created, this parameter may be null. Use the AffectedJobElement association to locate the created storage tier once the job completes.

Method Notes:

- Any required associations are created in addition to the instance of the StorageTier.

31.2.2 DeleteStorageTier

```
uint32 TierService.DeleteStorageTier(
      [IN, EmbeddedInstance ( "CIM_TierSettingData" )]
            string TierSettingData,
      [OUT] CIM_ConcreteJob REF Job,
      [IN]  CIM_StorageTier REF StorageTier );
```

This method allows a client to delete a storage tier. All associations to the deleted storage tier are also removed as part of the action.

31.2.3 AddToStorageTier

```
uint32 TierService.AddToStorageTier(
         [IN]  CIM_LogicalElement REF Members[],
         [IN, EmbeddedInstance ( "CIM_TierSettingData" )]
```

```
                        string TierSettingData,
                  [IN]  CIM_ManagedElement REF Goal,
                  [IN]  CIM_ResourcePool REF InPools[],
                  [OUT] CIM_ConcreteJob REF Job,
                  [IN]  CIM_StorageTier REF StorageTier );
```

This method allows a client to add storage objects to an existing storage tier.

31.2.4 RemoveFromStorageTier

```
         uint32 TierService.RemoveFromStorageTier(
                  [IN]  CIM_LogicalElement REF Members[],
                  [OUT] CIM_ConcreteJob REF Job,
                  [IN]  CIM_StorageTier REF StorageTier );
```

This method allows a client to remove storage objects from an existing storage tier. If empty storage tiers are not supported by the implementation, deleting all members will delete the storage tier.

31.2.5 CreateTierDomain

```
            uint32 TierService.CreateTierDomain(
                  [IN]  string ElementName,
                  [IN, EmbeddedInstance ( "CIM_TierSettingData" )]
                        string TierSettingData,
                  [OUT] CIM_ConcreteJob REF Job,
                  [OUT] CIM_AdminDomain REF TierDomain );
```

This method allows a client to create a new TierDomain.

31.2.6 DeleteTierDomain

```
            uint32 TierService.DeleteTierDomain(
                  [IN, EmbeddedInstance ( "CIM_TierSettingData" )]
                      string TierSettingData,
                  [OUT] CIM_ConcreteJob REF Job,
                  [IN]  Boolean Force,
                  [IN]  CIM_AdminDomain REF TierDomain );
```

This method allows a client to delete a TierDomain. If TierDomain is associated to storage tiers, the call will fail. However, if the Force parameter is set to true, the associated storage tiers will also be deleted unless the associated storage tiers are also associated to other TierDomains.

31.2.7 ModifyStorageTierDomainAssociation

```
            uint32 TierService.ModifyStorageTierDomainAssociation(
                  [IN]  CIM_AdminDomain REF RemoveFromTierDomain,
                  [IN]  CIM_AdminDomain REF AddToTierDomain,
                  [IN, EmbeddedInstance ( "CIM_TierSettingData" )]
                        string TierSettingData,
                  [OUT] CIM_ConcreteJob REF Job,
                  [IN]  CIM_StorageTier REF StorageTier );
```

This method allows a client to modify associations between a storage tier and tier domains.

If the parameter RemoveFromTierDomain is null, the storage tier will not be removed from an existing TierDomain. If the parameter AddToTierDomain is null, the storage tier will not be added to a TierDomain.

For example, to just add a storage tier to an existing TierDomain, do not supply the RemoveFromTierDomain parameter, but supply the parameter AddToTierDomain. Alternatively, to just remove a storage tier from a TierDomain without adding it to another TierDomain, supply the RemoveFromTierDomain, but do not supply the AddToTierDomain.

31.2.8 GetStorageTierCandidateObjects

```
uint32 TierService.GetStorageTierCandidateObjects(
      [IN]  CIM_LogicalElement REF InElements[],
      [IN, EmbeddedInstance ( "CIM_TierSettingData" )]
            string TierSettingData,
      [IN]  CIM_ManagedElement REF Goal,
      [OUT] CIM_ConcreteJob REF Job,
      [OUT] CIM_LogicalElement REF Candidates[] );
```

This method returns an array of storage objects that can form a storage tier. The selection criteria is based on the input parameters. The more input parameters, the narrower the search criteria. For example, it is possible to specify to return the candidate storage objects based on all storage extents of a storage pool, i.e., ConcreteComponents, that utilize solid state technology by supplying the appropriate InPools and TierSettingData.

31.3 Client Considerations and Recipes

31.3.1 Automated Storage Tiering

In general, there are two possible implementations of storage tiering. One, where the storage array manages the storage tiers, including creating the storage tiers. Two, all the clients need to do is to create the storage tiers and tier domains.

In both implementations, the storage array monitors the activities of the storage elements (e.g., volumes) and moves the storage elements to the appropriate storage tiers (with StorageTier.RelativePerformanceOrder having a value less than or equal to StorageVolume.RelativePerformanceOrder or one of the values in the RelativePerformanceOrderSet).

The following steps are recommended:

- Review the Discovery section to determine if Storage Tiering is supported. See section 31.1.
- Examine the storage tiering capabilities to determine whether a client needs to create the storage tiers or not. See 31.1.10 "Storage Tiering Capabilities".

31.3.2 Creating StorageVolumes with Storage Tiering

Clients can create a new storage volume and request the newly created storage volume to be placed in an appropriate storage tier. To do so, clients need to create an instance of AdvancedStorageSetting (using the method StorageCapabilities.CreateSetting) and set the properties AdvancedStorageSetting.InitialStorageTieringSelection and AdvancedStorageSetting.RelativePerformanceOrderLimit (or InitialRelativePerformanceOrderSet) as desired, by calling the intrinsic method ModifyInstance. Then, when calling the method StorageConfigurationService.CreateOrModifyElementFromStoragePool, supply the advanced storage setting. Alternatively, clients can set the property AdvancedStorageSetting.InitialStorageTierMethodology and let the system decide the appropriate storage tier. For example, clients can specify a newly created StorageVolume to be placed in a storage tier that has the “Highest Performance” by supplying the argument

AdvancedStorageSetting.InitialStorageTierMethodology=”Highest Performance” to the StorageConfigurationService.CreateOrModifyElementFromStoragePool method.

31.4 CIM Elements

Table 724 describes the CIM elements for Automated Storage Tiering.

Table 724 - CIM Elements for Automated Storage Tiering

Element Name	Requirement	Description
31.4.1 CIM_AdvancedStorageSetting	Optional	Representation of a StorageSetting. StorageSettings are covered in Block Services Package. Additional propperties are added for storage tiering.
31.4.2 CIM_AssociatedElementTier	Conditional	Experimental. Conditional requirement: Required if storage tiering is supported. Define the association between an element and one or more storage tiers.
31.4.3 CIM_AssociatedResourcePool	Optional	Experimental. Associates an object inherited from System to a dependent ResourcePool such as a StoragePool.
31.4.4 CIM_ConcreteDependency (TierDomain to StorageTier)	Optional	Associates storage tiers to tier domains.
31.4.5 CIM_ElementCapabilities	Mandatory	Associates TierServiceCapabilities and TierService.
31.4.6 CIM_HostedService	Mandatory	
31.4.7 CIM_MemberOfCollection (Identifies StorageExtents comprising a tier)	Optional	Associates a storage extent to a storage tier.
31.4.8 CIM_MemberOfCollection (Identifies StoragePools comprising a tier)	Optional	Associates a storage pool to a storage tier.
31.4.9 CIM_MemberOfCollection (Identifies StorageVolumes comprising a tier)	Optional	Identifies a storage volume contributing to a storage tier.
31.4.10 CIM_ServiceAffectsElement (Between TierService and StorageTier)	Optional	Associates TierService to StorageTier.
31.4.11 CIM_ServiceAffectsElement (Between TierService and TierDomain)	Optional	Associates TierService to TierDomain.
31.4.12 CIM_SettingsDefineCapabilities	Optional	Associates StorageTierCapabilities to TierSettingData.
31.4.13 CIM_StorageTier	Mandatory	Experimental. This class represents a collection of storage objects, such as a collection of storage objects identified as a storage tier.
31.4.14 CIM_StorageTierCapabilities	Optional	A subclass of StorageCapabilities that defines the Capabilities of a storage tier.

Table 724 - CIM Elements for Automated Storage Tiering

Element Name	Requirement	Description
31.4.15 CIM_StorageVolume (Constituent)	Conditional	Conditional requirement: Referenced from Pools from Volumes - Constituent StorageVolume is mandatory. The CIM_StorageVolume is an augmented version of the CIM_StorageVolume defined in the Block Services package. See CIM_StorageVolume in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.49 CIM_StorageVolume.
31.4.16 CIM_StorageVolume (Regular)	Mandatory	The CIM_StorageVolume is an augmented version of the CIM_StorageVolume defined in the Block Services package. See CIM_StorageVolume in section *Storage Management Technical Specification, Part 3 Block Devices, 1.6.0 Rev 4* 5.8.49 CIM_StorageVolume.
31.4.17 CIM_SystemComponent (TierDomain to ComputerSystem)	Optional	Storage TierDomains on a system.
31.4.18 CIM_TierDomain	Mandatory	TierDomain representing one or more storage tiers.
31.4.19 CIM_TierService	Mandatory	Experimental. The TierService class provides methods to allow a client to manage storage tiers. Methods are described in the Extrinsic Methods clause.
31.4.20 CIM_TierServiceCapabilities	Mandatory	Experimental. A subclass of Capabilities that defines the Capabilities of a TierService. An instance of TierServiceCapabilities is associated with a TierService using ElementCapabilities.
31.4.21 CIM_TierSettingData	Optional	Experimental. Contains special options for use by methods of TierService.

31.4.1 CIM_AdvancedStorageSetting

Created By: Extrinsic: StorageCapabilities.CreateSetting
Modified By: Static
Deleted By: Static
Requirement: Optional

Table 725 describes class CIM_AdvancedStorageSetting.

Table 725 - SMI Referenced Properties/Methods for CIM_AdvancedStorageSetting

Properties	Flags	Requirement	Description & 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.

Table 725 - SMI Referenced Properties/Methods for CIM_AdvancedStorageSetting

Properties	Flags	Requirement	Description & Notes
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.
InitialStorageTierMethodology	MN	Optional	Enumeration indicating the initial storage tier for the element. "None" or null means this element is not subject to storage tiering. Values: 0: None 3: Implementation Decides 4: Highest Performance 5: Lowest Performance 6: Highest Availability 7: Lowest Availability.
InitialStorageTieringSelection	MN	Optional	Use this value to set the property StorageTieringSelection of the associated element. Storage tiering examines StorageTieringSelection to determine whether to use RelativePerformanceOrder or RelativePerformanceOrderSet for selecting an appropriate storage tier. A value of 0 or null indicates the associated element is not subject to storage tiering. Values: 0: Unknown 2: Use RelativePerformanceOrder 3: Use RelativePerformanceOrderSet.

Table 725 - SMI Referenced Properties/Methods for CIM_AdvancedStorageSetting

Properties	Flags	Requirement	Description & Notes
RelativePerformanceOrderLimit	MN	Optional	The storage tiering limit applied to the element. This property indicates RelativePerformanceOrder not to exceed this value. For example: A system has storage tiers with RelativePerformanceOrder of 1, 3, and 5. If the request is to create a new storage volume with StorageSetting.RelativePerformanceOrderLimit of 2, the newly created storage volume is placed in a storage tier with RelativePerformanceOrder of 1.
InitialRelativePerformanceOrderSet	MN	Optional	A set of values. The associated element can only be placed in storage tiers that have a RelativePerformanceOrder value included in this set.

31.4.2 CIM_AssociatedElementTier

Experimental. Define the association between an element and one or more storage tiers.

Created By: Extrinsic: static
Modified By: Extrinsic: static
Deleted By: Extrinsic: Static
Requirement: Required if storage tiering is supported.

Table 726 describes class CIM_AssociatedElementTier.

Table 726 - SMI Referenced Properties/Methods for CIM_AssociatedElementTier

Properties	Flags	Requirement	Description & Notes
Allocated		Mandatory	Indicates what portion of the element is associated with (allocated from) this storage tier. None: Indicates the element is associated with this storage tier; however, currently none of the element's blocks are allocated from this storage tier. Values: 2: All 3: Partial 4: None.
GroupComponent		Mandatory	A storage tier.
PartComponent		Mandatory	Any element subject to storage tiering.

31.4.3 CIM_AssociatedResourcePool

Experimental. Associates an object inherited from System to a dependent ResourcePool such as a StoragePool.

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

Table 727 describes class CIM_AssociatedResourcePool.

Table 727 - SMI Referenced Properties/Methods for CIM_AssociatedResourcePool

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	A tier domain.
Dependent		Mandatory	A ResourcePool such as StoragePool.

31.4.4 CIM_ConcreteDependency (TierDomain to StorageTier)

Created By: Extrinsic: CreateStorageTier
Modified By: Static
Deleted By: Extrinsic: DeleteStorageTier
Requirement: Optional

Table 728 describes class CIM_ConcreteDependency (TierDomain to StorageTier).

Table 728 - SMI Referenced Properties/Methods for CIM_ConcreteDependency (TierDomain to StorageTier)

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	Tier Domain.
Dependent		Mandatory	A Storage Tier.

31.4.5 CIM_ElementCapabilities

Associates TierServiceCapabilities and TierService.

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

Table 729 describes class CIM_ElementCapabilities.

Table 729 - SMI Referenced Properties/Methods for CIM_ElementCapabilities

Properties	Flags	Requirement	Description & Notes
Capabilities		Mandatory	Instance of CIM_TierServiceCapabilities.
ManagedElement		Mandatory	Instance of CIM_TierService.

31.4.6 CIM_HostedService

Created By: Static

Modified By: Static
Deleted By: Static
Requirement: Mandatory

Table 730 describes class CIM_HostedService.

Table 730 - SMI Referenced Properties/Methods for CIM_HostedService

Properties	Flags	Requirement	Description & Notes
Antecedent		Mandatory	The hosting System.
Dependent		Mandatory	The Tier Service hosted on the System.

31.4.7 CIM_MemberOfCollection (Identifies StorageExtents comprising a tier)

Associates a storage extent to a storage tier.

Created By: Extrinsic: CreateStorageTier
Modified By: Static
Deleted By: Extrinsic: DeleteStorageTier
Requirement: Optional

Table 731 describes class CIM_MemberOfCollection (Identifies StorageExtents comprising a tier).

Table 731 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Identifies Storage-Extents comprising a tier)

Properties	Flags	Requirement	Description & Notes
Collection		Mandatory	Instance of CIM_StorageTier.
Member		Mandatory	A storage extent or a composite storage extent.

31.4.8 CIM_MemberOfCollection (Identifies StoragePools comprising a tier)

Associates a storage pool to a storage tier.

Created By: Extrinsic: CreateStorageTier
Modified By: Static
Deleted By: Extrinsic: DeleteStorageTier
Requirement: Optional

Table 732 describes class CIM_MemberOfCollection (Identifies StoragePools comprising a tier).

Table 732 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Identifies Storage-Pools comprising a tier)

Properties	Flags	Requirement	Description & Notes
Collection		Mandatory	Instance of CIM_StorageTier.
Member		Mandatory	A storage pool.

31.4.9 CIM_MemberOfCollection (Identifies StorageVolumes comprising a tier)

Identifies a storage volume contributing to a storage tier.

Created By: Extrinsic: CreateStorageTier
Modified By: Static
Deleted By: Extrinsic: DeleteStorageTier
Requirement: Optional

Table 733 describes class CIM_MemberOfCollection (Identifies StorageVolumes comprising a tier).

Table 733 - SMI Referenced Properties/Methods for CIM_MemberOfCollection (Identifies StorageVolumes comprising a tier)

Properties	Flags	Requirement	Description & Notes
Collection		Mandatory	Instance of CIM_StorageTier.
Member		Mandatory	A storage volume.

31.4.10 CIM_ServiceAffectsElement (Between TierService and StorageTier)

Associates TierService to StorageTier.

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

Table 734 describes class CIM_ServiceAffectsElement (Between TierService and StorageTier).

Table 734 - SMI Referenced Properties/Methods for CIM_ServiceAffectsElement (Between TierService and StorageTier)

Properties	Flags	Requirement	Description & Notes
AffectingElement		Mandatory	Tier Service.
AffectedElement		Mandatory	Storage Tier.

31.4.11 CIM_ServiceAffectsElement (Between TierService and TierDomain)

Associates TierService to TierDomain.

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

Table 735 describes class CIM_ServiceAffectsElement (Between TierService and TierDomain).

Table 735 - SMI Referenced Properties/Methods for CIM_ServiceAffectsElement (Between TierService and TierDomain)

Properties	Flags	Requirement	Description & Notes
AffectingElement		Mandatory	Tier Service.
AffectedElement		Mandatory	Tier Domain.

31.4.12 CIM_SettingsDefineCapabilities

Associates StorageTierCapabilities to TierSettingData.

Requirement: Optional

Table 736 describes class CIM_SettingsDefineCapabilities.

Table 736 - SMI Referenced Properties/Methods for CIM_SettingsDefineCapabilities

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	Reference to StorageTierCapabilities.
PartComponent		Mandatory	Reference to TierSettingData.

31.4.13 CIM_StorageTier

Experimental. This class represents a collection of storage objects, such as a collection of storage objects identified as a storage tier.

Created By: Extrinsic: CreateStorageTier
Modified By: Static
Deleted By: Extrinsic: DeleteStorageTier
Requirement: Mandatory

Table 737 describes class CIM_StorageTier.

Table 737 - SMI Referenced Properties/Methods for CIM_StorageTier

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	Within the scope of an array, the InstanceID opaquely and uniquely identifies an instance of this class.
ElementName		Optional	User-friendly name.
RelativePerformance Order		Mandatory	A number starting from 0 to indicate the relative performance characteristics of the storage tier. The smaller the number, the higher the performance characteristics.

Table 737 - SMI Referenced Properties/Methods for CIM_StorageTier

Properties	Flags	Requirement	Description & Notes
StorageObjectType		Optional	indicates the type of storage objects comprising a storage tier. Values: 2: StorageExtent 3: StoragePool 4: StorageVolume.
Technology		Optional	The technology of the underlying disk drives used. Mixed: a storage tier consists of a mix of different disk drive technologies. Values: 2: Not Applicable 3: Solid State Drive 4: Fibre Channel 5: SATA 6: SAS 7: Mixed 8: Hard Disk Drive.
State	M	Optional	This property indicates whether the storage tier is actively being used or not. Values: 2: Enabled 3: Disabled.
Dynamic	M	Optional	If true any new storage objects added to the system that have a similar performance characteristics (and QoS) to this tier become part of this tier automatically. If false, after the storage tier is created, any newly introduced storage objects need to be added to the storage tier manually.
DeleteOnEmptyElement	M	Optional	If true and empty storage tiers are allowed, the storage tier will be deleted when the last element is removed from the storage tier. If empty storage tiers are not allowed, the storage tier will be deleted automatically when the storage tier becomes empty.
Percentage	M	Optional	A value between 0 to 100 to indicate the maximum percentage of the underlying capacity that can be used for storage tiering activities. For example, a value of 80 indicates no more than 80 percent of the storage tier can be used for automated tiering. The remaining 20 percent of the underlying storage is set aside for elements that do not participate in automated storage tiering.
TotalCapacity		Optional	The total capacity of the storage tier in bytes.

31.4.14 CIM_StorageTierCapabilities

A subclass of StorageCapabilities that defines the Capabilities of a storage tier.

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

Table 738 describes class CIM_StorageTierCapabilities.

Table 738 - SMI Referenced Properties/Methods for CIM_StorageTierCapabilities

Properties	Flags	Requirement	Description & 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).
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.

Table 738 - SMI Referenced Properties/Methods for CIM_StorageTierCapabilities

Properties	Flags	Requirement	Description & Notes
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.
StorageTierCharacteristics		Optional	Indicates the storage tiering capabilities. Values: 2: BasedOnPerformance Only 3: BasedOnQOS Only 4: BasedOnPerformance And BasedOnQOS.
RelativePerformanceOrderMin		Optional	Indicates the minimum value for RelativePerformanceOrder that this storage tier can have. The minimum value for RelativePerformanceOrder, which represents the highest level of performance.

Table 738 - SMI Referenced Properties/Methods for CIM_StorageTierCapabilities

Properties	Flags	Requirement	Description & Notes
RelativePerformanceOrderMax		Optional	Indicates the maxmimum value for RelativePerformanceOrder that this storage tier can have. The maximum value for RelativePerformanceOrder, which represents the lowest level of performance.
RelativePerformanceOrderDefault		Optional	Indicates the default value of RelativePerformanceOrder for the storage tier -- the smaller the RelativePerformanceOrder, the more performant the storage tier.
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.

31.4.15 CIM_StorageVolume (Constituent)

Created By: Extrinsic: StorageConfigurationService.CreateOrModifyElementFromStoragePool
Modified By: Static
Deleted By: Extrinsic: StorageConfigurationService.ReturnToStoragePool
Requirement: Referenced from Pools from Volumes - Constituent StorageVolume is mandatory.

Table 739 describes class CIM_StorageVolume (Constituent).

Table 739 - SMI Referenced Properties/Methods for CIM_StorageVolume (Constituent)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	Opaque identifier.
ElementName		Optional	User-friendly name.

Table 739 - SMI Referenced Properties/Methods for CIM_StorageVolume (Constituent)

Properties	Flags	Requirement	Description & Notes
Name	CD	Mandatory	Identifier for this volume; based of datapath standards such as SCSI or ATAPI.
OtherIdentifyingInfo	CD	Optional	Additional correlatable names.
IdentifyingDescriptions		Optional	
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). In addition, the secondary OperationalStatus may be 19 (Relocating) with 2|3|15 (OK or Degraded 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	

Table 739 - SMI Referenced Properties/Methods for CIM_StorageVolume (Constituent)

Properties	Flags	Requirement	Description & Notes
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:Pool Component'.
CanDelete		Optional	Experimental. Indicates if the volume is able to be deleted by a client application.
StorageTieringSelection	N	Optional	Storage tiering examines this property to determine whether to use RelativePerformanceOrder or RelativePerformanceOrderSet for selecting an appropriate storage tier. A value of 0 or null indicates the element is not subject to storage tiering. Values: 0: None 2: Use RelativePerformanceOrder 3: Use RelativePerformanceOrderSet.
RelativePerformanceOrder	N	Optional	A storage volume can be associated with one or more storage tiers with equal or smaller StorageTier.RelativePerformanceOrder. The smaller the RelativePerformanceOrder, the more performant the tier.
RelativePerformanceOrderSet	N	Optional	A set of RelativePerformanceOrder values. The volume can only be associated with storage tiers that have a RelativePerformanceOrder value included in this set.

31.4.16 CIM_StorageVolume (Regular)

Created By: Extrinsic: StorageConfigurationService.CreateOrModifyElementFromStoragePool
Modified By: Static
Deleted By: Extrinsic: StorageConfigurationService.ReturnToStoragePool
Requirement: Mandatory

Table 740 describes class CIM_StorageVolume (Regular).

Table 740 - SMI Referenced Properties/Methods for CIM_StorageVolume (Regular)

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
DeviceID		Mandatory	Opaque identifier.
ElementName		Optional	User-friendly name.
Name	CD	Mandatory	Identifier for this volume; based of datapath standards such as SCSI or ATAPI.
OtherIdentifyingInfo	CD	Optional	Additional correlatable names.
IdentifyingDescriptions		Optional	
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). In addition, the secondary OperationalStatus may be 19 (Relocating) with 2|3|15 (OK or Degraded or Dormant).
BlockSize		Mandatory	

Table 740 - SMI Referenced Properties/Methods for CIM_StorageVolume (Regular)

Properties	Flags	Requirement	Description & Notes
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'.
CanDelete		Optional	Experimental. Indicates if the volume is able to be deleted by a client application.
StorageTieringSelection	N	Optional	Storage tiering examines this property to determine whether to use RelativePerformanceOrder or RelativePerformanceOrderSet for selecting an appropriate storage tier. A value of 0 or null indicates the element is not subject to storage tiering. Values: 0: None 2: Use RelativePerformanceOrder 3: Use RelativePerformanceOrderSet.
RelativePerformanceOrder	N	Optional	A storage volume can be associated with one or more storage tiers with equal or smaller StorageTier.RelativePerformanceOrder. The smaller the RelativePerformanceOrder, the more performant the tier.
RelativePerformanceOrderSet	N	Optional	A set of RelativePerformanceOrder values. The volume can only be associated with storage tiers that have a RelativePerformanceOrder value included in this set.

31.4.17 CIM_SystemComponent (TierDomain to ComputerSystem)

Storage TierDomains on a system.

Created By: Extrinsic: CreateTierDomain
Modified By: Static
Deleted By: Extrinsic: DeleteTierDomain
Requirement: Optional

Table 741 describes class CIM_SystemComponent (TierDomain to ComputerSystem).

Table 741 - SMI Referenced Properties/Methods for CIM_SystemComponent (TierDomain to ComputerSystem)

Properties	Flags	Requirement	Description & Notes
GroupComponent		Mandatory	Reference to top-level ComputerSystem.
PartComponent		Mandatory	Reference to the TierDomain.

31.4.18 CIM_TierDomain

TierDomain representing one or more storage tiers.

Created By: Extrinsic: CreateTierDomain
Modified By: Static
Deleted By: Extrinsic: DeleteTierDomain
Requirement: Mandatory

Table 742 describes class CIM_TierDomain.

Table 742 - SMI Referenced Properties/Methods for CIM_TierDomain

Properties	Flags	Requirement	Description & Notes
CreationClassName		Mandatory	Name of Class.
Name		Mandatory	An arbitrary name (implementation dependent).
NameFormat		Mandatory	Dependent on the arbitrary name chosen.
ElementName		Optional	A user friendly name for the storage tier domain (implementation dependent).
OtherIdentifyingInfo		Mandatory	For a storage tier domain, this property shall contain the value 'TIER'.
IdentifyingDescriptions		Mandatory	For a storage TIER AdminDomain, this property shall contain the value 'SNIA:DetailedType' in the index for the OtherIdentifyingInfo of 'TIER'.

31.4.19 CIM_TierService

Experimental. Base class for Automatic Storage Tiering.

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

Table 743 describes class CIM_TierService.

Table 743 - SMI Referenced Properties/Methods for CIM_TierService

Properties	Flags	Requirement	Description & Notes
SystemCreationClassName		Mandatory	
SystemName		Mandatory	
CreationClassName		Mandatory	
Name		Mandatory	
CreateStorageTier()		Conditional	Conditional requirement: Required if client manages storage tiers.
DeleteStorageTier()		Conditional	Conditional requirement: Required if client manages storage tiers.
AddToStorageTier()		Conditional	Conditional requirement: Required if client manages storage tiers.
RemoveFromStorageTier()		Conditional	Conditional requirement: Required if client manages storage tiers.
CreateTierDomain()		Conditional	Conditional requirement: Required if client manages tier domains.
DeleteTierDomain()		Conditional	Conditional requirement: Required if client manages tier domains.
ModifyStorageTierDomainAssociation()		Conditional	Conditional requirement: Required if client manages tier domains.
GetStorageTierCandidateObjects()		Conditional	Conditional requirement: Required if client manages tier domains.

31.4.20 CIM_TierServiceCapabilities

Experimental. A subclass of Capabilities that defines the Capabilities of a TierService. An instance of TierServiceCapabilities is associated with a TierService using ElementCapabilities.

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

Table 744 describes class CIM_TierServiceCapabilities.

Table 744 - SMI Referenced Properties/Methods for CIM_TierServiceCapabilities

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
ElementName		Mandatory	User Friendly name.

Table 744 - SMI Referenced Properties/Methods for CIM_TierServiceCapabilities

Properties	Flags	Requirement	Description & Notes
SupportedFeatures		Mandatory	Enumeration indicating the supported features of the storage tiering service. Values: 2: System Creates StorageTiers 3: Client Can Create StorageTiers 4: System Creates TierDomains 5: Client Can Create TierDomains 6: Supports SubLUN.
SupportedTierFeatures		Mandatory	Enumeration indicating the supported features of the storage tiers. Values: 2: StorageTiers Based On Performance Only 3: StorageTiers Based On QoS Only 4: StorageTiers Based On Performance and QoS 5: StorageTiers Based On Other Characteristics 6: StorageTiers Can Overlap 7: StorageTiers Can Be Empty 8: StorageTiers In Multiple TierDomains.
SupportedStorageObjects		Mandatory	Enumeration indicating the supported storage objects that can be used to form a storage tier. ComponentsOfStoragePool: In calling the method CreateStorageTier, it is possible to supply one or StoragePools and request the storage tiers to be created based on the "ConcreteComponents" of the StoragePools, i.e. StorageExtents. ElementsOfStoragePool: Storage tier is comprised of elements, e.g. volumes, allocated from a StoragePool. Values: 2: StorageVolume 3: LogicalDisk 4: StorageExtent 5: StoragePool 6: ComponentsOfStoragePool 7: ElementsOfStoragePool.

Table 744 - SMI Referenced Properties/Methods for CIM_TierServiceCapabilities

Properties	Flags	Requirement	Description & Notes
SupportedAsynchronousActions		Mandatory	Identify methods using job control. Values: 2: CreateStorageTier 3: AddToStorageTier 4: RemoveFromStorageTier 5: CreateTierDomain 6: DeleteTierDomain 7: ModifyStorageTierDomainAssociation 8: GetStorageTierCandidateObjects.
SupportedSynchronousActions		Mandatory	Identify methods not using job control. Values: 2: CreateStorageTier 3: AddToStorageTier 4: RemoveFromStorageTier 5: CreateTierDomain 6: DeleteTierDomain 7: ModifyStorageTierDomainAssociation 8: GetStorageTierCandidateObjects.

31.4.21 CIM_TierSettingData

Experimental. Contains special options for use by methods of TierService.

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

Table 745 describes class CIM_TierSettingData.

Table 745 - SMI Referenced Properties/Methods for CIM_TierSettingData

Properties	Flags	Requirement	Description & Notes
InstanceID		Mandatory	
ElementName		Mandatory	User Friendly name.
RelativePerformanceOrderGoal	MN	Optional	An integer starting from 0 to indicate the performance characteristic of the storage tier. The smaller the number, the higher the performance characteristics.

Table 745 - SMI Referenced Properties/Methods for CIM_TierSettingData

Properties	Flags	Requirement	Description & Notes
RelativePerformance OrderMin	MN	Optional	Indicates the minimum value for RelativePerformanceOrder that this storage tier can have. Basically, the minimum value for RelativePerformanceOrder represents the highest level of performance.
RelativePerformance OrderMax	MN	Optional	Indicates the maxmimum value for RelativePerformanceOrder that this storage tier can have. Basiclly, the maximum value for RelativePerformanceOrder represents the lowest level of performance.
StorageTierCharacter istics	MN	Optional	Indicates the storage tier charateristics. Performance generally relates to the underlying technology, for example, Solid State versus Fibre Channel drives. QoS refers to the Quality of Service, for example, RAID protected versus unprotected. Values: 2: BasedOnPerformance Only 3: BasedOnQOS Only 4: BasedOnPerformance And BasedOnQOS.
Technology	MN	Optional	The technology of the underlying disk drives used. Not Applicable: Storage tier is not based on technology of underlying components. Mixed: a storage tier consists of a mix of different disk drive technologies. Values: 2: Not Applicable 3: Solid State Drive 4: Fibre Channel 5: SATA 6: SAS 7: Mixed 8: Hard Disk Drive.
InitialState	MN	Optional	This property indicates the intial state of the storage tier. The default value is 2. Values: 2: Enabled 3: Disabled.
Dynamic	MN	Optional	If true any new extents added to the system that have a similar performance characteristics (and QoS) to this tier become part of this tier automatically. If false, the new extents needs to be added to this tier manually. The default value is false.

Table 745 - SMI Referenced Properties/Methods for CIM_TierSettingData

Properties	Flags	Requirement	Description & Notes
DeleteOnEmptyElement	MN	Optional	If true and empty storage tier are allowed, the storage tier will be deleted when the last element is removed from the storage tier. If empty storage tier are not allowed, the storage tier will be deleted automatically when the storage tier becomes empty. The default value is false.
Percentage	MN	Optional	A value between 0 to 100 to indicate the maximum percentage of the underlying capacity that can be used for storage tiering activities. For example, a value of 80 indicates no more than 80 percent of the storage tier can be used for automated tiering. The remaining 20 percent of the underlying storage is set aside for elements that do not participate in automated storage tiering.

Annex A (informative) SMI-S Information Model

This standard is based on DMTF’s CIM schema, version 2.29. The DMTF schema is available in the machinereadable Managed Object Format (MOF) format. DMTF MOFs are simultaneously released both as an "Experimental" and a "Final" version of the schema. This provides developers with early access to experimental parts of the models. Both versions are available at

http://www.dmtf.org/standards/cim/cim_schema_v2290

Most SMI-S Profiles are primarily based on the DMTF Final MOFs. Content marked as “Experimental” or “Implemented” may be based on DMTF’s Experimental MOFs. Some SMI-S Experimental Profiles may also use classes with a SNIA_ prefix; MOFs from these classes are available from SNIA.

Annex B (informative) Registry of StorageExtent Definitions

EXPERIMENTAL

Table B.1 lists a registry of StorageExtent definitions in SMI-S and the properties that distinguish the extents from each other.

Table B.1 - Registry of StorageExtent Definitions

Extent (Usage)	Profile	Primordial	ExtentDiscriminator
StorageExtent (Intermediate)	Extent Composition	'false'	'SNIA:Intermediate'
StorageExtent (Pool Component)	Extent Composition	'false'	'SNIA:Pool Component'
CompositeExtent (Composite Intermediate)	Extent Composition	'false'	'SNIA:Intermediate' and 'SNIA:Composite'
CompositeExtent (Composite Pool Component)	Extent Composition	'false'	'SNIA:Pool Component' and 'SNIA:Composite'
StorageExtent (Remaining)	Extent Composition	'false'	'SNIA:Remaining'
StorageExtent (Primordial Disk Drive Extent)	Disk Drive Lite	'true'	'SNIA:Pool Component' and 'SNIA:DiskDrive'
StorageExtent (Imported Extents)	Storage Virtualizer	'true'	'SNIA:Pool Component' and 'SNIA:Imported'
StorageExtent (Spare)	Disk Sparing	either	'SNIA:Spare'
StorageVolume (Allocated)	Block Services, Disk Sparing	'false'	'SNIA:Allocated'
LogicalDisk (Allocated)	Block Services, Disk Sparing	'false'	'SNIA:Allocated'
StorageVolume (Constituent)	Pools from Volumes	'false'	'SNIA:Pool Component'
StorageVolume (Shadow)	Storage Virtualizer, NAS Head, Replication Services	'false'	'SNIA:Shadow'
LogicalDisk (Shadow)	Host Filesystem	'false'	'SNIA:Shadow'
LogicalDisk (Filesystem)	NAS Head, Self-contained NAS	'false'	'SNIA:Allocated' and 'SNIA:Reserved'
LogicalDisk (Intermediate)	Volume Management	'false'	'SNIA:Intermediate'
LogicalDisk (Primordial)	Volume Management	'true'	'SNIA:Imported'

These definitions are not mutually exclusive. That is, a single StorageExtent instance may satisfy multiple of these definitions. For example, it would not be uncommon for a StorageExtent (Primordial Disk Drive Extent) to also be a StorageExtent (Spare). However, some are mutually exclusive. For example, all the extents defined in Extent Composition are mutually exclusive with the StorageExtent (Primordial Disk Drive Extent). The Extent Composition extents all have Primordial='false' and the StorageExtent (Primordial Disk Drive Extent) has Primordial='true'. So an instance cannot be both a disk drive StorageExtent and an Extent Composition storage extent. The known valid combinations are discussed in section B.3.

B.1 ExtentDiscriminator Definitions

The Values for ExtentDiscriminator are defined as follows:

SNIA:Pool Component - A StorageExtent (or CompositeExtent) that represents storage of a StoragePool, but is not a remaining extent.

SNIA:Intermediate - A StorageExtent (or CompositeExtent) that is neither a Pool Component nor a Remaining Extent (it does not represent storage in the pool, remaining or otherwise).

SNIA:Composite - A StorageExtent that is a CompositeExtent.

SNIA:Remaining - A StorageExtent that has an AssociatedRemainingExtent to a StoragePool (representing free storage in the StoragePool).

SNIA:DiskDrive - A StorageExtent that is the media on a Disk Drive.

SNIA:Imported - A StorageExtent that is imported from an external source.

SNIA:Spare - A StorageExtent that acts as a spare for other StorageExtents (and has the IsSpare association).

SNIA:Allocated - A StorageExtent that is subclassed to StorageVolume or LogicalDisk, and has an AllocatedFromStoragePool association from a Concrete StoragePool.

SNIA:Shadow - A StorageExtent (or subclass) that represents a StorageExtent in another autonomous profile (e.g., the StorageVirtualizer has StorageVolumes (Shadow) that represent StorageVolumes exported by Arrays).

SNIA:Reserved - A StorageExtent that is reserved for some system use within the autonomous profile (e.g., in NAS profiles, an Allocated LogicalDisk is reserved for holding Filesystems).

B.2 Association Significance of the Various Extent Definitions

Each of the Extent Definitions has significance relative to the associations that may exist for the Extent definition. This section lists the associations implied by the various definitions.

B.2.1 StorageExtent (Intermediate)

An intermediate StorageExtent has the following associations defined on it:

- The Antecedent of a BasedOn Association from a StorageVolume (or LogicalDisk) (Optional)
- The Antecedent of a "mid level" BasedOn association (Optional)
- The Dependent of a "mid level" BasedOn association (Optional)
- The Antecedent on a CompositeExtentBasedOn (Optional)

B.2.2 StorageExtent (Pool Component)

A pool component StorageExtent has the following associations defined on it:

- The PartComponent of a ConcreteComponent to a "concrete" StoragePool (Mandatory, but Deprecated)
- The PartComponent of a AssociatedComponentExtent to a "concrete" StoragePool (Mandatory)
- The Antecedent of a BasedOn Association from a StorageVolume (or LogicalDisk) (Optional)
- The Antecedent of a "mid level" BasedOn association (Optional)

- The Dependent of a "mid level" BasedOn association (Optional)
- The Antecedent on a CompositeExtentBasedOn (Optional)

B.2.3 CompositeExtent (Composite Intermediate)

An intermediate CompositeExtent has the following associations defined on it:

- The Dependent on a CompositeExtentBasedOn (Optional)
- The Dependent on a BasedOn in striping cases (Optional)
- The Antecedent of a BasedOn Association from a StorageVolume (or LogicalDisk) (Optional)
- The Antecedent of a "mid level" BasedOn association (Optional)
- The Antecedent on a CompositeExtentBasedOn (Optional)

B.2.4 CompositeExtent (Composite Pool Component)

A pool component CompositeExtent has the following associations defined on it:

- The PartComponent of a ConcreteComponent to a "concrete" StoragePool (Mandatory, but Deprecated)
- The PartComponent of a AssociatedComponentExtent to a "concrete" StoragePool (Mandatory)
- The Dependent on a CompositeExtentBasedOn (Mandatory)
- The Antecedent of a BasedOn Association from a StorageVolume (or LogicalDisk) (Optional)
- The Antecedent of a "mid level" BasedOn association (Optional)
- The Antecedent on a CompositeExtentBasedOn (Optional)

B.2.5 StorageExtent (Remaining)

A remaining StorageExtent has the following associations defined on it:

- The Dependent of a "mid level Remaining" BasedOn association (Mandatory)
- The PartComponent of a AssociatedRemainingExtent to a "concrete" StoragePool (Mandatory)
- The PartComponent of a ConcreteComponent to a StoragePool (Mandatory, but Deprecated)

B.2.6 StorageExtent (Primordial Disk Drive Extent)

A Disk drive StorageExtent has the following associations defined on it:

- The Dependent of a MediaPresent to DiskDrive (Mandatory)
- The PartComponent of a SystemDevice to a ComputerSystem (Mandatory)
- The PartComponent of a ConcreteComponent to a "Primordial" StoragePool (Mandatory, but Deprecated)
- The Antecedent of a "Bottom level" BasedOn association (Conditional on Extent Composition)
- The PartComponent of a AssociatedComponentExtent to a "Primordial" StoragePool (Conditional on Extent Composition)
- The Dependent of a ProtocolControllerAccessesUnit to ProtocolController (Optional)
- The LogicalUnit of a SCSIInitiatorTargetLogicalUnitPath to Initiator & Target ProtocolEndpoints (Optional)

B.2.7 StorageExtent (Imported Extents)

An imported StorageExtent has the following associations defined on it:

- The PartComponent of a SystemDevice to a ComputerSystem (Mandatory)
- The PartComponent of a ConcreteComponent to a "Primordial" StoragePool (Mandatory, but Deprecated)
- The Antecedent of a "Bottom level" BasedOn association (Conditional on Extent Composition)
- The PartComponent of a AssociatedComponentExtent to a "Primordial" StoragePool (Conditional on Extent Composition)

B.2.8 StorageExtent (Spare)

A spare StorageExtent has the following associations defined on it:

- The Antecedent of an IsSpare association (Mandatory)
- The Antecedent of a Spared association (Mandatory)
- The PartComponent of a AssociatedComponentExtent to a "Primordial" StoragePool (Conditional on Extent Composition)
- The PartComponent of a ConcreteComponent to a StoragePool

B.2.9 StorageVolume (Allocated)

An Allocated StorageVolume has the following associations defined on it:

- The ManagedElement of an ElementSettingData to a StorageSetting (Mandatory)
- The Dependent of an AllocatedFromStoragePool to a "Concrete" StoragePool (Mandatory)
- The PartComponent of a SystemDevice to a ComputerSystem (Mandatory)
- The ManagedElement of an ElementCapabilities to a StorageCapabilities (Optional)

B.2.10 LogicalDisk (Allocated)

An Allocated LogicalDisk has the following associations defined on it:

- The ManagedElement of an ElementSettingData to a StorageSetting (Mandatory)
- The Dependent of an AllocatedFromStoragePool to a "Concrete" StoragePool (Mandatory)
- The PartComponent of a SystemDevice to a ComputerSystem (Mandatory)
- The ManagedElement of an ElementCapabilities to a StorageCapabilities (Optional)

B.2.11 StorageVolume (Pool Component)

A Pool Component StorageVolume has the following associations defined on it:

- The Dependent of an AllocatedFromStoragePool to a "Concrete" StoragePool (Mandatory)
- The PartComponent of a SystemDevice to a ComputerSystem (Mandatory)

B.2.12 StorageVolume (Shadow)

A Shadow StorageVolume has the following associations defined on it:

- The PartComponent of a SystemDevice to a "Shadow" ComputerSystem (Mandatory)

- A SystemElement of a LogicalIdentity to an "Imported" StorageExtent (Mandatory)
- A member of a MemberOfCollection to a SNIA_AllocatedResources (Mandatory)
- A member of a MemberOfCollection to a SNIA_RemoteResources (Optional)

B.2.13 LogicalDisk (Shadow)

A Shadow LogicalDisk has the following associations defined on it:

- The PartComponent of a SystemDevice to a "Shadow" ComputerSystem (Mandatory)
- A SystemElement of a LogicalIdentity to an "Imported" StorageExtent (Mandatory)
- A member of a MemberOfCollection to a SNIA_AllocatedResources (Mandatory)
- A member of a MemberOfCollection to a SNIA_RemoteResources (Optional)

B.3 Example Valid Combinations of Extent Definitions

Table B.2 shows the known valid combinations of Extent Definitions. These refer to StorageExtent instances that have multiple Usages.

Table B.2 - Example Valid Combinations of Extent Definitions

Extent Usage	Extent Usage	Primordial	ExtentDiscriminators	Notes
Primordial Disk Drive Extent	Spare	'true'	'SNIA:Pool Component', 'SNIA:DiskDrive' and 'SNIA:Spare'	A disk drive extent may be a spare.
Imported Extents	Spare	'true'	'SNIA:Pool Component', 'SNIA:Imported' and 'SNIA:Spare'	An imported extent may be a spare.
Composite Pool Component	Spare	'false'	'SNIA:Pool Component', 'SNIA:Composite' and 'SNIA:Spare'	A composite Pool component (a concrete extent) may be a spare
Pool Component	Spare	'false'	'SNIA:Pool Component' and 'SNIA:Spare'	A Pool Component (a concrete extent) may be a spare

B.4 Combinations of Extent Definitions not defined in this Release of the Standard

Currently, this release of the standard does not directly or indirectly support the combinations of Extent Definitions. Some example of combinations not defined in this standard are identified in Table B.3.

Table B.3 - Extent Combinations not defined in this Release of the Standard

Extent Usage	Extent Usage	Primordial	ExtentDiscriminators	Notes
Primordial Disk Drive Extent	Intermediate	Conflicted	'SNIA:Intermediate' and 'SNIA:DiskDrive'	An Intermediate Extent is always a concrete extent
Imported Extents	Intermediate	Conflicted	'SNIA:Intermediate' and 'SNIA:Imported'	An Intermediate Extent is always a concrete extent

Table B.3 - Extent Combinations not defined in this Release of the Standard

Extent Usage	Extent Usage	Primordial	ExtentDiscriminators	Notes
Remaining	Intermediate	‘false’	'SNIA:Intermediate' and 'SNIA:Remaining'	An Intermediate Extent is used to represent storage that is in use (and remaining is free space).
Remaining	Pool Component	‘false’	'SNIA:Pool Component' and 'SNIA:Remaining'	A Remaining Extent represents unallocated storage in a Pool and cannot be a component of a Pool.
Primordial Disk Drive Extent	Composite Pool Component	Conflicted	'SNIA:Pool Component', 'SNIA:Composite' and 'SNIA:DiskDrive'	An Composite Extent is always a concrete extent and a drive extent is primordial.
Imported Extents	Composite Pool Component	Conflicted	'SNIA:Pool Component', 'SNIA:Composite' and 'SNIA:Imported'	An Composite Extent is always a concrete extent and an imported extent is primordial.
Primordial Disk Drive Extent	Imported Extents	‘true’	'SNIA:DiskDrive' and 'SNIA:Imported'	An extent cannot be both imported and represent a DiskDrive
Spare	Intermediate	‘false’	'SNIA:Spare' and 'SNIA:Intermediate'	This version of the standard only defines sparing of Pool Components
Spare	Remaining	‘false’	'SNIA:Spare' and 'SNIA:Remaining'	This version of the standard only defines sparing of Pool Components

Several of the rows in Table B.3 have the value “Conflicted” in the Primordial column. This means one type of extent is supposed to have the value ‘true’ and the other type of extent is supposed to have the value ‘false’. For example, the standard defines a “Primordial Disk Drive Extent” to always have Primordial=’true’ and a “Composite Pool Component” to always have Primordial=’false’. So a “Primordial Disk Drive Extent” can never be a “Composite Pool Component”.

EXPERIMENTAL